Combining Artificial Bee Colony and Genetic Algorithms to Enhance the GPGPU-based ANN Classifier for Identifying Students with Learning Disabilities

Tung-Kuang Wu, Shian-Chang Huang,
Chin-Yu,Hsu, Chih-Han Tai
Dept. of Information Management
National ChangHua University of Education
Changhua City, Taiwan
e-mail: tkwu@im.ncue.edu.tw,
shhuang@cc.ncue.edu.tw, sw9856@gmail.com,
p198711@gmail.com

Ying-Ru Meng
Dept. of Special Education
National HsinChu University of Education
HsinChu City, Taiwan
e-mail: myr321@mail.nhcue.edu.tw

***Abstract*—Diagnosis of students with learning disabilities (LD) is a difficult procedure that requires extensive man power and takes a long time. Fortunately, through genetic-based (GA) parameters optimization, artificial neural network (ANN) classifier may be a good alternative to the above procedure. However, GA-based ANN model construction is computation-intensive and may take quite a while to process. In this study, we examine another optimization algorithm, the artificial bee colony (ABC) algorithm, which is based on the foraging behavior of honey bee swarm, to search for the appropriate parameters in constructing ANN-based LD classifier. We also integrate ABC algorithm with GA evolution strategy by first applying the former to derive a set of values of the ANN parameters and then use these values as the starting points for the latter GA evolution procedure. In addition, to speed-up the above process, a low-cost general purpose graphics processing unit (GPGPU), specifically, the nVidia graphics card, is adopted for the ANN model training and validation. The experimental results show that ABC can achieve better correct identification rate (CIR) than GA with less computation time. In addition, the strategy of using ABC as a pre-processing step for GA evolution has improved the correct identification rate by as much as 2.5% in two of our three data sets when compared to using GA alone.**

Keywords-learning disabilities; neural network; CUDA; ABC

I. INTRODUCTION

The term "learning disabilities" (LD) was first used in 1963 [1]. However, experts in this field have not yet completely reach an agreement on the definition of LDs and its exact meaning [2]. In fact, a person can be of average or above average intelligence, without having any major sensory problems (like visual or hearing impairment), and yet struggles to keep up with people of the same age in learning and regular functioning. Due to such implicit characteristics of learning disabilities, the identification of students with LDs has long been a difficult and time-consuming process. In the United States, the so called "Discrepancy Model" [3], which states that a severe discrepancy between intellectual ability and academic achievement has to exist in one or more of these academic areas: (1) oral expression, (2) listening comprehension (3) written expression (4) basic reading skills (5) reading comprehension (6) mathematics calculation, used to be one of the commonly adopted criteria to evaluate whether a student is eligible for special education services. However, the newer Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition (DSM 5) by the American Psychiatric Association (APA) [4] has eliminated this requirement and replace it with four other criteria.

In Taiwan, the diagnosis procedure pretty much follows the "Discrepancy Model" despite the shift of criteria made by the APA. The sources of input parameters required in such prolonged process include information from parents, general education teachers, students' academic performance and a number of standard achievement and IQ tests. To guarantee collection of required information regarding students suspected with LD, usually checklists of various aspects are developed to assist parents and regular education teachers. The Learning Characteristics Checklists (LCC), a Taiwan locally developed LD screening checklist [5], is commonly used in most counties of Taiwan. Among the standard tests, the Wechsler Intelligence Scale for Children, Third or Fourth Edition (WISC III or IV) plays the most important role in this LD diagnosis model. WISC-III consists of 13 sub tests [6]. The scores of the sub-tests are then used to derive 3 IQs, which include Full scale IQ (FIQ), Verbal IQ (VIQ), Performance IQ (PIQ), and 4 indexes, which include Verbal Comprehension Index (VCI), Perceptual Organization Index (POI), Freedom from Distractibility Index (FDI), Processing Speed Index (PSI). There are also a number of locally developed standard achievement tests (AT), which typical consist of reading, math, and fields that are related to students' academic achievement.

Diagnosis of students with LDs then involves mainly interpreting the standard test scores and comparing them to the norms that are derived from statistical method. As an example, in case the difference between VIQ and PIQ is greater than 15, representing significant discrepancy between a student's cultural knowledge, verbal ability, and his/her ability in recognizing familiar items, interpreting action as depicted by pictures, is a strong indicator in differentiating between students with or without LD [6]. A number of

 ISBN: 978-1-61208-416-9

similar indicators together with the students' academic records and descriptive data (if there is any) are then used as the basis for the final decision. Confirmed possible LD students are then evaluated for one year before admitting to special education. However, it is important to note that a previous study reveals that the certainty in predicting whether a student is having a LD using each one of the currently available predictors is in fact less than 50% [7].

