Accelerated Parallel Training of Logistic Regression using OpenCL

HamadaM. Zahera, Ashraf El-Sisi
Faculty of Computers and Information, Menoufia University, Egypt

Abstract: *This paper presents an accelerated approach for training logistic regression in parallel and running on Graphics Processing Units (GPU). Many prediction applications employed logistic regression for building an accomplished prediction model. This process requires a long time of training and building an accurate prediction model. Many scientists have worked out in boosting performance of logistic regression using different techniques. Our study focuses on showing the tremendous capabilities of GPU processing and OpenCL framework. GPU and OpenCL are the low cost and high performance solution for scaling up logistic regression to handle large datasets. We implemented the proposed approach in OpenCL C/C++ and tested on different data sets. All results showed a significant improvement in execution time in large datasets, which is reduced almost with the available GPU devices.*

Keywords: *Logistic Regression, Parallel Computing, GPU, OpenCL.*

Received August 13, 2015; accepted March 24, 2016

1. Introduction

Today, machine learning algorithms have been applied into many problems like prediction, recommendation, and classification. Logistic regression is one of machine learning algorithms used in Predicting tasks; it can anticipate the output value based on a set of attributes or input variables. First, we need to build a logistic model through training with previous instances. The datasets should have a variety of training examples and consider many cases.

The training complexity in logistic regression depends on problem characteristics and datasets volume. Such training like this will take long hours and even days in order to achieve the desired accuracy in logistic model[1]. Also, finding the best-fit setting for building a logistic model needs a certain amount of cross-validation experiments that can also be very time-consuming.

In the past decade, many techniques were proposed to speed the training process. For example, Multithreaded, Multi-core CPUs, Message Passing Interface (MPI) and OpenCL have been investigated into different applications to achieve higher performance. Each technology has its own deployment characteristics and execution cost. We expect an exponential growth of performance shortly. By now, many researchers focus their attention in parallelizing a variety of computational intelligence algorithms [2] based on the recent capabilities of new hardware architectures.

OpenCL is a development framework developed by Khronos group to support general purpose computations on GPUs. OpenCL provides an industry standard for parallel programming of heterogeneous computing platforms[2], it's not dedicated to specific GPU vendors compared to CUDA which is restricted only for Nvidia GPUs. The recent developments of GPU have shown a superb computational performance withthe current multi-core CPUs. Nevertheless, GPUs are specially designed to facilitate accelerated graphics processing; they have been used as general purpose computing devices (often called General Purpose GPU: GPGPU [3]). Certain high-performance GPUs are now designed to solve general-purpose processing instead of graphics rendering, which it was the only usage of GPUs before [4].

The specifics of OpenCL architecture are well considered into our implementation so as to devise the parallelization method carefully. Consequently, this method would be effective in different conditions and useful for future research. The results showed that our parallel implementation has a distinct advantage over the original algorithm, which can be obtained in a wide range of available GPU hardware.

In this paper, we present a GPU implementation of the training process in logistic regression. Our implementation achieved a significant performance, the training process takes few milliseconds rather than seconds in the original algorithm. We encourage researchers in other areas of study who facing problem of large data processing to parallelize their algorithms and boost performance using GPU and OpenCL frameworks. This solution does not require extra resources and make a considerable utilization of machine resources: CPU and GPU. The main contributions of this paper are as follows:

An accelerated and parallel approach for the training process in logistic regression. Our proposed method will help in accelerating processing many complex application (e.g., face detection, speech recognition) in a short time.

- Our implementation is a platform independent. The proposed approach can run on any GPU-supported device.
- Maximize resource utilization and engaging available computing resources in processing such as CPU and all available GPU devices.

This paper is organized as follows. In section 2, we discussed the relevant works that have done before in accelerating the training process with respect to our approach.The proposed approach is presented in detail in section 3 along with sequential training process of logistic regression. Section 4analyses the results obtained from running several experiments on different platforms. Finally, the paper is concluded in section 5.

