WCCI 2010 IEEE World Congress on Computational Intelligence
July, 18-23, 2010 - CCIB, Barcelona, Spain CEC IEEE

Data Mining Using Parallel Multi-Objective Evolutionary Algorithms on Graphics Hardware

Man-Leung Wong, *Member, IEEE*, and Geng Cui

Abstract— **An important and challenging data mining application in marketing is to learn models for predicting potential customers who contribute large profit to a company under resource constraints. In this paper, we first formulate this learning problem as a constrained optimization problem and then converse it to an unconstrained Multi-objective Optimization Problem (MOP). A parallel Multi-Objective Evolutionary Algorithm (MOEA) on consumer-level graphics hardware is used to handle the MOP. We perform experiments on a real-life direct marketing problem to compare the proposed method with the parallel Hybrid Genetic Algorithm, the DMAX approach, and a sequential MOEA. It is observed that the proposed method is much more effective and efficient than the other approaches.**

I. Introduction

How to improve marketing productivity or the return on marketing investment under resource constraints is one of the most challenging issues facing marketing professionals and researchers. The issue seems to be more pressing in hard economic times and given the increasing emphasis on customer relationship management - containing cost and channeling precious marketing resources to the high value customers who contribute greater profit to a company. Such situations include 1) upgrading customers - how to provide sizable incentives to the customers who are the most likely to upgrade and contribute greater profit over the long run? 2) modeling customer churn or retention - how to identify and prevent the most valuable customers from switching to a competitor? and 3) predicting loan default - how to predict the small minority who default on their large loans or credit card bills? This problem is particularly acute in direct marketing operations that typically have a fixed budget to target, from the vast list of customers in a company's database, those customers who are the most likely to respond to a marketing campaign and purchase greater amounts.

Most marketing activities, as espoused by marketing scholars and practitioners, are targeted marketing in nature - to reach customers who are the most responsive to marketing activities. Until recently, statistical methods such as regression and discriminant analysis have dominated the modeling of consumer responses to marketing activities. These methods, however, suffer from two shortcomings. First, methods based on OLS regression (i.e., mean regression) survey the entire population and focus on the average customer in estimating parameters. Segmentation methods, such as discriminant analyses, are not informative of their marketing responses. Thus, these methods by design are not entirely compatible with the objectives of targeted marketing. Second, researchers have so far focused on modeling either consumer responses or purchase quantity. Few models jointly consider consumers responses and the revenue/profit that they generate.

Man-Leung Wong is with the Department of Computing and Decision Sciences, Lingnan University, Tuen Mun, Hong Kong (phone: 852 2616 8093 email: mlwong@ln.edu.hk).

Geng Cui is with the Department of Marketing and International Business, Lingnan University, Tuen Mun, Hong Kong (phone: 852 2616 8245 email: gcui@ln.edu.hk).

These problems are particularly acute in modeling consumer responses to direct marketing and result in suboptimal performance of marketing campaigns. Conventional methods generate the predicted purchase probabilities for the entire population and do not focus on the top portion of the population, e.g. the top 20% most attractive customers. This constraint is crucial as most firms have a limited marketing budget and can only target the most attractive customers. Thus, improving the accuracy of predicting purchase and potential sales or profit for these customers is crucial for augmenting the performance of targeted marketing. This is an urgent research problem given the increasing emphasis on customer relationship management and differentiating customers based on their profitability. Predicting loan default, customer churning, and service intervention represent other situations where resources are limited, budget constraints need to be considered, and targeted efforts are required.

To improve the accuracy of customer selection for targeted marketing, we formulate this problem as a constrained optimization problem. Recently, several researchers suggested using Multi-Objective Evolutionary Algorithms (MOEAs) to solve constrained optimization problems [1], [2].

However, MOEAs may execute for a long time for some difficult problems, because several objective value evaluations must be performed on huge datasets containing information about customers. Moreover, the non-dominance checking and the non-dominated selection procedures are also time consuming. A promising approach to overcome this limitation is to parallelize these algorithms. In this paper, we propose implementing a parallel MOEA for constrained optimization within the CUDA^{TM} (Compute Unified Device Architecture) environment on an nVidia Graphics Processing Unit (GPU). We perform experiments on a real-life direct marketing problem to compare our parallel MOEA with a parallel Hybrid Genetic Algorithm (HGA) [3], the DMAX approach [4], and a sequential MOEA. It is observed that the parallel MOEA is much more effective and efficient than the other approaches. Since consumer-level GPU are available in omnipresent personal computers and these computers are easy to use and manage, more people will be able to use our parallel MOEA to handle different real-life targeted marketing problems.

In the rest of the paper, we first give an overview of constrained optimization for direct marketing, MOEAs, and GPU. In Section III, our parallel MOEA for constrained optimization on GPU is discussed. The experiments and the results are reported in Section IV. In Section V, conclusions and possible future research are discussed.