The above identification procedure involves extensive manpower and resources. Furthermore, a lack of nationally regulated standard for the LD diagnosis procedure and criteria result in possible variations on the outcomes of diagnosis. In some cases, the difference can be quite significant [8].

With the advance in artificial intelligence (AI) and its successful applications to various classification problems, it is interesting to investigate how these AI-based techniques perform in identifying students with LDs. In our previous study, we have shown that ANN classifier does well in positively identifying students with LDs [8]. In subsequent studies, we combined various feature selection techniques and genetic-based parameters optimization with the ANN classifier, which further improve the overall identification accuracy [9]. However, despite the ANN-based classifier performs well in LD diagnosis problem, the procedure is computation-intensive and may take quite a while to process. Accordingly, multi-threaded programming, grid-based and cloud-based parallel computing have been used to speedup the ANN model training and validation [10][11][12].

In this paper, we are still focusing on the ANN classification model and work on porting the computation intensive ANN classifier to the general purpose graphics processing units (GPGPU). In addition, the ABC algorithm is evaluated and compared to the GA approach in terms of performance in optimizing the above mentioned ANN classifier.

The rest of the paper is organized as follows. Section 2 briefly describes the history of applying AI techniques to special education and gives a short introduction to ABC algorithm that is used in our implementation. Sections 3 and 4 present our experiment settings, design and corresponding results. Finally, Section 5 gives a brief conclusion of the paper and lists issues that deserve further investigation.

II. RELATED WORK

Artificial intelligence techniques have long been applied to special education. However, most of the studies occurred more than one or two decades ago and mainly focused on using the expert systems to assist special education in various ways [8]. There were also numerous classification techniques other than neural networks that were developed and widely used in various applications [13]. Among all the classification techniques, ANN has received lots of attentions due to its demonstrated performance and has gained wide acceptance [14].

An ANN is a mathematical representation that is inspired by the way the brain processes information. Many types of ANN models have been suggested in literature, with the most popular one for classification being the multilayer perceptron (MLP) with back propagation. The goal of this type of network is to create a model that correctly maps the input to the output using historical data so that the model can then be used to predict the outcome when the desired output is unknown. MLP with back propagation is typically composed of an input layer, one or more hidden layers and an output layer, each consisting of several neurons. Each neuron processes its inputs and generates one output value that is transmitted to the neurons in the subsequent layer. Figure 1 provides an example of an MLP with one hidden layer and one output neuron.

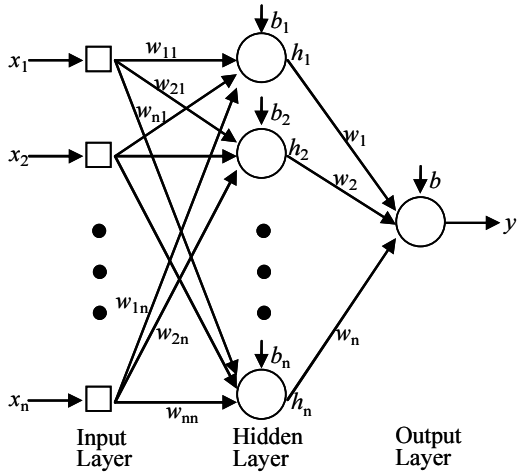

Figure 1. MLP with one hidden layer.

The output of *i*-th hidden neuron is computed by processing the weighted inputs and its bias term b_i as follows:

$$h_i = f^h\left(b_i + \sum_{j=1}^{n} w_{ij} x_j\right) \quad (1)$$

where w_{ij} denotes the weight connecting input x_j to hidden unit h_i. Similarly, the result of the output layer is computed as follows:

$$y = f^{output}\left(b + \sum_{j=1}^{n} w_j x_j\right) \quad (2)$$

with n being the number of hidden neurons and w_j represents the weight connecting hidden unit j to the output neuron. A threshold function is then applied to map the network output y to a classification label. The transfer functions f^h and f^{output} allow the network to model non-linear relationships in the data. Also note that the number of hidden layer nodes does not need to be the same as the number of input nodes.