2. Related Works

Recently, the datasets volumes have grown rapidly by such a way that became difficult to be handled with current implementations. Several techniques were purposed to accelerate the training process of logistic regressions using technologies such as Multithreading, MPI and Open Multiprocessing (OpenMP). Thanh-Nghi *et al.* [6] have presented a new parallel multiclass logistic regression algorithm that achieves high performances for classifying large amounts of images into many classes. The parallel multiclass algorithm is also developed for efficiently classifying large image datasets into a very large number of classes on multi-core computers. They applied two major parallel models: MPI and OpenMP are mostly famous for shared memory. While OpenCL uses the powerful abilities provided in GPU devices. Another approach for training logistic regression in a distributed fashion has been suggested by Yong Zhuang *et al.* [7]. Many interesting techniques are discussed for reducing communication cost and speeding up computation. They have released an MPI-based implementation for public use.

Peng *et al.* [8] analysed three optimization approaches along with two computing platforms to train a logistic model on a large-scale and high-dimensional dataset for classification. Based on extensive experiments. Sequential algorithms with memory intensive operations can perform very well if datasets can fit in memory. For massive datasets, if limited by machine resources, Mahout with its online algorithm is a good choice with a slightly lower precision. The authors considered distributed frameworks to scale up logistic regression in their implementation.

Nowadays, GPU devices are well utilized for executing general applications in addition to processing graphics and games. Complex applications can run on CPU and GPU devices in parallel. This implementation has been investigated before by Kulkarni *et al.* [9] in designing database processing model by linear regression on GPU using CUDA. The authors observed in their results the performance improvement when multithreading architecture and SIMD approach of CUDA are employed. There is a tremendous difference in the results obtained on CPU and GPU. So, CUDA is one of the best programming approaches to optimize the time for various algorithms that require a large amount of data. Also, Mark van Heeswijk *et al.* [10] presented an approach that allows for performing regression on large data sets in reasonable time. The central component of this approach consists in speeding up the slowest operation by running it on GPU, instead of the processor CPU. Experiments show a speedup of an order of magnitude by using the GPU, and enhanced performance on the regression task.

In this paper, we used OpenCL framework for accelerating training processin logistic regression. Our proposed approach is executed on available GPU devices in parallel and achieved the highest performance speed. In thenext sections,we discussour reasons behind choice of OpenCL in the proposed algorithm.

3. Parallelization Using OpenCL

3.1. OpenCL Programming Model

OpenCL [11] is an open standard that defines a parallel programming framework for programs that execute across heterogeneous platforms. Programs that conform to the specifications may be implemented on CPU, GPU, and other devices that have interfaces to the standard specification.

Processing elements in OpenCL are the computing device such as GPU, which may have one or more compute units (e.g. a GPU multiprocessor), which in turn may consist of one or more processing elements (e.g., a GPU scalar processor). The processing elements are accessed by a host, which may be a computer system where OpenCL programs are initiated.

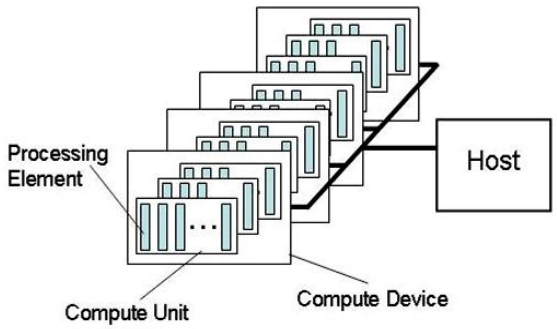

Figure 1. Platform model of OpenCL architecture: one host and one or more compute devices each with one or more compute units each with one or more processing elements [4].

The programming elements in OpenCL are based on work items or kernels, which are equivalent to tasks in parallel programming models, and which execute on a single processing element. A set of multiple work items that are processed in parallel by a computing unit is usually called a workgroup. OpenCL memory model distinguishes different levels of memory:

- Global memory is a multipurpose memory on the compute device that is the largest and the slowest one.
- Constant memory is a part of the global memory that remains unchanged during the kernel execution.

- Local memory is allocated to a computing unit and is smaller and faster than the global memory.
- private memory is the smallest and the fastest (e.g. registers), and is accessible only from the processing element it belongs to.

The global memory is the only way of communication between host and computing device, whereas other types of memory may be used for internal computation.

In case more private memory is requested than physically available, global memory will be used instead.Memory model of OpenCL architecture is shown in figure 2 with respect to the compute units and device.