II. Overview

A. Constrained Optimization for Direct Marketing

Recent emphasis on customer relationship management require marketers to focus on the high profit customers as in the 20/80 principle: 20% of the customers account for 80% profit of a firm. Thus, another purpose of direct marketing models is to predict the amount of purchase or profit from the buyers. However, the distribution of customer sales and profit data is also highly skewed with a very long tail, indicating a concentration of profit among a small group of customers [5]. In empirical studies of profit forecasting, the skewed distribution of profit data also creates problem for researchers. Given the limited and often a fixed marketing budget, the profit maximization approach to customer selection, which include only those customers with an expected marginal profit, is often not realistic [6]. Thus, the ultimate goal of target customer selection is to identify those customers who are the most likely to respond as well as contribute a larger amount of revenue or profit. Overall, unbalanced class distribution and skewed profit data, i.e., the small number of buyers and that of high profit customers remain significant challenges in direct marketing forecasting [7]. Even a small percentage of improvement in the predictive accuracy in terms of customer purchase probability and profit can mean tremendous cost-savings and augment profit for direct marketers.

To date, only a few researchers have considered treating direct marketing forecasting as a problem of constrained optimization. Bhattacharyya [4] applied a genetic algorithm to a linear model that maximizes profitability at a given depth of file using the frontier analysis method. The DMAX model was built for a 10%-of-file mailing. The decile analysis indicates the model has good performance as well as a very good representation of the data as evidenced by the total profit at the top decile. However, a closer look at the decile analysis reveals the model may not be as good as initially believed. The total profit shows unstable performance through the deciles, i.e., profit values do not decrease steadily through the deciles. This unstable performance, which is characterized by major "jumps" in several deciles, indicates the model is inadequately representing the data distribution and may not reliable for decision support. The probable cause for this 'lack-of-fit' is the violation of the assumption of normal distribution in the dependent variable.

Optimization of the classifier does not necessarily lead to maximization of Return On Investment (ROI), since maximization of the true positive rate is often different from the maximization of sales or profit, which determines the ROI under a fixed budget constraint. To solve this problem, Yan and Baldasare [8] proposed an algorithm that uses gradient descent and the sigmoid function to maximize the monetary value under the budget constraint in an attempt to maximize the ROI. By comparison with several classification, regression, and ranking algorithms, they find that their algorithm may result in substantial improvement of the return on investment.

B. Multi-Objective Evolutionary Algorithms

Without loss of generality, we discuss the definitions for minimization multi-objective problems. However, either by using the *duality* principle [9] or by simple modifications to the domination definitions, these definitions and algorithms can be used to handle maximization or combined minimization and maximization problems.

For a multi-objective function Γ from $A(\subseteq \Re^N)$ to a finite set $Y(\subset \Re^m, m \geq 2)$, a decision vector $\vec{a}^* = [a^*(1), a^*(2), \cdots, a^*(N)]^T$ is *Pareto optimal* if and only if for any other decision vector $\vec{a} \in X$, their objective vectors $\vec{y}^* = \Gamma(\vec{a}^*) = [y^*(1), y^*(2), \cdots, y^*(m)]^T$ and $\vec{y} = \Gamma(\vec{x})$ holds either

$$y^*(i) \leq y(i) \text{ for any objective } i \ (1 \leq i \leq m),$$

or there exist two different objectives i, j such that

$$(y^*(i) < y(i)) \wedge (y(j)^* > y(j)) .$$

Thus, for a Pareto optimal decision vector $\vec{a}^*$, there exists no decision vector $\vec{a}$ which would decrease some objective values without causing a simultaneous increase in at least one other objective. These Pareto optimal decision vectors are good tradeoffs for the multi-objective optimization problem. For finding these vectors, dominance in the objective space plays an important role. An objective vector $\vec{y}_1 = \Gamma(\vec{a}_1) = [y_1(1), y_1(2), \cdots, y_1(m)]^T$ *dominates* another objective vector $\vec{y}_2 = \Gamma(\vec{a}_2)$ if and only if the former is partially less than the latter in each objective, i.e.,

$$\begin{cases} y_1(i) \leq y_2(i), & \forall i \in \{1, \cdots, m\} \\ y_1(j) < y_2(j), & \exists j \in \{1, \cdots, m\}. \end{cases} \quad (1)$$

It is denoted as $\vec{y}_1 \prec \vec{y}_2$. Given the set of objective vectors Y, its *Pareto front* Y^* contains all vectors $\vec{y}^* \in Y$ that are not dominated by any other vector $\vec{y} \in Y$. That is, $Y^* = \{\vec{y}^* \in Y | \not\exists \vec{y} \in Y, \vec{y} \prec \vec{y}^*\}$. We call its subset a *Pareto optimal set*. Each $\vec{y}^* \in Y^*$ is *Pareto optimal* or *non-dominated*. A Pareto optimal solution reaches a good tradeoff among these conflicting objectives: one objective cannot be improved without worsening any other objective.

In the general case, it is impossible to find an analytic expression of the Pareto front. The normal procedure to find the Pareto front is to compute the objective values of decision vectors sufficiently enough, and then determine the Pareto optimal vectors to form the Pareto front [10]. However, for many multi-objective optimization problems, the Pareto front Y^* is of substantial size, and the determination of Y^* is computationally prohibitive. Thus, the whole Pareto front Y^* is usually difficult to get and maintain. All practical implementations of MOEAs have maintained a bounded archive of best (non-dominated) solutions found so far [11].