The training of a neural network is the process of presenting the network with sample data and modifying the weights to approximate the desired function. In particular, an epoch indicates one iteration through the process of providing the network with a sample input and updating the network's weights. Let N_i, N_h and N_o respectively represent input feature size, number of hidden and output nodes, the total order of complexity is then $O(N_i \times N_h \times N_o + N_h \times N_o)$ for one

epoch [14]. Since a typical ANN training process usually takes 500 epochs, the computation complexity for training of an ANN model is roughly equal to $500\times N\times O(N_i\times N_h\times N_o+N_h\times N_o)$, where N represents the size of input samples for training.

In the field of special education, ANN has been used in a number of applications [8]. To improve the ANN classification accuracy, genetic-based algorithms have been used in the training and construction of ANN model [15]. A number of other approaches, such as particles swarm (PSO), ant colony (ACO) and asynchronous parallel pattern search (APPS) have also been proposed and applied to various optimization problems. In addition to the above optimization methods, a newer ABC algorithm simulating the foraging behavior of honey bees has also been proposed and shown to be performing better than GA and the other two optimization methods [16]. In this model, the colony of artificial bees consists of three different types of bees, which are employed bees, onlookers and scouts. The first half of the colony consists of employed bees and the second half includes onlookers. Each employed bee is in charge of one food source. In other words, the number of employed or on-looker bees is equal to the number of food sources. The employed bee may become a scout when its food source has been exhausted. Without regard to all the details, the procedure of ABC can be simply presented in Table I.

TABLE I. ABC PROCEDURE.

1. Generate initial n food sources S_i, i=1, 2 ⋯, n.
2. Each employed bee i, i =1, 2 ⋯, n, computes and memories the fitness of each food source S_i.
3. $cycle$ = 1
4. repeat
   4.1. For each on-looker bee i, i =, 2 ⋯, n, selects a new food source with roulette wheel method (with probability calculated using fitness values, $P_i=\frac{CIR(i)}{\sum_{k=1}^{n}CIR(k)}$) and generates its new position (v_{ij}) according to the following equation: $v_{ij}=x_{ij}+\phi_{ij}*(x_{ij}-x_{kj})$, i, $k\in\{1, 2,\ldots, n\}$, $j\in\{1, 2,\ldots, D\}$, x_{ij} is the current position of food source evaluated by employed bee i for parameter j to be optimized, k is a randomly selected number that is not equal to i, ϕ_{ij} is also a random number in the range [-1,1], and D is the number of parameters to be optimized.
   4.2. For each employed bees i, computes the new fitness value. If the fitness value of the food source is not improved for a continuous *limit* tries, abandons the food source. The employed bee, i, becomes a scout and randomly generates a new food source.
   4.3. Memorizes the best food source.
   4.4. $cycle = cycle +1$
5. Until $cycle = MCN$

According to Table I, there are three major parameters with ABC procedure: (1) maximum cycle number (*MCN*), which is similar to the number of generations in GA, (2) the number of *food source* (also, the number of employed as well as on-looker bees), which resembles the population size in GA, (3) the maximal continuous exploitation attempts to a food source without improvement, *limit*, before an employed bee becomes a scout. This parameter may prevent ABC from trapping into some local maximum, which is somewhat similar to mutation mechanism in GA.

However, optimization procedures such as those mentioned above (like GA or ABC) usually require numerous applications of the ANN training and validation processes (depending on the number of chromosomes / food sources and evolution generations / number of cycles), and thus usually takes quite a long time to process. Accordingly, researches have been applying parallel processing, which may provide affordable computational power, to speedup the time-consuming process. For network connected cluster or grid environment, message passing interface (MPI) is usually used to coordinate computing nodes for completing a common task. On the other hand, to take full advantage of the currently available multi-core processor technology, OpenMP may be used explicitly to direct multi-threaded, shared memory parallelism. In addition to nodes or CPU-level parallelism, the fast advancing GPU technology now finds its way to all kinds of applications that require computation power [17]. Although originally designed for 3D graphics application, the Compute Unified Device Architecture (CUDA) by nVIDIA has made the low-cost and general purpose use of GPU possible [17].

In this study, we will work on porting the ABC algorithm to the enhancement of the ANN classifier for LD identification, and use GPU to speed-up the ANN training and validation processes.