3.2. Training with Sequential Logistic Regression

Logistic regression is a statistical method for analysing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured as a binary variable (in which there are only two possible outcomes). To build a predictive model using logistic regression, first we need to train this model using previous training instances. The output value is computed based on a set of instance attributes. Each attribute has a coefficient value named attribute's weight. The goal of the training process is to find which are the best weights values that fit this problem and could achieve a high prediction accuracy later on.

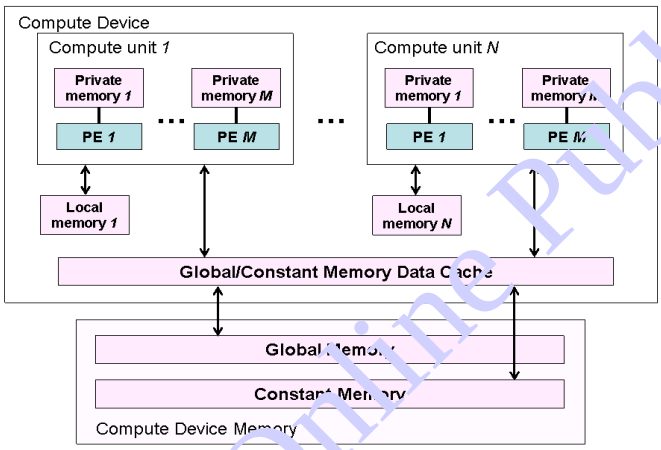

Figure 2. Memory model of OpenCL architecture with respect to the compute units and device [4].

The logistic regression steps are stated in Algorithm1 named sequential training of logistic regression [12]. The algorithm runs sequentially from step2 to step4. The predicted output is computed in step2 using logistic function called sigmoid function that based on instance attributes values.

Algorithm1Sequential Training of Logistic Regression
Initialize weights to small random values
Compute the actual output using sigmoid function

$$Sigmoid(Z) = \frac{1}{1 + e^{-z}}$$

Where : $Z = \sum_{i=k}^{k} W_i . X_i$
X: Instance attribute
W:Attribute weight

1. *Estimate error and update weights using Delta Rule*
2. *If (Actual Output(O) ≠ Target Output(T))*
3. $W_i = W_i + (T - O) \times X_i$

Else : try new Instance

We did a comparative study between sequential and parallel training of logistic regression. Despite in complex problems, the training process requires very large datasets which consume to the extent long time.As shown in Figure 3, the training process goes sequentially to train a logistic model instance by instance. This approach fits well simple prediction problems that require small datasets for training.

Recently, the logistic models are designed to work out complex problems with many attributes. This kind of problems required to be trained with large datasets.

Unfortunately, the sequential logistic regression takes a long time in training and building an efficient model. Many researchers addressed this problem in the past few years using different parallel techniques.

The motivation of our research work is to parallelize training in logistic regression using available resources in a standard PC such as GPU devices as well as running on CPU.

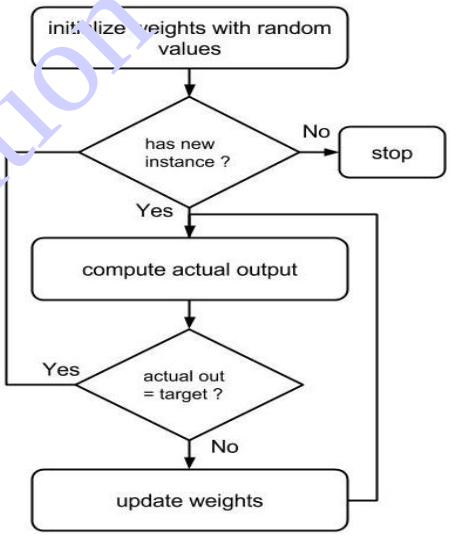

Figure 3. Sequential training of logistic regression.

3.3. Training with parallel Logistic Regression

This section discusses the main aspects of logistic regression implementation that uses OpenCL to exploit multi-core CPU and GPU processors. The proposed logistic regression steps are stated in Algorithm2 named Parallel Training of Logistic Regression.The proposed method includes three main steps in order to train logistic regression with a large-scale dataset in a response time. Main issues required to deal with big data; it should fit memory by processing it in parallel or distributed manner. The first step is to divide training dataset T into N equal parts where N corresponds to a number of parallel kernels.