A number of elitist MOEAs have been developed to address diversity of the archived solutions. The diversity exploitation mechanisms include mating restriction, fitness sharing (NPGA [12]), clustering (SPEA [13], SPEA2 [14]), nearest neighbor distance or crowding distance (NSGA-II [15]), crowding count (PAES [16], PESA-II [17], DMOEA [18]), or some preselection operators [9]. Most of them are quite successful, but they cannot ensure convergence to Pareto optimal sets.

C. Graphics Processing Units

The demand from the multimedia and games industries for accelerating 3D rendering has driven several graphics hardware companies devoted to the development of high-performance parallel graphics accelerator. This resulted in the birth of the GPU (Graphics Processing Unit), which handles the rendering requests using a 3D graphics application programming interface (API). Developers can write their own C-like programs, which are called *shaders*, on GPU by using a shading language. Due to the wide availability, programmability, and high-performance of these consumer-level GPUs, they are also cost-effective for many general purpose computations.

Recently, the CUDA$^{\text{TM}}$ technology has been developed [19]. It allows researchers to implement their GPU-based applications more easily. In CUDA, multiple threads can execute the same kernel program in parallel. Threads can access data from multiple memory spaces including the local, shared, global, constant, and texture memory. Because of the Single Instruction Multiple Thread (SIMT) architecture of GPU, certain limitations are imposed. Data-dependent for-loops are not efficient because each thread may perform a different number of iterations. Moreover, the if-then-else construct is also inefficient, as the GPU will execute both true- and false-statements in order to comply with the SIMT design. A number of GPU-based Evolutionary Programming [20], [21], Genetic Algorithms [3], and Genetic Programming [22], [23], [24], [25], [26], [27] have been proposed by researchers.

III. Parallel MOEA for Constrained Optimizatin on Graphics Processing Units

We propose a learning algorithm to handle the constrained optimization and cost-sensitive problems. Let $E = \{e_1, e_2, \cdots, e_K\}$ be the set of K potential customers and $m(e_i)$, $1 \leq i \leq K$, be the amount of money spent by the customer e_i. Assume that $r\%$ of the customers will be solicited. If we can learn a regression function to predict their expected profits or induce a ranking function to arrange the cases in descending order according their expected profits, we can solicit the first $\lceil K * r\% \rceil$ cases to achieve the goal of maximizing the total expected profits of the solicited cases. However, Yan and Baldasare [8] pointed out that this approach tackles an unnecessarily difficult problem and often results in poor performance.

On the other hand, we can learn a scoring function f that divides the K cases into 2 classes: U and L. The sizes of U and L should be $|U|$ and $|L|$, respectively. Consider a case e_i in U, its $f(e_i)$ must be greater than the scoring values of all cases in L. Moreover, the total of the expected profits of all cases e_i in U is maximized. In other words, we can formulate the learning problem as the following constrained optimization problem,

Find a scoring function f that

$$max\{\sum_{e_i \in U} m(e_i)\}, U = \{e_i \in E |\ \nexists e_j \in L[f(e_i) \leq f(e_j)]\} \tag{2}$$

subject to

$$\begin{cases} |U| = \lceil K * r\% \rceil \\ |L| = K - \lceil K * r\% \rceil \end{cases} \tag{3}$$

Since the orderings of all cases in U and all cases in L are insignificant to our objective, it would be easier to learn the scoring function that achieves an optimal partial ranking (ordering) of cases. Although the problem of learning the scoring function is easier, the procedure of finding U and L is still time-consuming because it is necessary to find the (100 - r) percentile of E. Thus, we simplify the above constrained optimization problem to the following constrained optimization problem,

Find a scoring function f and a threshold τ that

$$max\{\sum_{e_i \in U} m(e_i)\}, U = \{e_i \in E | f(e_i) > \tau\} \tag{4}$$

subject to

$$|U| = \lceil K * r\% \rceil \tag{5}$$

We can converse the constrained optimization problem to an unconstrained multi-objective optimization problem with two objectives [2],

$$max\{\sum_{e_i \in U} m(e_i)\}, U = \{e_i \in E | f(e_i) > \tau\} \tag{6}$$

$$min\{maximum(0, |U| - \lceil K * r\% \rceil)\} \tag{7}$$

By limiting f to be a linear function, a MOEA can be used to find the parameters of the scoring function f and the value of τ. We apply a parallel MOEA on GPU [28] that finds a set of non-dominated solutions for a multi-objective function Γ that takes a vector $\vec{a}$ containing the parameters of the scoring function f as well as the value of τ and returns an objective vector $\vec{y}$, where $\vec{a} = [a(1), a(2), \cdots, a(N)]^T$ and $\vec{y} = [y(1), y(2), \cdots, y(m)]^T$. The algorithm is given in Fig. 1.

In the algorithm given in Fig. 1, $\vec{a_i}$ is a vector of variables evolving and $\vec{\eta_i}$ controls the vigorousness of mutation of $\vec{a_i}$. Firstly, an initial population is generated and the objective values of the individuals in the initial population are calculated by using the multi-objective function Γ.

