Solving Credit Card Fraud Detection Problem by the New Metaheuristics Migrating Birds Optimization

Ekrem Duman[1] and Ilker Elikucuk[2]

[1] Özyeğin University, Faculty of Engineering,
Industrial Engineering Department, Istanbul, Turkey
ekrem.duman@ozyegin.edu.tr
[2] Intertech, Decision Support Systems Department, Istanbul Turkey
Ilker.elikucuk@intertech.com.tr

Abstract. Statistical fraud detection problem is a very difficult problem in that there are very few examples of fraud. The great majority of transactions are legitimate. On the other hand, for this binary classification problem the costs of the two types of classification errors (FP=false positive and FN=false negative) are not the same. Thus, the classical data mining algorithms do not fit to the problem exactly. Departing from this fact, we have solved this problem by genetic algorithms and scatter search. Now, we apply the recently developed new metaheuristics algorithm namely the migrating birds optimization algorithm (MBO) to this problem. Results show that it outperforms the former approach. The performance of standard MBO is further increased by the help of some modified benefit mechanisms.

Keywords: migrating birds optimization algorithm, fraud, credit cards, genetic algorithms.

1 Introduction

When a credit card is copied or stolen, the transactions made by them are labeled as fraudulent. These fraudulent transactions should be prevented or detected in a timely manner otherwise the resulting losses can be huge. Banks typically use two layers of fraud prevention/detection systems; rule based layer and statistical analysis layer. Here we are concerned with the statistical layer only where an incoming transaction is compared to card usage behavior of the card holder and if there is a considerable deviation, a high suspicion score is returned.

Statistical fraud detection is not an easy problem at all due to several reasons. First, fraud data sets are extremely skewed; out of 100.000 transactions only a few turn out to be fraud [1]. Secondly, the techniques used by fraudsters change in time gradually [2-4]. Thus, a model developed now may not be effective enough in future. In addition to these the idea exchanges between the banks are very limited because of the privacy issues; no one wants other banks know how many frauds they were faced with and no bank shares details of their solution if they think they have a good one. In this regard, our study differentiates from many in literature and although we will not

I. Rojas, G. Joya, and J. Cabestany (Eds.): IWANN 2013, Part II, LNCS 7903, pp. 62–71, 2013.

be able to share all details, we will be talking about a fraud detection solution developed using real data and implemented in real life.

Due to its importance it is possible to find a lot of studies on fraud detection in literature. The most commonly used fraud detection methods are rule-induction techniques, decision trees, neural networks, Support Vector Machines (SVM), logistic regression, and meta-heuristics such as genetic algorithms [5-12]. These techniques can be used alone or in collaboration using ensemble or meta-learning techniques to build classifiers. Quah and Sriganesh [13], suggest a framework which can be applied real time where first an outlier analysis is made separately for each customer using self organizing maps and then a predictive algorithm is utilized to classify the abnormal looking transactions. Panigrahi et al. [14] suggest a four component fraud detection solution which is connected in a serial manner. The main idea is first to determine a set of suspicious transactions and then run a Bayesian learning algorithm on this list to predict the frauds. Sanchez et al. [15] presented a different approach and used association rule mining to define the patterns for normal card usage and indicating the ones not fitting to these patterns as suspicious. The study of Bolton and Hand [2] provides a very good summary of literature on fraud detection problems.

In most of the studies listed above the classical accuracy based model performance measures are used. Among these the accuracy ratio, the capture rate, the hit rate, the gini index and the lift are the most popular ones [16-17]. However, since the fraudsters use all available limit on the card they captured, an algorithm which is more successful in detecting the cards with higher available limits is more prominent. In this case the cost of making a false positive error and a false negative error will not be the same and actually false negative error (labeling a fraudulent transaction as legitimate) will be a variable. We will take this cost function here similar to few studies in literature [18-19]. A rather more detailed review of these two studies will be given in the next section.

The contributions of this study to the literature are three fold. First, we are talking on models built with real data and implemented in real life. Second, the new metaheuristic MBO is used to solve a credit card detection problem for the first time and this will be one of the very few studies where any member of the metaheuristic algorithms family is used. Third, the performance of MBO is improved further through the use of modified benefit mechanisms.

The rest of the paper is organized as follows. In the next section, the fraud detection problem we were faced is described in detail together with the explanation of closely related previous work. In the third section we describe the MBO algorithm as it is used to solve the quadratic assignment problem in [1]. Implementation of MBO on the credit card fraud detection problem and the modifications made on it to improve its performance are detailed in section four. The paper is finalized in section five by giving a summary of the study and the major conclusions.