III. HARDWARE PLATFORM & IMPLEMENTATION ISSUES

The data sets used in this study are summarized in Table II, which together with the corresponding pre-processing (such as normalization and feature selection) are exactly the same as those used in [10][11][12].

TABLE II. DATA SETS AND THEIR FEATURE SIZE USED IN THIS STUDY

	sample size	number of features
data set 1	652	7
data set 2	125	7
data set 3	159	10

A workstation running Ubuntu 12.04 with hardware specifications listed in Table III is set up for the above objectives. The communication between CPU and GPU is accomplished through a PCI express bus with theoretical maximum 16GB/sec bandwidth (Although the GPU supports higher PCIe 3.0 standard, the main board we use supports only up to PCIe 2.0).

TABLE III. HARDWARE DETAILS USED IN THIS STUDY

	Processing Unit	No. of cores	Memory
PC	Intel Xeon Processor E3-1230 v2 @ 3.30GHz	4 physical cores	16 GB
GPU	nVIDIA GeForce GTX 660 @ 1.03GHz	960 physical cores (5 stream multi-processors, each with 192 stream processors)	2 GB

The implementation of the ANN classifier (for LD identification) is divided into two parts: (1) the PC host is responsible for the optimization procedures (either GA or ABC), and (2) the GPU takes care of the most time-

consuming neural network model training and validation procedure, as shown in Figure 2. Switching in between the PC host and GPU, there are two I/O operations (data transfer through PCI express slot) in each iteration of GA or ABC in Figure 2. For GA, each generation may contain only two I/Os, while for ABC the number may reach six within one cycle in the worst cases. This is due to the fact that in each cycle of ABC, there may consist of fitness function computations as a result of employed, on-looker and scout bees. Considering the bandwidth between CPU and GPU, the I/O operation may potentially become the bottleneck of overall computation. However, the advantage of such a design is that it is easy to incorporate different kind of optimization algorithms in the future.

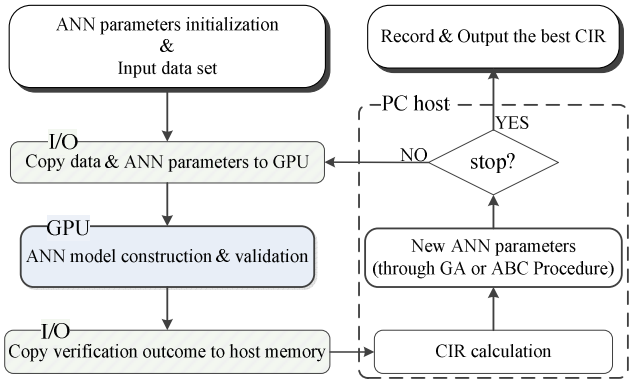

Figure 2. Computation workload distribution between the PC host and GPU.

Three parameters of an ANN classifier with one hidden layer, include number of hidden nodes (ranges in between 1 and 26), learning rate and momentum (both range in between 0.0 and 1.0), together with random number seed (which might affect the initial weights and bias of neural network) are encoded into the food sources (for ABC) or chromosomes (for GA). The training and validation of ANN model are sub-divided into six (for training) and four (for validation) independent procedures as shown in Table IV. Each of the ten procedures contains various numbers of computations that may potentially be parallelized by transforming them to the so called kernel functions.

TABLE IV. MAPPING OF ANN PROCEDURE TO GPGPU.

1. ANN Training Stage
For (i=1 → EPOCH) {
Initialize ANN input vector
Compute results of Input layer neurons
Compute results of Hidden layer neurons
Compute result of Output layer neuron & its error
Adjust Output layer neurons' weights
Adjust Hidden layer neurons' weights }
2. ANN Validation Stage
Initialize ANN input vector
Compute results of Input layer neurons
Compute results of Hidden layer neurons
Compute result of Output layer

As an example, for a GA-optimized (with size of population equals P) N_f-fold cross-validation ANN experiment with N_i input nodes and N_h hidden nodes, the total number of multiplications added up to $P \times N_f \times N_i \times N_h$ for the second procedure (Compute results of Input layer neurons) in the training stage of Table IV.

```
1. Determine threadID
2. Determine input vector & hidden layer weight with threadID
3. Compute the product of input vector & hidden layer weight
```

Figure 3. Pseudo kernel function code for computing results of Input layer neurons in the ANN training stage.

