Intelligent Web Caching Using Neurocomputing and Particle Swarm Optimization Algorithm

Sarina Sulaiman[1], Siti Mariyam Shamsuddin[2], Fadni Forkan[3], Ajith Abraham[4]

[1,2,3] *Soft Computing Research Group,*
Faculty of Computer Science and Information System, Universiti Teknologi Malaysia,
81310 Skudai, Johor, Malaysia.
[4]*Centre for Quantifiable Quality of Service in Communication Systems,*
Norwegian University of Science and Technology, Trondheim, Norway.
sarina@utm.my[1], mariyam@utm.my [2], fuelcon@gmail.com[3], ajith.abraham@ieee.org[4]

Abstract

Web caching is a technology for improving network traffic on the internet. It is a temporary storage of Web objects (such as HTML documents) for later retrieval. There are three significant advantages to Web caching; reduced bandwidth consumption, reduced server load, and reduced latency. These rewards have made the Web less expensive with better performance. In this paper, an Artificial Intelligence (AI) approach is introduced for Web caching to determine the type of Web request, either to cache or not, and to optimize the performance on Web cache. Two methods are employed in this study; Artificial Neural Network (ANN), and Particle Swarm Optimization (PSO). The experimental results have revealed that some improvements have been accomplished compared to the existing technique in terms of Web cache size.

1. Introduction

Caching operation can be executed at the client application, and generally it is embedded in most Web browsers. There are a number of products that extend or replace the embedded caches with systems that contain larger storage, more features, or better performance. In any cases, these systems only cache net objects from many servers for a single user.

Caching can also be operated between the client and the server as a part of proxy cache, which is often located close to network gateways to decrease the bandwidth connections. These systems can serve many users (clients) with cached objects from many servers. In fact, the usefulness of web caching (reportedly up to 80% for some installations) is in caching objects requested by one client for later retrieval by another client. Even for better performance, many proxy caches are part of cache hierarchies; a cache can appeal neighboring caches for a requested document to lessen the need for direct fetching.

Furthermore, caches can be situated directly in front of a particular server, to reduce the number of requests that the server must handle. Most proxy caches can be used in this fashion with different names; reverse cache, inverse cache, or sometimes httpd accelerator, to replicate the fact that it caches objects for many clients but normally from one server [1][8].

The rest of the paper is organized as follows: Section 2 describes some fundamentals of Artificial Neural Network (ANN), followed by Particle Swarm Optimization (PSO) in Section 3. Section 4 discusses on Intelligent Web caching, while Section 5 illustrates the performance evaluation of both ANN and PSO. Finally, Section 6 concludes the article.

2. Artificial Neural Network (ANN)

ANN is comprised of architecture and a learning algorithm. The architecture is the arrangement of the neurons within the network, i.e. how they are linked together. The learning algorithm is the program that each neuron runs every time the network is executed.

Most artificial neurons are a simple summing program by taking the sum of their input activations. Subsequently, the output is activated and it is depended on their internal state. The internal state changes in response to input activations over time, as well as output activations. The supervised learning

paradigm dictates that a network must be informed whether or not it has produced an acceptable response. ANN is judged on its ability to successfully produce a correct output given a certain set of inputs. An unsuccessful attempt induces a change in the neurons internal states.

Back-propagation (BP) is the mathematical technique for calculating errors in a complex mathematical system [9], such as ANN. It is one of the several types of gradient descent algorithms, which are inversely similar to more traditional AI approaches such as gradient assent algorithms. Such algorithms map the function onto a three-dimensional surface, with low land valleys and up land hills. Depending on the problem, the lower the point on the landscape the better the output of the function (this situation is reversed for gradient assent algorithms).

3. Particle Swarm Optimization (PSO)

PSO is an attractive approach due to its easiness in dealing with very few parameters for weight adjustment. The first application represents an approach that can be used for many applications, i.e., evolving ANN [3][9]. PSO is being used to optimize not only the network weights, but also the network structure. The method is straightforward and efficient, and generally, it is widely implemented with traditional ANN training algorithms.