Algorithm2Parallel Training of Logistic Regression
Initialize weights to small random values
For each node $i \in N$ *do*

Compute the actual output using sigmoid function

$$Sigmoid(Z) = \frac{1}{1 + e^{-z}}$$

Where : $Z = \sum_{i=k}^{k} W_i . X_i$

X: Instance attribute

W:Attribute weight

Estimate error and update weights using Delta Rule

If (Actual Output(O) ≠ Target Output(T))

$$W_i = W_i + (T - O) \times X_i$$

Else: try new Instance

End for

Each part of data can fit in memory and trained by logistic regression. We considered batch training approach because it can be parallelized directly rather than online training. In batch learning [13], we accumulated the error values over all parts and then updated the weights. Secondly, for each part of the dataset; logistic regression is trained by initialized weights with random values at first [14]. Then, train instance by instance to adapt weights and find the best fit values using delta rule as shown in Equation 1.

$$W_{ij} = W_{ij} + \eta \times (T - O) \times X_i \quad (1)$$

Where W_{ij} representsthe weight for instance input $X_i$to node j.T, O corresponds to target value and actual value calculated by the algorithm respectively. In each kernel, the actual output O is computed using a sigmoid function and compared with target value T. Then, weights need to be adapted in case of actual output is not the same as the target value. The algorithm runs in parallel starting from step 2, each kernel output delta weights which used to get final weights in the last step as shown in Figure 4.

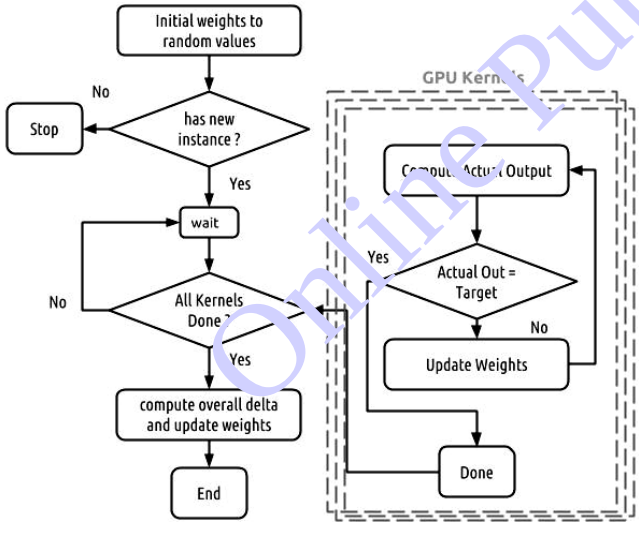

Figure 1. Parallel training of logistic regression.

We considered applying data parallelism in GPU due to many reasons: The GPU-based solutions are cost-effective and don't require any additional resources for extreme processing application.

In our proposed work, we focused on parallelizing the training process of logistic regression, we did not work out the full procedure of logistic regression: training and testing. However, the training step consumes most of processing time in machine learning application. The overall accuracy and precision of logistic regression depends on the training process. The more training examples we apply; the more accuracy we expect. This approach allows to training with large datasets in a short time and try out cross-validation technique [14] that achieve highest possible accuracy with the available datasets.

We used OpenCL C/C++ in GPU programming. Finally, once each kernel finished adapting weights, the final weights are obtained by accumulated all weights come from different kernels as shown in Equation 2.

$$\Delta W_{ij} = \sum_{n=1}^{N} \Delta W_{ij}^{T_k} \quad (2)$$

4. Experimental Results

4.1 Environment Setup

We analysed the performance between sequential and parallel training of logistic regression in different datasets and platforms. We carried out many experiments to estimate the speedup in both approaches. We implemented these experiments using OpenCL C/C. We run them onto two different platforms: platform A includes Mac OSX Yosemite with Intel(R) Core(TM) i5-2415M CPU 2.30GHz and platform B includes Mac OSX Yosemite with Intel(R) Core(TM) i5-3210M CPU 2.50GHz. The following subsections give more information about datasets and results to demonstrate the performance boosting when GPU is involved in the computation.

4.2 Dataset

At first, we have prepared dataset of Amazon product's reviews [15] from raw structure format into a vector space model. Each record has product ID, price, user ID, profile name, helpfulness, score, time, summery and review text. We tried our experiments with various sizes 1000, 5000, 10000, 50000 and 100000 instances as shown in Tables 1 and 2. The speedup in all datasets was measured in microseconds and It can be seen the speed up exponentially in sequential training, while, it goes constant in parallel training. These results demonstrate the computation power in GPU devices and the performance achieved when they were used.