Next, the rankings and the crowding distances of the individuals are found. All non-dominated individuals will

```
1) Set $t$, the generation count, to 0.
2) Generate the initial population $P(t)$ of $\mu$ individuals, each of which
   can be represented as a set of real vectors, $(\vec{a_i}, \vec{\eta_i}), i = 1, \ldots, \mu$.
   Both $\vec{a_i}$ and $\vec{\eta_i}$ contain $N$ independent variables,
       $\vec{a_i} = \{a_i(1), \cdots, a_i(N)\}$
       $\vec{\eta_i} = \{\eta_i(1), \cdots, \eta_i(N)\}$
3) Evaluate the objective vectors of all individuals in $P(t)$ by using the
   multi-objective function.
4) Calculate the rankings and crowding distances of all individuals
     • Execute Dominance-checking($P(t)$, $C$, $S$).
     • Execute Non-dominated-selection($P(t)$, $C$, $S$, $V$, $\mu$).
5) While the termination condition is not satisfied
     a) For $i$ from 1 to $\mu/2$, select two parents $P^1_{parent_i1}$ and
        $P^1_{parent_i2}$ from $P(t)$ using the tournament selection method.
     b) For $i$ from 1 to $\mu/2$, recombine $P^1_{parent_i1}$ and $P^1_{parent_i2}$
        using single point crossover to produce two offspring that are
        stored in the temporary population $P^2$. The population $P^2$
        contains $\mu$ individuals.
     c) Mutate individuals in $P^2$ to generate modified individuals that
        are stored in the temporary population $P^3$. For an individual
        $P^2_i = (\vec{a_i}, \vec{\eta_i})$, where $i = 1, \ldots, \mu$, create a new individual
        $P^3_i = (\vec{a_i}', \vec{\eta_i}')$ as follows:
        for $j = 1, \ldots, N$
            $a_i'(j) = a_i(j) + \eta_i(j)R(0,1)$,
            $\eta_i'(j) = \eta_i(j)\exp(\frac{1}{\sqrt{2N}}R(0,1) + \frac{1}{\sqrt{2\sqrt{N}}}R_j(0,1))$
        where $a_i(j), \eta_i(j), a_i'(j)$, and $\eta_i'(j)$ denote the $j$-th compo-
        nent of $\vec{a_i}, \vec{\eta_i}, \vec{a_i}'$, and $\vec{\eta_i}'$ respectively. $R(0,1)$ denotes a
        normally distributed 1D random number with zero mean and
        standard deviation of one. $R_j$(0,1) indicates a new random
        value for each value of $j$.
     d) Evaluate the objective vectors of all individuals in $P^3$.
     e) Combine the parent population $P(t)$ with $P^3$ to generate a
        population $P^4$ containing $2\mu$ individuals.
     f) Check the dominance of all individuals in $P^4$ by executing
        Dominance-checking($P^4$, $C$, $S$).
     g) Select $\mu$ individuals from $P^4$ and store them in the next
        population $P(t+1)$. The individuals are selected by executing
        Non-dominated-selection($P^4$, $C$, $S$, $V$, $\mu$).
     h) $t = t + 1$.
6) Return the non-dominated individual with the smallest value for the
   second objective in the last population.
```

Fig. 1. The MOEA Algorithm.

be assigned a ranking of 0. The crowding distance of a non-dominated individual is the size of the largest cuboid enclosing it without including any other non-dominated individuals. In order to find the rankings and the crowding distances of other individuals, the non-dominated individuals are assumed to be removed from the population and thus another set of non-dominated individuals can be obtained. The rankings of these individuals should be larger than those of the previous non-dominated individuals. The crowding distances of the individuals can also be found. Similarly, the same approach can be applied to find the rankings and the crowding distances of all other individuals. The procedures Dominance-checking and Non-dominated-selection are used to find these values.

Then, $\mu/2$ pairs of parents will be selected from the population. Two offspring will be generated for each pair of parents by using crossover and mutation. In other words, there will be μ offspring. The objective vectors of all offspring will be obtained and the parent population will be combined with the μ offspring to generate a selection pool. Thus there are 2μ individuals in the selection pool. The rankings and the crowding distances of all individuals in the selection pool can be obtained by using the Dominance-checking and Non-dominated-selection procedures. μ individuals will be selected from the selection pool and they will form the next population of individuals. This evolution process will be repeated until the termination condition is satisfied.

Finally, the non-dominated individual with the smallest value for the second objective in the last population will be returned. In general, the computation of the parallel MOEA can be roughly divided into five types: (I) fitness value evaluation (steps 3 and 5(d)); (II) parent selection (step 5(a)); (III) crossover and mutation (steps 5(b) and 5(c) respectively); (IV) the Dominance-checking procedure designed for parallel algorithms (steps 4(a) and 5(f)); and (V) the Non-dominated-selection procedure which selects individuals from the selection pool (steps 4(b) and 5(g)). The whole MOEA program, except the non-dominated selection procedure, is executed on GPU. Thus, our parallel MOEA gains the most benefit from the SIMT (Single Instruction Multiple Threads) architecture of GPU.

IV. Experiments

In this section, the parallel MOEA is applied to a data mining problem in direct marketing. The objective of the problem is to predict potential prospects from the buying records of previous customers. Advertising campaign, which includes mailing of catalogs or brochures, is then targeted on the group of potential prospects. Hence, if the prediction is accurate, it can help to enhance the response rate of the advertising campaign and increase the return of investment (ROI). The direct marketing problem requires ranking the customer database according to the customers' scores obtained by the prediction models [29].