PSO, similar to other evolutionary computation algorithms, can be applied to solve most optimization problems and problems that can be converted to optimization problems. It is a population-based search algorithm derived from the simulation of the social behavior of birds within a flock. The initial intent of the particle swarm concept was to graphically simulate the graceful and unpredictable choreography of a bird flock [2]. The aim is to discover the patterns that govern the ability of birds to fly synchronously, and to suddenly change direction with a regrouping in an optimal formation. From the initial objective, the concept evolved into a simple and efficient optimization algorithm.

A swarm consists of a set of particles, where each particle represents a potential solution. Subsequently, the particles are flown through the hyperspace, and the position of each particle is changed according to its own experience and its neighbors.

Let $\vec{x}_i(t)$ denotes the position of particle P_i in hyperspace, at time step *t.* Subsequently, the position of P_i is changed by adding a velocity $\vec{v}_i(t)$ to the current position.

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t)$$

The velocity vector drives the optimization process and reflects the social exchange information. Moreover, two main algorithms regularly used in PSO are a local best algorithm (lbest) and a global best algorithm (gbest). In the lbest algorithm, each particle moves towards its previous best position, and also towards the best particle in its restricted neighborhood and thus maintains multiple attractors. The gbest algorithm maintains only a single best solution, and each particle moves towards its previous best position and towards the best particle in the whole swarm. Eventually all particles will converge to this position.

4. Intelligent Web caching

Hammami [4] was the pioneer in investigating the possibility of using ANN in placing a new cache block placement. In his work, he adapted the ANN in block placement strategy in computer cache memory widely known as Random Access Memory (RAM). Though in 1990's the setback to this approach is the computational burden on the CPU processing for the ANN learning, his study has marked a new era of caching systems. His promising results on a set of benchmark data has shown and sparked the exploitation of ANN in solving caching problems.

Implementing ANN is an added progress towards improving the performance of Web caching. A significant performance improvement in employing ANN in computer cache memory for data clustering shows that further exploration of executing this technique in Web caching is possible.

The relative performance of ANN in various applications is assorted towards different applications (e.g.: performance analysis, prediction, and data clustering). Performance of various methods and policies in Web caching should be visible once exploring the capabilities of ANN in Web caching. Selecting the best value for each user predefined values such as learning rate and error tolerance is needed in ANN for better results. These selections will affect the forecasting ability of the network in Web caching.

By employing BP and PSO algorithm for the caching scheme analysis in selecting cache objects, a chosen input need to be set up to visualize and handle the environment of the Web caching system. In selecting the best input variables, critical components and variables of affected server, contemporary caching

approaches, and end users' are desirable to be analyzed. The end user perspective is particularly important for online applications while a perspective from a single monitoring server is adequate for most infrastructure applications.

5. Performance evaluation

Several steps are involved to conduct the performance and evaluation of BP algorithm and PSO in Web caching. Figure 1 depicts the workflow of the proposed intelligent Web caching.