2 Problem Definition and Previous Work

There has been a growing amount of financial losses due to credit card frauds as the usage of the credit cards become more and more common. As such, many papers reported huge amounts of losses in different countries [2, 20].

Credit card frauds can be made in many ways such as simple theft, application fraud, counterfeit cards, never received issue (NRI) and online fraud (where the card holder is not present). In online fraud, the transaction is made remotely and only the card's details are needed. A manual signature, a PIN or a card imprint are not required at the time of purchase. Though prevention mechanisms like CHIP&PIN decrease the fraudulent activities through simple theft, counterfeit cards and NRI; online frauds (internet and mail order frauds) are still increasing in both amount and number of transactions. According to Visa reports about European countries, approximately 50% of the whole credit card fraud losses in 2008 are due to online frauds [21].

When the fraudsters obtain a card, they usually use (spend) all of its available (unused) limit. According to the statistics, they do this in four - five transactions, on the average [18]. Thus, for the fraud detection problem, although the typical prediction modeling performance measures are quite relevant, as indicated by the bank authorities, a performance criterion, measuring the loss that can be saved on the cards whose transactions are identified as fraud is more prominent. In other words, detecting a fraud on a card having a larger available limit is more valuable than detecting a fraud on a card having a smaller available limit.

As a result, what we are faced with is a classification problem with variable misclassification costs. Each false negative has a different misclassification cost and the performance of the model should be evaluated over the total amount of saved available usable limits instead of the total number of frauds detected.

If we define;

TP = the number of correctly classified alerts
TN = the number of correctly classified legitimates
FP = the number of false alerts
FN = the number of transactions classified as legitimate but are in fact fraudulent
c = the cost of monitoring an alert
TFL = the total amount of losses due to fraudulent transactions
S = savings in TFL with the use of fraud detection system
ρ = savings ratio

Then,

TFL = sum of the available limits of the cards whose transactions are labeled as TP or FN

c = cost of monitoring including staff wages, SMSs, phone calls. On the average, it is a small figure (less than a dollar)

S = (available limits of the cards of TP transactions) - c(FP+TP)

ρ = S/TFL

where the maximum value S can take is TFL and ρ can take is 1. A good predictor will be the one having a high ρ.

Duman and Ozcelik [18] tackled the same problem for another bank in Turkey. After putting the problem in the same way and pointing out the classical DM algorithms may not perform well for the objective of maximizing savings ratio they implemented a metaheuristic approach which is a hybrid of genetic algorithms and scatter search (the GASS algorithm). In GASS, the number of parent solutions was taken as 50 and

child solutions were generated by the recombination operator. Each parent is recombined by every other parent so that the number of children was 1225 in each generation. One of the child solutions is selected randomly and one of its genes is mutated. As the fitness function the savings ratio is used. In the selection process, besides the fittest members which are determined by roulette selection, the most diverse solutions are also inherited to the next generation in [18]. GASS improved the savings ratio by more than 200% with a cost of 35% increase in the number of alerts. However, it had some limitations in that the authors were allowed to improve the score generated by some groups of variables only and the variables about MCCs (merchant categegory codes) and out of country expenditures were left out of the scope of the study. The problem was that a second score was being generated with these left out variables and the authors had no control on how these two scores were interfering.

The study of Duman and Sahin [19] gives a summary of previous results obtained in another bank where this study is carried out also. The results obtained by classical decision trees (C5.0, CART, CHAID), artificial neural networks, SVM, logistic regression and GASS are compared. The results obtained by a special cost sensitive decision tree where in splitting a node the savings ratios of the resulting leaf nodes are considered, is also given. It was shown that the GASS and the newly proposed cost sensitive decision trees were the two best performing methods.

The studies [22-24] are the other studies that tackle cost sensitive decision trees in literature.

3 The MBO Algorithm

The MBO algorithm is a neighborhood search technique [1]. It starts with a number of initial solutions corresponding to birds in a V formation. Starting with the first solution (corresponding to the leader bird), and progressing on the lines towards the tails, each solution is tried to be improved by its neighbor solutions (for the implementation of QAP (quadratic assignment problem), a neighbor solution is obtained by pairwise exchange of any two locations). If the best neighbor solution brings an improvement, the current solution is replaced by that one. There is also a benefit mechanism for the solutions (birds) from the solutions in front of them. Here we define this benefit mechanism as sharing the best unused neighbors with the solutions that follow (here "unused" means a neighbor solution which is not used to replace the existing solution). In other words, a solution evaluates a number of its own neighbors and a number of best neighbors of the previous solution and considered to be replaced by the best of them. Once all solutions are improved (or tried to be improved) by neighbor solutions, this procedure is repeated a number of times (tours) after which the first solution becomes the last, and one of the second solutions becomes first and another loop starts. The algorithm is stopped after a specified number of iterations.