Each of the multiplications of the above example corresponds to the so called thread in CUDA's term and implemented as a kernel function as shown in Figure 3. These threads are then distributed to the 960 available cores of the GTX 660.

IV. EXPERIMENT DESIGNS AND RESULTS

The long-term goal of our series studies has been constructing an accurate ANN classifier for the identification of students with learning disabilities. The purposes of this study are to achieve the goal by evaluating ABC algorithm and exploring ways to further improve its performance. In addition, we would also like to speedup the above process with the adoption of GPGPU. The outcomes of this study will be compared to results by using the other optimization methods (such as GA).

We have designed and conducted four experiments for the purposes addressed above. In all the four experiments, CIR of ANN five-fold cross validation is used to evaluate the fitness and final outcomes of optimization (GA or ABC).

In the first experiment, we compare the performance of ABC and GA algorithms in optimizing our LD ANN classifier without the use of GPU. *MCN* and *Limit* (as explained in Table I) are set to 50 (the same as GA's number of generation) and 20 (will be examined in more details in experiment 3), while varying the number of food source. For genetic algorithm, real-value encoding is adopted with the crossover rate, mutation rate, population size and number of generation set to 0.8, 0.1, 100 and 50, respectively. Note, both ABC and GA codes are not parallelized (i.e., with OpenMP), which mean they utilize only one of the four available CPU cores. The results are shown in Table V.

TABLE V. PERFORMANCE COMPARISON BETWEEN SEQUENTIAL GA AND ABC IMPLEMENTATIONS. (ALL NUMBERS ARE AVERAGES OF 10 CONSECTIVE EXECUTIONS OF CODE AND ALL TIME IN SECONDS)

data set / method	1		2		3	
	CIR	*execution time*	*CIR*	*execution time*	*CIR*	*execution time*
GA	87.5%	9296	84.9%	3317	86.9%	6363
ABC(10*)	87.2%	1307	85.3%	713	86.5%	1026
ABC(20*)	87.6%	3390	85.6%	1261	87.0%	2126
ABC(30*)	87.6%	5399	85.8%	1751	87.2%	3188

* number of food sources.

As we can see, ABC can achieve approximately the same (or slightly better) CIR as GA algorithm (using 100 chromosomes) does with number of food sources equals 20, while take only about one third of time.

The second experiment is pretty much similar to the first one except the ANN code is now ported and executed in GPU with the number of food sources varying from 20 to 100. The results are shown in Table VI.

TABLE VI. PERFORMANCE COMPARISON BETWEEN GPU-ASSISTED GA AND ABC IMPLEMENTATIONS. (ALL NUMBERS ARE AVERAGES OF 10 CONSECTIVE EXECUTIONS OF CODE AND ALL TIME IN SECONDS)

data set / method	1		2		3	
	CIR	*execution time*	*CIR*	*execution time*	*CIR*	*execution time*
GA	87.9%	1304	85.2%	368	87.3%	641
ABC(20*)	87.9%	1342	85.2%	335	88.5%	480
ABC(40*)	88.1%	1590	86.2%	451	89.0%	722
ABC(60*)	88.1%	1889	86.3%	537	89.0%	894
ABC(80*)	88.2%	2339	86.7%	645	88.9%	1100
ABC(100*)	88.2%	2598	87.0%	729	89.1%	1267

* number of food sources.

The first thing to note in Table VI is that the GPU version ABC no longer has the edge in execution time as the sequential one. The major reason would be the increased number of I/Os (in each cycle of ABC) as we mentioned in Section III. We also notice that more food sources correspond to better CIR, which is not much surprise. However, the execution time does not seem to increase proportionally with the number of food sources. Apparently, more food sources mean the time spent in GPU computation is longer, which reduces the percentage of time wasted in I/O. Accordingly, using more food sources to improve the overall CIR of LD classifier should be a viable option in this case. Finally, like the sequential version, ABC with number of food sources equals 20 performs comparatively to GA in terms of CIR.