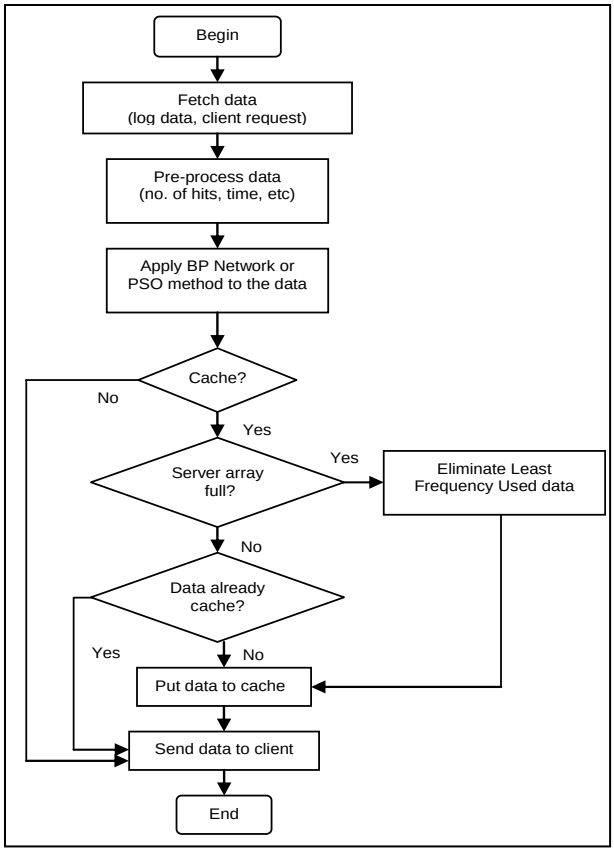

Figure 1. Workflow of intelligent Web caching

The number of ANN input nodes correspond to the number of variables in the input vector used to analyze the system performance. In this paper, the number of inputs is based on three attributes proposed by Mohamed [11]. As well, the number of input nodes is probably the most critical decision variables since it contains the important information about the complex autocorrelation structure in web application performance.

The hidden layer and nodes play crucial roles in mapping the precise weight for the network output. It is the role of the hidden nodes in the hidden layer that allow ANN to identify the feature, to capture the pattern in the web performance data, and to perform complex nonlinear mapping between input and output variables.

In this paper, the number of hidden nodes is determined by using *2n+1* [10]. The number of output nodes is relatively easy to specify as it is directly related to the undertaken problem. In this study, only one output node is needed; about the decision to cache or not to cache the data. BP training parameters for the Web caching are set as follow:

Learning rate = 0.7
Total error = 0.005
Individual error = 0.005
Number of epoch = 20
Number of hidden layer = 1
Number of nodes in hidden layer = 7
Stopping condition = total error reached or maximum number of epoch

For the reason a convergent process for BP is time consuming for this particular data and possibility of not converge, the number of epoch for BP are limited to 20 only.

In designing the ANN for Web caching, the following procedures are determined:

- The pre-processing data
- The number of input nodes
- The number of hidden layers and hidden nodes
- The number of output nodes

The selection of these parameters is basically problem-dependent. Although there exist many different approaches that can be applied, such as the pruning algorithm, the polynomial time algorithm, the canonical decomposition technique, and the network information criterion for finding the optimal architecture of ANN, these methods are usually quite complex in nature and are difficult to implement. Furthermore none of these methods can guarantee the optimal solution for the Web caching problems.

The pre-processing is the key component in Web cache. At this stage, three attributes are proposed, which are based on the attributes that are widely used by the researchers in the area of Web performance analysis [5][6]. The attributes used in this study are:

1. Time: Time is the counter that observes the time takes to receive a data. The time stated in seconds.
2. Script Size: The size of the data that is fetched. The size is expressed in bytes and kilobytes.
3. Numbers of Hit: Observing the number of hits per data. Where on each request done for a Web file, the Number of Hit counter for requested file will be increased.

Each attribute must be multiplied with defined Priority Value (PV) [11] to get the total of the

attributes for target output generation of the network. An example is shown as:

*Expected target = (size *0.266667) + (hit *0.200000) + (time *0.066667)*

The total value determines the expected target for current data. The total value is compared to a threshold number, and this threshold values are dynamic. A new threshold calculation is proposed based on the latency ratio on singular hit rate data [7].

The threshold is calculated and updated for every epoch of the training. If the *expected_target* is smaller than the threshold, then the expected target would be 0, and becomes 1 if the *expected_target* is equal to the threshold and greater shown as given below:

$$Expected\ \ Network\ \ Output = \begin{cases} 0 & \text{if } \text{expected\_target} < threshold, \\ 1 & \text{if } \text{expected\_target} \geq threshold. \end{cases}$$

The network incorporates simplicity in generating output for the web caching to cache or not to cache. For each output generated from the non-training mode, the outputs can be illustrated by employing sigmoid function that bounded between 0 and 1. For each output values that represent between the interval of [0.5,1], the data will be cached in the caching storage, and for each output that represent values less than 0.5 the data will be fetched directly from the originating database resource in case the data is not found in the cache storage. PSO parameters for web caching are assigned as:

Number of particle = 7
Global acceleration constant = 1.4
Local acceleration constant = 1.4
Time step = 0.1
Inertia weights = 0.729844
Minimum error = 0.005

In this section, we present a performance comparison between standard Web cache, ANN Web cache and PSO Web cache. The standard Web cache fills requests from the Web server, stores the requested information locally, and sends the information to the client. If the Web cache gets a request for the similar information for the next time, it simply returns the locally cached data instead of searching over the Internet. On the other hand, ANN and PSO Web cache request from the Web server and determine which request should be stored locally using AI approach.