Below, first the notation used and then the pseudo-code of the MBO algorithm are given. Let,

n = the number of initial solutions (birds)
k = the number of neighbor solutions to be considered

x = the number of neighbor solutions to be shared with the next solution
m = number of tours
K = iteration limit

Pseudocode of MBO:

1. Generate n initial solutions in a random manner and place them on an hypothetical V formation arbitrarily.
2. i=0
3. while(i<K)
4. for (j=0;j<m;j++)
5. Try to improve the leading solution by generating and evaluating k neighbors of it.
6. i=i+k
7. for each solution s_r in the flock (except leader)
8. Try to improve s_r by evaluating (k-x) neighbors of it and x unused best neighbors from the solution in the front.
9. i=i+(k-x)
10. endfor
11. endfor
12. Move the leader solution to the end and forward one of the solutions following it to the leader position.
13. endwhile
14. return the best solution in the flock

As should already be noticed, the MBO algorithm has great similarities with the migrating birds' story. First it treats the solutions as birds aligned on a V formation. The number of neighbors generated (k) can be interpreted as the induced power required which is inversely proportional to the speed (recall the discussion above). With a larger k we would assume that birds are flying at a low speed where we can also make the analogy that while traveling at a low speed, one can explore the surrounding in more detail. The benefit mechanism between the birds is respected and by generating fewer neighbors for the solutions at the back, it was made possible that they get tired less and save energy by using the neighbors of the solutions in the front. The parameter x is seen as the WTS where an optimum value can be sought for. Its optimum value could be interpreted as the optimum overlap amount of the wingtips. In line 4, the parameter m can be regarded as the number of wing flaps or the profile power needed where we can assume that, as each bird travels the same distance, they all spend the same profiling energy. In line 12, similar to the real birds' story, the bird who spent the most energy and thus got tired moves back to get some rest and another bird fills its position.

For the MBO to perform better, it is necessary to determine the best values of some parameters. These are the number of birds to have in the flock (n), the speed of the flight (k), the WTS (x) and the number of wing flaps before a change in the order of the birds or the profiling energy spent (m). Similar to birds' story, one could expect some certain values of these parameters and their combinations might increase the

performance of the algorithm. For the printed circuit board assembly originated quadratic assignment problem the best parameter values were obtained to be n = 51, m = 10, k = 11 and x = 1 in [1].

The philosophy of the MBO is that, by starting with a number of solutions, it is aimed to explore more areas of the feasible solution space. The exploration is made possible by looking at the neighbor solutions. Each time one of the solutions (the one in the front) is explored in more detail. When one of the solutions fails to improve itself by its own neighbors and if the solution in the front is more promising, it is replaced by one of the neighbors of the solution in the front. This way the neighborhood around the more promising solution will be explored in a greater detail (by the combined forces of two birds or solutions). Still after a few iterations these solutions may go to different directions as long as they find improvements along their ways. However, after some time we might expect most of the solutions converge to one or several neighborhoods where local optima or even the global optimum are contained. The convergence can be faster with larger values of x but in that case the termination could take place before the feasible region is thoroughly explored and thus the results obtained might not be good.

4 Results and Discussion

In the following subsections first the details of the experimental setting are given. Then, the results obtained by standard MBO and GASS are displayed together with their parameter fine tuning experiments. This is followed by some improvement attempts on MBO by employing different neighborhood functions.

4.1 Details of Experimental Setting

The original data of the time period used to form the training set have about 22 million transactions. The distribution of this data with respect to being normal or fraudulent is highly skewed so that only 978 transactions were fraudulent in this set. So, to enable the models to learn both types of profiles, some under sampling or oversampling techniques should be used. Instead of oversampling the fraudulent records by making multiple copies of them, we use stratified sampling to under sample the legitimate records to a meaningful number. Firstly, we identify the variables which show the most different distributions w.r.t. being fraudulent or normal. Then, we use these variables as the key variables in stratified sampling so that the characteristics of their distributions w.r.t. being fraudulent or not remains the same. For stratified sampling, we use those five variables which show the most different distributions to form a stratified sample with a ratio of nine legitimate transactions to one fraudulent transaction.