Table 1. Execution times in microseconds between sequential and parallel training on platform a.

Data Size	Sequential Logistic Regression	GPU-Logistic Regression
1000	334	142
5000	1209	159
10000	2188	192
50000	10038	292
100000	15000	349
111000	80000	395

Table 2. Execution times in microseconds between sequential and parallel training on platform b.

Data Size	Sequential Logistic Regression	GPU-Logistic Regression
1000	299	150
5000	1076	165
10000	1947	171
50000	8745	230
100000	15000	272
111000	80000	289

4.3 Result Analysis and Discussion

The results are visualized in Figures 5 and 6 to highlight comparison between execution time in parallel and sequential logistic algorithm. As shown, the performance of proposed parallel approach outperforms sequential training of logistic regression. In Figures 5 and 6, the speedup of GPU-accelerated logistic regression is 11 times faster than the speed of sequential approach.GPU has powerful computing capabilities that point out rendering graphics applications in the past [16]. Lately, GPU-based applications are developed using programming frameworks such as CUDA and OpenCL. CUDA could be used only with Nvidia GPUs. We chose to implement the proposed approach in OpenCL to build platform-independent approach.

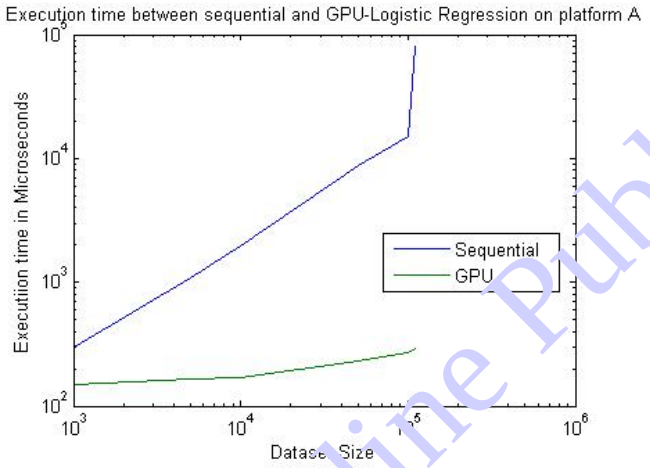

Figure 5. Execution time between sequential and GPU-Logistic Regression on platform A

OpenCL has many other capabilities in building parallel applications on multiple GPU devices. OpenCL supports vector processing which means it can add two vectors of *n* length in the same execution cycle instead of scalar processing in CPU that takes *n*execution cycle. Second, OpenCL provides parallel programming at two levels: data parallelism and task parallelism. In our implementation, we considered data parallelization in the training process to handle larger datasets and process them in parallel. OpenCL provides an efficient solution for many complex problems that usually demand high resources of memory and CPU. OpenCL shows extremely high performance in execution time when the problem is well-designed with OpenCL parallelization parameters: work item and work groups.

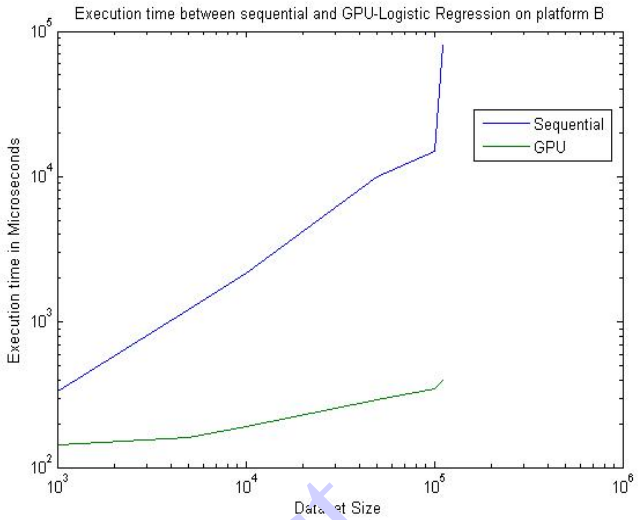

Figure 6. Execution time between sequential and GPL-Logistic Regression on platform B