We compare the parallel MOEA, the parallel HGA [3], and the DMAX approach for learning prediction models. Since the parallel HGA is a single-objective optimization algorithm, it is used to optimize the objective defined in Eq. 6. The experiment test bed was an Intel Pentium Dual$^{\text{TM}}$ E2220 CPU with an affordable PCI Express enabled consumer-level GeForce 9600 GT display card, with 2,048 MB main memory and 512 MB GPU memory. The CPU speed is 2.40 GHz and the GPU contains 64 unified shaders. Microsoft Windows XP Professional, Microsoft Visual C++ 2008, and nVidia CUDA$^{\text{TM}}$ version 2.3 are used to develop the parallel MOEA and the parallel HGA. On the other hand, the DMAX approach is developed in Java. The following parameters have been used in the experiments:

- population size: μ = 256
- tournament size: $q = 2$
- maximum number of generation: $G = 500$
- the percentage of customers to be solicited: $r\%$ = 20%

A. Methodology

The prediction models are evaluated on a large real-life direct marketing dataset from a U.S.-based catalog company. It sells multiple product lines of merchandise including gifts, apparel, and consumer electronics. This dataset contains the records of 106,284 consumers in a recent promotion as well as their purchase history over a twelve-year period. Furthermore, demographic information from the 1995 U.S. Census and credit information from a commercial vendor were appended to the main dataset. Altogether, each record contains 361 variables. The most recent promotion sent a catalog to every customer in this dataset and achieved a 5.4% response rate, representing 5,740 buyers.

Typical in any data mining process, it is necessary to reduce the dimension of the data set by selecting the attributes that are considered relevant and necessary. Towards this feature selection process, there are many possible options. For instance, we could use either a *wrapper* selection process or a *filter* selection process [30]. In a wrapper selection process, different combinations are iteratively tried and evaluated by building an actual model out of the selected attributes. In a filter selection process, certain evaluation function, which is based on information theory or statistics, is defined to score a particular combination of attributes. Then, the final combination is obtained in a search process. In this experiment, we have applied the forward selection method to select 17 variables, that are relevant to prediction, out of the 361 variables.

Since direct marketers usually have a fixed budget and can only contact a small portion of the potential customers in their dataset (e.g., top 20%), simple error rates or overall classification accuracy of models are not meaningful. To support direct marketing and other targeted marketing decisions, maximizing the number of true positives at the top deciles is usually the most important criterion for assessing the performance of prediction models [4], [31].

To compare the performance of different prediction models, we use decile analysis which estimates the enhancement of the response rate and the profit for marketing at different depth-of-file. Essentially, the ranked list is equally divided into ten deciles. Customers in the first decile are the top ranked customers that are most likely to give response and generate high profit. On the other hand, customers in the tenth decile are ranked lowest. To measure the performance of a model at different depths of file, direct marketing researchers have relied on the "lift," which is the ratio of true positives to the total number of records identified by the model in comparison with that of a random model at a specific decile of the file. Thus, comparing the performance of models across depths of file using cumulative lifts or the "response rate" are necessary to inform decisions in direct marketing. Profit lift is the amount of extra profit generated with the new method over that generated by a random method. In this sense, the goal to achieve higher lifts in the upper deciles becomes a ranking problem based on the scores returned by the model and help to evaluate the effectiveness of targeted marketing and to forecast sales and profitability of promotion campaigns.

TABLE I
CUMULATIVE LIFTS OF THE MODELS LEARNED BY DIFFERENT METHODS.

Decile	Parallel MOEA	Parallel HGA	DMAX
0	**358.47 (25.11)**	147.53 (16.84)$^+$	310.60 (34.10)$^+$
1	**270.46 (9.54)**	132.51 (6.98)$^+$	234.20 (20.50)$^+$
2	**219.11 (6.58)**	125.68 (5.82)$^+$	195.50 (12.30)$^+$
3	**182.48 (3.92)**	120.65 (5.88)$^+$	170.60 (6.30)$^+$
4	**156.11 (2.54)**	115.84 (4.70)$^+$	150.90 (3.90)
5	**138.17 (2.51)**	111.59 (3.42)$^+$	136.60 (2.90)
6	124.95 (2.99)	109.12 (2.69)$^+$	**125.20 (2.10)**
7	114.51 (2.63)	106.17 (1.88)$^+$	**115.40 (1.60)**
8	106.77 (1.18)	103.66 (0.84)$^+$	**106.90 (1.20)**
9	100.00 (0.00)	100.00 (0.00)	100.00(0.00)

B. Cross-validation results

In order to compare the robustness of the prediction models, we adopt a 10-fold cross-validation approach for performance estimation. A data set is randomly partitioned into 10 mutually exclusive and exhaustive folds. Each time, a different fold is chosen as the test set and other nine folds are combined together as the training set. Prediction models are learned from the training set and evaluated on the corresponding test set.

In Table I, the average of the cumulative lifts of the models learned by different methods are summarized. Numbers in the parentheses are the standard deviations. The highest cumulative lift in each decile is highlighted in bold. The superscript $^+$ represents that the cumulative lift of the model obtained by the parallel MOEA is significant higher at 0.05 level than that of the models obtained by the corresponding methods. The superscript $^-$ represents that the cumulative lift of the model obtained by the parallel MOEA is significant lower at 0.05 level than that of the corresponding models.