4.2 Results Obtained by MBO

The total number of variables in the data mart was 139 (all binary). Starting with the full set of variables, a variable reduction is made first. Each time the variables having

coefficients close to zero (in the trained model) are eliminated and a new model is generated. This resulted in 15 variables where the coefficients of the variables were significantly different than zero. During these runs the MBO parameter values are used same as the best set obtained in [1].

Then a set of parameter fine tuning experiments are made on MBO. According to this analysis the best set of parameters is determined to be:

Table 1. MBO versus GASS

	MBO			**GASS**	
RUN	ρ	TPR		ρ	TPR
1	94,74	91,94		90,92	79,34
2	95,13	91,94		85,64	77,69
3	94,21	91,53		90,89	78,1
4	94,74	91,94		90,63	77,89
5	94,26	91,74		90,94	79,34
6	91,3	77,89		94,78	91,32
7	94,91	91,74		91,22	79,75
8	94,91	91,74		93,78	89,26
9	94,71	91,94		94,26	91,74
10	91,04	78,1		91,41	79,96
AVG	**93,98**	**88,91**		**91,45**	**82,78**

Number of Birds: 15
Number of Neighbors : 7
Number of Flaps: 3
Number of Overlaps: 2

The parameter values of the GASS algorithm implemented here are determined in accordance with the study [18], namely:

Number of variables (genes): 15

Number of initial solutions: 15

Maximum number of tested solutions: 15000

Mutation procedure: One of the 15 genes is selected randomly and a uniform number between 0 and 1 (U(0,1)) is added to its value. If the result is greater than the upper bound (1.0), the final value is obtained by subtracting the upper bound from the new value.

Mutation rate: 8% of the intermediate solutions are selected for mutation.

Selection procedure: Roulette wheel

The results obtained by GASS and MBO are compared in Table 1 where TPR stands for true positive rate.

The results in Table 2 indicate that the MBO algorithm performs significantly better than the GASS algorithm (about three per cent higher limit saving and about five per cent higher TPR is obtained). This superiority is also validated by an independent t-test where the p-values turn out to be 2.3 per cent and 2.6 per cent for the limit savings and TPR, respectively.

Table 2. MBO when two genes are shared

	Standard MBO			**Gene Sharing**	
RUN	ρ	TPR		ρ	TPR
1	94,74	91,94		94,71	91,94
2	95,13	91,94		95,13	91,94
3	94,21	91,53		94,96	91,94
4	94,74	91,94		95,13	91,94
5	94,26	91,74		94,74	91,94
6	91,3	77,89		94,27	91,74
7	94,91	91,74		94,26	91,74
8	94,91	91,74		90,33	85,74
9	94,71	91,94		94,74	91,94
10	91,04	78,1		95,19	92,15
AVG	**93,98**	**88,91**		**94,35**	**91,3**

4.3 Modifications on MBO

After finding out that the MBO algorithm is a good solver in fraud detection, we continued with testing a different mechanism in MBO. For this purpose instead of sharing some neighbor solutions with the followers, inspired by the genetic algorithms, we wanted to see the effects of sharing some genes (coefficient values). A neighbor solution of a current solution is generated by getting a number of genes (say, *y*) from the front solution (the selection of which *y* genes is determined randomly) and the rest of the genes are copied from the current solution where one of them is mutated like in the standard MBO. For the value of *y* (the number of genes shared), values from 1 to 5 are tested. We saw that two genes sharing happened to bring the most improvement. To test the significancy of this improvement, we tabulated the results of 10 runs in Table 2 and applied t-test. According to t-test, the differences on the performances of the two are not significant with p-values of 29.2% and 11.5% for ρ and TPR, respectively.

To conclude, we can say that when the standard MBO of Duman et al. [1] is coupled with a good benefit mechanism with gene sharing between solutions, its performance can slightly be increased.

5 Summary and Conclusions

In this study we solved the credit card fraud detection problem by the new metaheuristics migrating birds optimization algorithm. This study is one of the few studies where a metaheuristic algorithm is utilized in fraud detection and it performed better than a previous implementation of a combination of genetic algorithms and scatter search. The standard MBO is further improved by testing alternative benefit mechanisms.

As future work, alternative settings of V formation (like more than two tails etc) can be tested.