In the third experiment, we try to evaluate the effect of adjusting the *limit* parameter of ABC. As explained in Table I, *limit* is the threshold for an employed bee to escape away from some potential local maximum. We try to vary *limit* from 20 to 40 on data set 2 and record the count of final best CIR that is contributed by the scout bee (referred to as success count hereafter). For example, in Table VII the success count equals 3 when *limit* is set to 20, which means three final best results (out of ten consecutive executions of ABC optimized ANN classifier) are achieved as a result of new food sources discovered by scout bees. However, the success count dropped as *limit* increases, which indicates the effect of scout bee becomes less significant (or completely no contribution when success count drops to zero). Accordingly, it appears that setting *limit* to 20 may be a good balance in our study. In addition, with the original ABC algorithm, the scout bee operates by abandoning all the previous work and then randomly generates a new food source for a fresh start. Instead of just throwing all the efforts so far, we record the best food source (among all exploitation before reaching *limit*) by the employed bee and apply the mutation strategy (as that used by the GA algorithm) on this food source to derive a new one for the scout. The results of the modified mutated strategy (using data set 2) are also presented in Table VII. As we can see, the modified mutated strategy appears to guarantee the contribution of scout bee. Also, in general the averaged CIR does show some improvement as a result of this slight modification.

TABLE VII. EFFECT OF *LIMIT* PARAMETER ON DATA SET 2 (ALL NUMBERS ARE AVERAGES OF 10 CONSECTIVE EXECUTIONS OF CODE)

strategy / *limit*		*20*	*25*	*30*	*35*	*40*
random scout	**CIR**	87.0%	87.0%	86.9%	87.0%	87.0%
	success count*	3	1	0	0	0
mutated scout	**CIR**	87.3	86.9	87.3	87.2	87.0
	success count*	3	2	3	3	2

* success count indicates the number of times that the best results are derived by the scout (out of ten consecutive runs).

In the last experiment, we use ABC or GA as the preprocessing step of each other. More specifically, we perform the procedures depicted in Figure 2 twice by first adopting ABC (or GA) and preserving the optimized ANN parameters and random number seeds in the form of food sources (or chromosomes). The food sources (or chromosomes) are then used as the initial values for the following GA (or ABC) procedure. The two variations are referred to as ABC2GA (first ABC followed by GA) and GA2ABC (first GA followed by ABC), respectively. Both numbers of cycle / generation and food source / population are set to 50 and 100. In the case of ABC, *limit* is set to 20 with the mutated strategy (as used in previous experiment) adopted. The outcomes are listed in Table VIII. Also shown in Table VIII are results in previous studies [10][11][12], referred to as GA-grid1, GA-grid2 and API2APPS.

TABLE VIII. PERFORMANCE COMPARISON AMONG VARIOUS METHODS (ALL NUMBERS ARE AVERAGES OF 10 CONSECTIVE EXECUTIONS OF CODE AND ALL TIME IN SECONDS)

data set / method	1		2		3	
	CIR	*execution time*	*CIR*	*execution time*	*CIR*	*execution time*
GA	87.9%	1,304	85.2%	368	87.3%	641
ABC	88.2%	2,598	87.3%	729	88.9%	1,267
GA2ABC	88.3%	3,870	87.4%	1,092	89.3%	1,892
ABC2GA	**88.4%**	3,881	**87.7%**	1,091	**89.8%**	1,893
GA-Grid1	87.8%	4,931	87.5%	1,623	87.5%	2,521
GA-Grid2	-	-	87.2%	7,769	88.3%	13,234
API2APPS	88.2%	3,207	87.3%	1,285	87.6%	2,178

It appears that ABC2GA performs slightly better than its counterpart (GA2ABC). Yet, most important of all,

 ISBN: 978-1-61208-416-9

both perform better than methods used in all other previous implementations with the same data sets in terms of average CIR. Another matter deserved mentioning is the cost incurred by the leasing of Amazon EC2 virtual hosts for the experiments in [11] (referred to by GA-grid2) was nearly 2500 US dollars (including environment setup, code testing and final production). However, in this study, we have demonstrated that with the right methodologies (like ABC), a 200 US dollars low cost graphics card can achieve much better results both in CIR and execution time.

V. CONCLUSIONS AND FUTURE WORK

In this study, we modify the original ABC algorithm, i.e., the characteristic of scout bee, and applied this modified ABC to enhance the ANN classifier for identifying students with learning disabilities. A low cost GPU is also used to speed-up the ANN training and validation processes. As our experiments show (Table VIII), the resulted solution (ABC-optimized and GPU-assisted ANN classifier) itself not only outperforms its GA counterpart (for as many as 2.1% gain), but also in average achieve better CIR than our previous studies. Furthermore, when this modified ABC is used as the pre-processing step (for finding initial starting points) for GA evolution, the CIRs can be further improved by nearly 1% (data set 3). This is the best that we have ever got on these three data sets in terms of average CIR.