From Table I, the models generated by the parallel MOEA have the average cumulative lifts of 358.47 and 270.46 in the first two deciles respectively, suggesting that by mailing to the top two deciles alone, the models generate over twice as many respondents as a random mailing without a model. Moreover, the average cumulative lifts of the models learnt by the parallel MOEA are significantly higher than those of the models obtained by the other methods for the first four deciles.

The average of the cumulative profit lifts of the models learned by different methods are summarized in Table II. It is observed that the average cumulative profit lifts of the models learnt by the parallel MOEA are significantly higher than those of the models obtained by the other methods for the first three deciles. The average profits for different models are listed in Table III. Direct marketers can get $11,461.63 if they use the parallel MOEA to generate models for selecting 20% of the customers from the dataset. On the other hand, they can get only $10,514.24 if they apply the DMAX approach for selecting customers. The parallel HGA cannot learn good models because the objective (i.e. Eq. 7) representing the constraint is not considered in this approach.

In order to study the effect of the value of r on the

TABLE II

CUMULATIVE PROFIT LIFTS OF THE MODELS LEARNED BY DIFFERENT METHODS.

Decile	Parallel MOEA	Parallel HGA	DMAX
0	**621.00 (49.16)**	242.35 (38.54)$^+$	550.80 (61.00)$^+$
1	**382.03 (14.14)**	188.26 (16.5)$^+$	350.80 (24.10)$^+$
2	**278.72 (10.26)**	166.24 (9.71)$^+$	261.70 (15.70)$^+$
3	**221.96 (8.05)**	151.66 (12.04)$^+$	213.30 (7.40)
4	**181.81 (4.90)**	139.39 (9.96)$^+$	179.10 (5.30)
5	155.32 (5.27)	129.93 (5.91)$^+$	**156.30 (4.20)**
6	135.55 (4.51)	121.79 (4.74)$^+$	**137.60 (3.90)**
7	121.20 (3.71)	114.63 (2.95)$^+$	**122.10 (3.00)**
8	**110.20 (1.57)**	108.2 (1.16)$^+$	109.70 (2.00)
9	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)

TABLE III

AVERAGE PROFITS FOR THE MODELS LEARNED BY DIFFERENT METHODS.

Decile	Parallel MOEA	Parallel HGA	DMAX
0	**$9,339.04**	$3,646.49	$8,254.34
1	**$11,461.63**	$5,650.60	$10,514.24
2	**$12,545.96**	$7,472.12	$11,765.58
3	**$13,317.80**	$9,077.67	$12,786.13
4	**$13,631.38**	$10,429.06	$13,420.04
5	$13,974.12	$11,677.79	**$14,053.96**
6	$14,234.71	$12,768.95	**$14,434.60**
7	$14,542.35	$13,744.59	**$14,638.41**
8	**$14,867.79**	$14,588.65	$14,795.77
9	$14,986.09	$14,986.09	$14,986.09

performance of the models learnt by the parallel MOEA, we apply different values of r and compare the cumulative lifts and the cumulative profit lifts of the induced models. From Tables IV and V, it is found that our approach is quite stable because it can learn good models for different values of r.

C. Comparison between GPU and CPU approaches

We compare the CPU and the GPU implementations of the MOEA. The average execution time of different steps of the CPU implementation is summarized in Table VI. The ratios of the time used in fitness evaluations to the overall execution time are also reported in this table. It can be observed that the fitness evaluation time is significantly higher than that of the other steps because the training sets are very large. The average execution time of the GPU implementation is summarized in Table VII. The parallel MOEA takes about 126 seconds to learn a model. On the other hand, it takes about 648 seconds and 7,315 sseconds respectively for the CPU implementation of the MOEA and the DMAX approach to learn a model.

Table VIII displays the speedups of the overall programs and different steps of the programs. The speedups of the GPU implementation of the dominance checking procedure ranges from 1.95 to 2.26. On the other hand, the non-dominated selection procedure of the GPU implementation is slower than that of the CPU approach. The overall speedup is about 5.1.

TABLE IV

CUMULATIVE LIFTS OF THE MODELS LEARNED BY THE PARALLEL MOEA.

Decile	$r = 10\%$	$r = 30\%$	$r = 40\%$	$r = 50\%$
0	363.28 (24.59)	354.39 (27.66)	363.28 (24.59)	363.28 (24.59)
1	267.32 (8.40)	267.02 (12.43)	267.32 (8.40)	267.32 (8.40)
2	214.89 (5.59)	220.53 (7.28)	214.89 (5.59)	214.89 (5.59)
3	180.27 (4.61)	185.51 (4.78)	180.27 (4.61)	180.27 (4.61)
4	155.63 (3.76)	161.84 (3.74)	155.63 (3.76)	155.63 (3.76)
5	137.96 (3.03)	143.03 (1.98)	137.96 (3.03)	137.96 (3.03)
6	124.69 (2.68)	128.90 (1.85)	124.69 (2.68)	124.69 (2.68)
7	114.10 (1.95)	117.33 (1.48)	114.10 (1.95)	114.10 (1.95)
8	106.16 (1.50)	108.72 (0.85)	106.16 (1.50)	106.16 (1.50)
9	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)

TABLE V

CUMULATIVE PROFIT LIFTS OF THE MODELS LEARNED BY THE PARALLEL MOEA.