5. Conclusions and Future Work

In this paper, we proposed a method for accelerating training process of logistic regression using GPU. The speedup is analysed and compared in sequential and GPU-based logistic algorithm. The results showed that OpenCL can afford a new trend in application development to have performance boosting and make a full resource utilization. The OpenCL model allows the implementation to be deployed on different platforms however, the maximum efficiency is achieved by GPU device, the program can also be executed on CPU. The proposed method's speed approximately reaches 11 X faster than the speed of sequential logistic regression. The results demonstrated high impact in performance when taking into account developing GPU-based applications. In future work, we will implement the proposed approach in MapReduce to run it on Hadoop framework as well. The final version of the proposed approach can process big data sets in distributed framework such as Hadoop and parallel on GPU devices

Acknowledgment

This research was partially supported by XEROX Research Centre in Europe. Thanks to Frederic Roulland and Jean-Marc Andreoli for their advice in developing the proposed approach

References

[1] D. W. Hosmer Jr, S. Lemeshow and R. X. Sturdivant, "Applied logistic regression" vol. 398, John Wiley Sons, 2013.

[2] U. Lotri and A. Dobnikar, "Parallel implementations of recurrent neural network learning", Springer, 2009.

[3] M. Harris, "GPGPU: General-purpose computation on GPUs," SIGGRAPH 2005 GPGPU COURSE, 2005.

[4] J. Nickolls and W. J. Dally, "The GPU computing era," IEEE micro, no. 2, pp. 56-69, 2010.
[5] N. Diamantidis, D. Karlis and E. A. Giakoumakis, "Unsupervised stratification of cross-validation for accuracy estimation," Artificial Intelligence, vol. 116, no. 1, pp. 1-16, 2000.
[6] T.-N. Do and F. Poulet, "Parallel Multiclass Logistic Regression for Classifying Large Scale Image Datasets," in Advanced Computational Methods for Knowledge Engineering, Springer, 2015, pp. 255-266.
[7] Y. Zhuang, W.-S. Chin, Y.-C. Juan and C.-J. Lin, "Distributed Newton Methods for Regularized Logistic Regression," in Advances in Knowledge Discovery and Data Mining, Springer, 2015, pp. 690-703.
[8] H. Peng, D. Liang and C. Choi, "Evaluating parallel logistic regression models," in Big Data, 2013 IEEE International Conference on, 2013.
[9] J. B. Kulkarni, A. Sawant and V. S. Inamdar, "Database processing by Linear Regression on GPU using CUDA," in Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference on, 2011.
[10] M. van Heeswijk, Y. Miche, E. Oja and A. Lendasse, "GPU-accelerated and parallelized ELM ensembles for large-scale regression," Neurocomputing, vol. 74, no. 16, pp. 2430-2437, 2011.
[11] K. O. W. Group and others, "The opencl specification, version: 2.0 document revision 22,", vol. 29, 2014.
[12] P. D. Allison, "Logistic regression using SAS: Theory and application" SAS Institute, 2012.
[13] O. Schuessler and D. Loyola, "Parallel training of artificial neural networks using multithreaded and multicore CPUs," in Adaptive and Natural Computing Algorithms, Springer, 2011, pp. 70-79.
[14] Z.-P. Liu, "Logistic regression," in Encyclopedia of Systems Biology, Springer, 2013, pp. 1142-1143.
[15] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in Proceedings of the 7th ACM conference on Recommender systems, 2013.
[16] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips, "GPU computing," Proceedings of the IEEE, vol. 96, no. 5, pp. 879-899, 2008.

Hamada Zaherareceived the B.Sc. and M.Sc. degrees in computers and information from Menoufia University, Faculty of computers and information in 2007 and 2012. Nowheis a teaching assistant at faculty of computers and information since October 2012. His research is concerned with data science with multidisciplinary data mining, machine learning and big data.

Ashraf El-Sisi received the B.Sc. and M.Sc. in Electronic Engineering and Computer Science Engineering from Menoufia University, Faculty of Electronic in 1989 and 1995, respectively and received his PhD in Computer Engineering and Control from Zagazig University, Faculty of Engineering in 2001. His research interest includes cloud computing, privacy preserving data mining, AI approaches in software testing, Intelligent Agent and Intelligent systems.