References

1. Duman, E., Uysal, M., Alkaya, A.F.: Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. Information Sciences 217, 65–77 (2012)
2. Bolton, R.J., Hand, D.J.: Statistical fraud detection: A review. Statistical Science 28(3), 235–255 (2002)
3. Kou, Y., et al.: Survey of fraud detection techniques. In: Proceedings of the 2004 IEEE International Conference on Networking, Sensing and Control, Taipei, Taiwan, March 21-23 (2004)
4. Phua, C., et al.: A comprehensive survey of data mining-based fraud detection research. Artificial Intelligence Review (2005)
5. Sahin, Y., Duman, E.: An overview of business domains where fraud can take place, and a survey of various fraud detection techniques
6. Brause, R., Langsdorf, T., Hepp, M.: Neural data mining for credit card fraud detection. In: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence (1999)
7. Hanagandi, V., Dhar, A., Buescher, K.: Density-Based Clustering and Radial Basis Function Modeling to Generate Credit Card Fraud Scores. In: Proceedings of the IEEE/IAFE, Conference (1996)
8. Juszczak, P., Adams, N.M., Hand, D.J., Whitrow, C., Weston, D.J.: Off-the-peg and bespoke classifiers for fraud detection. Computational Statistics & Data Analysis 52(9) (2008)
9. Quah, J.T., Sriganesh, M.: Real-time credit card fraud detection using computational intelligence. Expert Systems with Applications 35(4) (2008)
10. Shen, A., Tong, R., Deng, Y.: Application of classification models on credit card fraud detection. In: International Conference on Service Systems and Service Management, Chengdu, China (June 2007)
11. Wheeler, R., Aitken, S.: Multiple algorithms for fraud detection. Knowledge-Based Systems 13(2/3) (2000)
12. Chen, R.-C., Chiu, M.-L., Huang, Y.-L., Chen, L.-T.: Detecting credit card fraud by using questionnaire-responded transaction model based on support vector machines. In: Yang, Z.R., Yin, H., Everson, R.M. (eds.) IDEAL 2004. LNCS, vol. 3177, pp. 800–806. Springer, Heidelberg (2004)
13. Han, J., Camber, M.: Data mining concepts and techniques. Morgan Kaufman, San Diego (2000)

14. Quah, J.T.S., Srinagesh, M.: Real-time credit fraud detection using computational intelligence. Expert Systems with Applications 35, 1721–1732 (2008)
15. Panigrahi, S., Kundu, A., Sural, S., Majumdar, A.: Credit Card Fraud Detection: A Fusion Approach Using Dempster-Shafer Theory and Bayesian Learning. Information Fusion, 354–363 (2009)
16. Sanchez, D., Vila, M.A., Cerda, L., Serrano, J.M.: Association rules applied to credit card fraud detection. Expert Systems with Applications 36, 3630–3640 (2009)
17. Kim, M., Han, I.: The Discovery of Experts' Decision Rules from Qualitative Bankrupcy Data Using Genetic Algorithms. Expert Systems with Applications 25, 637–646 (2003)
18. Gadi, M.F.A., Wang, X., do Lago, A.P.: Credit Card Fraud Detection with Artificial Immune System. In: Bentley, P.J., Lee, D., Jung, S. (eds.) ICARIS 2008. LNCS, vol. 5132, pp. 119–131. Springer, Heidelberg (2008)
19. Duman, E., Ozcelik, M.H.: Detecting credit card fraud by genetic algorithm and scatter search. Expert Systems with Applications 38, 13057–13063 (2011)
20. Duman, E., Sahin, Y.: A Comparison of Classification Models on Credit Card Fraud Detection with respect to Cost-Based Performance Metrics. In: Duman, E., Atiya, A. (eds.) Use of Risk Analysis in Computer-Aided Persuasion. NATO Science for Peace and Security Series E: Human and Societal Dynamics, vol. 88, pp. 88–99. IOS Press (2011)
21. Gartner Reports, from the World Wide Web (May 10, 2010), http://www.gartner.com
22. Mena, J.: Investigate Data Mining for Security and Criminal Detection. Butterworth-Heinemann, Amsterdam (2003)
23. Ling, C.X., Sheng, V.S.: Test Strategies for Cost-Sensitive Decision Trees. IEEE Transactions on Knowledge and Data Engineering 18(8) (2006)
24. Liu, X.: A Benefit-Cost Based Method for Cost-Sensitive Decision Trees. In: 2009 WRI Global Congress on Intelligent Systems, pp. 463–467 (2009)