However, there are still a few things that we may be working on. The first of all would be improving the performance in terms of execution efficiency. It appears that our current implementations take too much time in the I/O operations (data transfer between CPU and GPU memory). We will try to port the optimization algorithms (GA or ABC) to GPU to reduce unnecessary I/Os in the future. On the other hand, we may also distribute the optimization code into a number of virtual nodes, which according to our past experience should further improve the CIR, and use (multiple-)GPU as the computation engine.

ACKNOWLEDGMENT

This work was supported in part by the National Science Council of Taiwan, R.O.C. under Grant NSC 103-2511-S-018-007.

REFERENCES

[1] S. A. Kirk, "Behavioral diagnosis and remediation of learning disabilities," Proc. of the Conference on the Exploration into the Problems of the Perceptually Handicapped Child, 1963, pp.1–7.

[2] J. M. Fletcher, W. A. Coulter, D. J. Reschly, and S. Vaughn, "Alternative approach to the definition and identification of learning disabilities: some questions and answers," Annals of Dyslexia, vol. 54, no. 2, 2004, pp. 304–331.

[3] J. Schrag, "Discrepancy approaches for identifying learning disabilities," http://www.specialed.us/discoveridea/topdocs/nasdse/discld.pdf, retrieved: April, 2015.

[4] American Psychiatric Association, "Diagnostic and Statistical manual of mental disorder-V (5th ed)," Washington, DC: Book Promotion & Service LTD, 2013.

[5] Y.-R. Meng and L.-R. Chen, "On discussing the differences about the learning characteristics of LD," Bulletin of Special Education, vol. 233, 2002, pp. 75–93. (in Chinese)

[6] C. L. Nicholson and C. L. Alcorn, "Interpretation of the WISC-III and its subtests," Paper presented at the 25th Annual Meeting of the National Association of School Psychologists, Washington, DC, 1993, pp. 1–16.

[7] T.-S. Huang, "A Study on the characteristics of WISC-III for students with learning disabilities," Master thesis, Graduate Institute of Special Education, National HsinChu University of Education, Hsinchu, Taiwan. (in Chinese)

[8] T.-K. Wu, S.-C. Huang, and Y.-R. Meng, "Evaluation of ANN and SVM classifiers as predictors to the diagnosis of students with learning disabilities," Expert Systems with Applications, vol. 34, no. 3, April 2008, pp. 1846–1856.

[9] T.-K. Wu, S.-C. Huang, and Y.-R. Meng, "Effects of feature selection on the identification of students with learning disabilities using ANN," Lecture Notes in Computer Science, Springer Berlin/Heidelberg, vol. 4221, 2006, pp. 565–574.

[10] T.-K. Wu, S.-C. Huang, Y.-R. Meng, Y.-L. Lin, and H. Chang, "On the parallelization and optimization of the genetic-based ANN classifier for the diagnosis of students with learning disabilities," Proc. 2010 IEEE Conference on Systems, Man and Cybernetics, 2010, pp. 4263–4269.

[11] T.-K. Wu, S.-C. Huang, Y.-R. Meng, and T.-H. Wu, "Experiences on constructing neural network based learning disabilities identification model with the Amazon elastic compute cloud," Paper presented at the 2012 International Conference on Internet Studies (NETs 2012), Ausust 17-19, 2012, Bangkok, Thailand, pp. 1–10.

[12] T.-K. Wu, S.-C. Huang, T.-J. Chang and Y.-R. Meng (2014), Improving learning disabilities students classification accuracy by integrating api and apps algorithms on the hadoop cloud environment, work presented at the 2014 International Conference on Internet Studies (NETs 2014), August 16-17, 2014, Singapore, pp. 1.

[13] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, and J. Vanthienen, "Benchmarking state-of-the-art classification algorithms for credit scoring," Journal of the Operational Research Society, vol. 54, 2003, pp. 627–635.

[14] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, UK, 1995.

[15] E. Cantú-Paz and C. Kamath, "An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems," IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 35, no. 5, 2005, pp. 915–927.

[16] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report-TR06, Erciyes University, Computer Engineering Department, Kayseri/Türkiye, Oct., 2005.

[17] I. Buck, "GPGPU: General-purpose computation on graphics hardware-GPU computation strategies & tricks," ACM SIGGRAPH course notes 8, 2004.

 ISBN: 978-1-61208-416-9