Decile	$r = 10\%$	$r = 30\%$	$r = 40\%$	$r = 50\%$
0	621.08 (46.31)	616.99 (51.25)	621.08 (46.31)	621.08 (46.31)
1	377.88 (14.76)	377.51 (18.49)	377.88 (14.76)	377.88 (14.76)
2	276.31 (9.96)	281.15 (11.13)	276.31 (9.96)	276.31 (9.96)
3	217.91 (6.91)	222.98 (10.05)	217.91 (6.91)	217.91 (6.91)
4	181.69 (5.75)	185.63 (6.98)	181.69 (5.75)	181.69 (5.75)
5	154.24 (4.39)	158.46 (3.67)	154.24 (4.39)	154.24 (4.39)
6	134.84 (4.25)	139.18 (3.18)	134.84 (4.25)	134.84 (4.25)
7	119.91 (2.99)	123.25 (2.36)	119.91 (2.99)	119.91 (2.99)
8	108.40 (2.52)	111.31 (0.82)	108.40 (2.52)	108.40 (2.52)
9	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)	100.00 (0.00)

TABLE VI

THE AVERAGE EXECUTION TIME (IN SECONDS) OF THE CPU IMPLEMENTATION. THE OT, DC, NS, AND FE ROWS SHOW THE AVERAGE TIME IN PERFORMING RESPECTIVELY ALL STEPS, THE DOMINANCE CHECKING STEP, THE NON-DOMINATED SELECTION STEP, AND FITNESS EVALUATIONS.

	Generation								
	100	150	200	250	300	350	400	450	500
OT	129.29	194.15	259.10	324.09	389.11	454.15	519.20	584.29	648.11
DC	0.50	0.79	1.06	1.34	1.60	1.88	2.15	2.41	2.67
NS	0.01	0.02	0.02	0.03	0.03	0.05	0.06	0.06	0.07
FE	128.78	193.35	258.02	322.73	387.48	452.22	516.99	581.82	645.37
ratio	0.9960	0.9959	0.9958	0.9958	0.9958	0.9958	0.9958	1.00	1.00

TABLE VII

THE AVERAGE EXECUTION TIME (IN SECONDS) OF THE GPU IMPLEMENTATION.

	Generation								
	100	150	200	250	300	350	400	450	500
OT	25.27	37.76	50.32	62.95	75.61	88.32	101.05	113.81	126.37
DC	0.25	0.36	0.47	0.60	0.73	0.83	0.95	1.06	1.19
NS	0.02	0.04	0.04	0.05	0.07	0.08	0.09	0.10	0.11
FE	25.00	37.36	49.81	62.31	74.81	87.40	100.01	112.65	125.07
ratio	0.989	0.989	0.990	0.990	0.989	0.990	0.990	0.990	0.990

TABLE VIII

THE SPEEDUPS OF THE GPU IMPLEMENTATION WITH THE CPU IMPLEMENTATION.

	Generation								
	100	150	200	250	300	350	400	450	500
OT	5.116	5.142	5.149	5.148	5.146	5.142	5.138	5.134	5.129
DC	1.997	2.166	2.259	2.253	2.196	2.255	2.261	2.263	2.248
NS	0.547	0.436	0.485	0.487	0.439	0.576	0.622	0.615	0.620
FE	5.151	5.175	5.180	5.180	5.180	5.174	5.170	5.165	5.160

Since a marketing campaign often involves huge dataset and large investment, prediction models which can categorize more prospects into the target list are valuable as they will enhance the response rate as well as the return on investment. From the experimental results, the prediction models generated by the parallel MOEA are more effective than the other models and the parallel MOEA is significantly faster than the DMAX approach.

V. Conclusions

An important issue in targeted marketing is how to find potential customers who contribute large profit to a firm under constrained resources. In this research, we have proposed a data mining method to learn models for identifying valuable customers. We have formulated this learning problem as a constrained optimization problem of finding a scoring function f and a threshold value τ. We have then conversed it to an unconstrained Multi-objective Optimization Problem (MOP).

By limiting f to be a linear function, a parallel MOEA on GPU has been used to handle the MOP and find the parameters of f as well as the value of τ. We have used 10-fold cross-validation and decile analysis to compare the performance of the parallel MOEA, the parallel HGA, and the DMAX approach for a real-life direct marketing problem. Based on the cumulative lifts, cumulative profit lifts, and average profits, it can be concluded that the models generated by the parallel MOEA significantly outperform the models learnt by other methods in many deciles. Thus, the parallel MOEA is more effective. Moreover, it is significantly faster than the DMAX approach.

We have performed experiments to compare our parallel MOEA and a CPU implementation of MOEA. It is found that the overall speedup is about 5.1. Thus, our approach will be very useful for solving difficult direct marketing problems that involves large datasets and require huge population sizes.

For future work, we will extend our method to learn non-linear scoring functions and apply it to other targeted marketing problems under resource constrains.

Acknowledgment

This work is supported by the Lingnan University Direct Grant DR08B2.

References

[1] Thomas Philip Runarsson and Xin Yao, "Search Biases in Constrained Evolutionary Optimization," *IEEE Transactions on Systems, Man, and Cybernetics - Part C*, vol. 35, no. 2, pp. 233–243, May 2005.

[2] Yong Wang, Zixing Cai, Guanqi Guo, and Yuren Zhou, "Multi-objective Optimization and Hybrid Evolutionary Algorithm to Solve Constrained Optimization Problems," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, vol. 37, no. 3, pp. 560–575, June 2007.

[3] M. L. Wong, T. T. Wong, and K. L. Fok, "Parallel Hybrid Genetic Algorithms on Consumer-level Graphics Hardware," in *Proceedings of the 2006 Congress on Evolutionary Computation (CEC'2006)*, July 2006, pp. 10330–10337.

[4] Siddhartha Bhattacharyya, "Direct marketing performance modeling using genetic algorithms," *INFORMS J. on Computing*, vol. 11, no. 3, pp. 248–257, 1999.

[5] F. J. Mulhern, "Customer profitability analysis: Measurement, concentration, and research directions," *Journal of Interactive Marketing*, vol. 13, no. 1, pp. 25–40, 1999.

[6] J. R. Bult and T. Wansbeek, "Optimal selection for direct mail," *Management Science*, vol. 14, no. 4, pp. 1362–1381, 1995.

[7] Geng Cui and Man Leung Wong, "Implementing neural networks for decision support in direct marketing," *International Journal of Market Research*, vol. 46, no. 2, pp. 1–20, 2004.

[8] Lian Yan and Patrick Baldasare, "Beyond Classification and Ranking: Constrained Optimization of the ROI," in *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining*, 2006, pp. 948–953.

[9] Kalyanmoy Deb, *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, 2001.

[10] Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga, "Handling Multiple Objectives With Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, June 2004.

[11] Joshua D. Knowles and David W. Corne, "Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 100–116, April 2003.

[12] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg, "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, June 1994, vol. 1, pp. 82–87.

[13] Eckart Zitzler and Lothar Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, November 1999.

[14] Eckart Zitzler, Marco Laumanns, and Lothar Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," in *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, Eds., 2002, pp. 95–100.

[15] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[16] Joshua D. Knowles and David W. Corne, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149–172, 2000.

[17] David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates, "PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, 2001, pp. 283–290.

[18] Gary G. Yen and Haiming Lu, "Dynamic Multiobjective Evolutionary Algorithm: Adaptive Cell-Based Rank and Density Estimation," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 253–274, June 2003.

[19] nVidia, "NVIDIA CUDA^{tm} Programming Guide Version 2.3," http://developer.nvidia.com/object/cuda.html, 2009.

[20] K. L. Fok, T. T. Wong, and M. L. Wong, "Evolutionary Computing on Consumer-level Graphics Hardware," *IEEE Intelligent Systems*, vol. 22, no. 2, pp. 69–78, 2007.

[21] M. L. Wong, T. T. Wong, and K. L. Fok, "Parallel Evolutionary Algorithms on Graphics Processing Unit," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC'2005)*, September 2005, pp. 2286–2293.

[22] Simon Harding and Wolfgang Banzhaf, "Fast Genetic Programming on GPUs," in *Proceedings of the 10th European Conference on Genetic Programming (EuroGP'2007)*, April 2007, pp. 90–101.

[23] W. B. Langdon, "Evolving GeneChip Correlation Predictors on Parallel Graphics Hardware," in *Proceedings of the 2008 Congress on Evolutionary Computation (CEC'2008)*, June 2008, pp. 4152–4157.

[24] Denis Robilliard, Virginie Marion-Poty, and Cyril Fonlup, "Population Parallel GP on the G80 GPU," in *Proceedings of the 11th European Conference on Genetic Programming (EuroGP'2008)*, April 2008, pp. 98–109.

[25] W. B. Langdon and W. Banzhaf, "A SIMD Interpreter for Genetic Programming on GPU Graphics Cards," in *Proceedings of the 11th European Conference on Genetic Programming (EuroGP'2008)*, April 2008, pp. 73–85.

[26] Darren M. Chitty, "A Data Parallel Approach to Genetic Programming Using Programmable Graphics Hardware," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, July 2007, vol. 2, pp. 1566–1573.

[27] G. Wilson and W. Banzhaf, "Linear Genetic Programming GPGPU on Microsoft's Xbox 360," in *Proceedings of the 2008 Congress on Evolutionary Computation (CEC'2008)*, June 2008, pp. 378–385.
[28] Man Leung Wong, "Parallel Multi-Objective Evolutionary Algorithms on Graphics Processing Units," in *GECCO–2009: Proceedings of the Genetic and Evolutionary Computation Conference*, 2009, pp. 2515–2522.
[29] Olivia Parr Rud, *Data Mining Cookbook: modeling data for marketing, risk and customer relationship management*, Wiley, 2001.
[30] Moninder Singh, *Learning Bayesian Networks for Solving Real-World Problems*, Ph.D. thesis, University of Pennsylvania, 1998.
[31] J. Zahavi and N. Levin, "Applying neural computing to target marketing," *Journal of Direct Marketing*, vol. 11, no. 4, pp. 76–93, 1997.