Figures 1a through Figure 4a depict the BP training process, while Figure 1b through Figure 4b show the PSO training process. The testing log data are from Mosaic clients, which were running at the Boston University from November 1994 to February 1995. Three different parameters are chosen for training; size, retrieval time, and hit rate. The training results reveal the minimum error, total training iteration, accuracy in training both of the approach, and total size in cache (refer to Figures 1a- Figure 4a and Figure 1b- Figure 4b). The minimum error and the accuracy are generated by ANN and PSO during training in standard Web cache.

Table 1 illustrates the results of BP network and PSO. The results are promising in resolving status of data either to cache or not. It is also stored more than half of the data in cache, hence improves the processing time of a server. The advantages of PSO are due to less iteration during training process and achieve minimum error faster compare to the BP network. However, the drawback of PSO is the more the particles are used, the higher the computational cost are incurred.

Moreover, from the results (see Figure 5, 6 and 7) of executing each algorithm, we find the mean squared error and the total training iteration of BP algorithm is higher than PSO from November 1994 to February 1995. Therefore, the percentage of accuracy test for PSO is privileged compare to BP (see Figure 7). This is because PSO is used to optimize the weight in ANN and in this case, the training process using PSO converge faster than BP network. It seems that this result indicates that PSO is preferred.

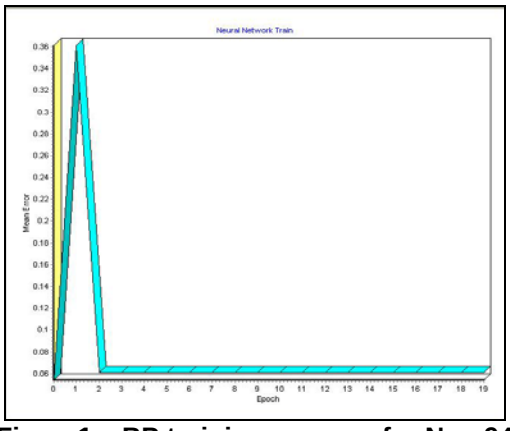

Figure1a. BP training process for Nov-94

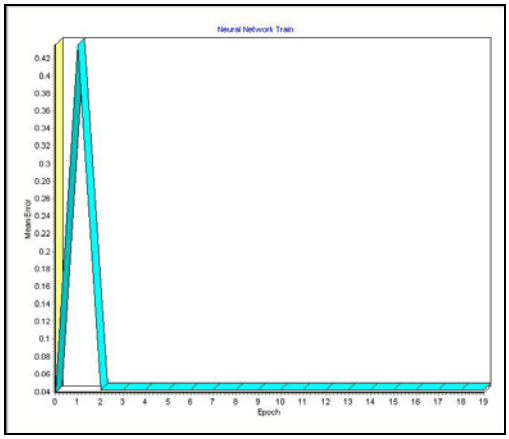

Figure2a. BP training process for Dec-94

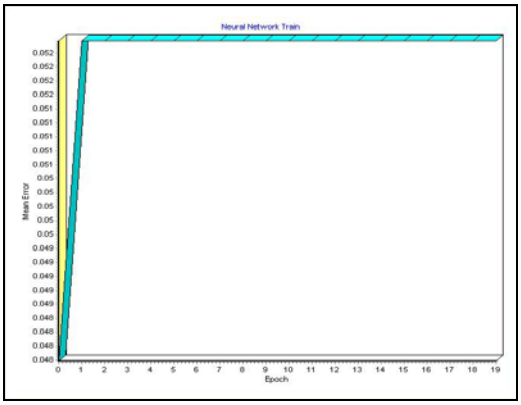

Figure3a. BP Training process for Jan-95

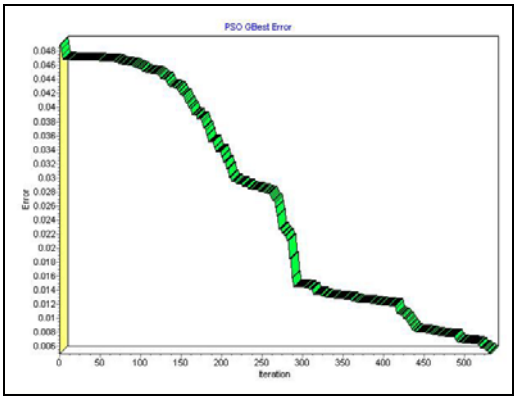

Figure 3b. PSO training process for Jan-95

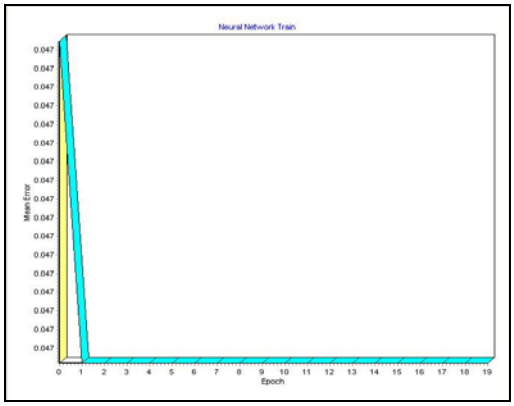

Figure4a. BP training process for Feb-95

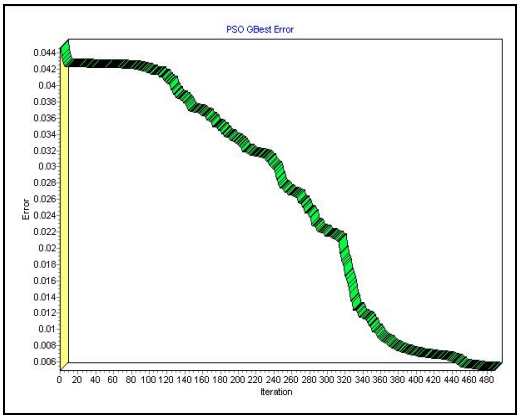

Figure 4b. PSO training process for Feb-95

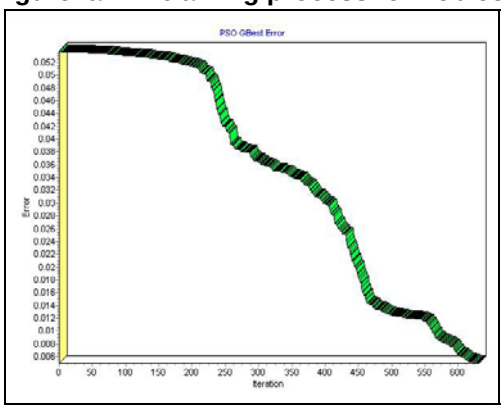

Figure 1b. PSO training process for Nov-94

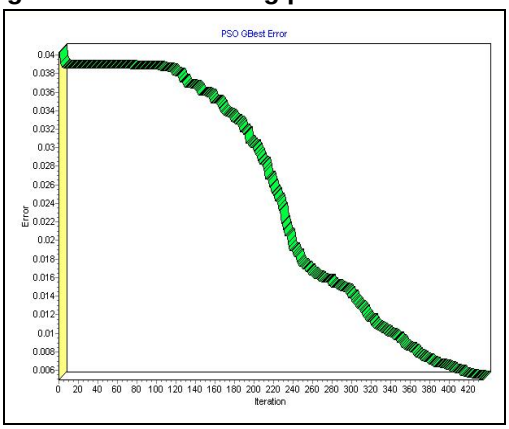

Figure 2b. PSO training process for Dec-94

Table 1. Performance on Nov-94 until Feb-95

	BP	PSO	Standard
November 1994			
Error	0.061	0.005	-
Total Iteration	24520	631	-
Accuracy Test	87.68	99.18	-
Total Cache Size (kb)	57409.67	57409.67	74859.44
December 1994			
Error	0.042	0.005	-
Total Iteration	63740	470	-
Accuracy Test	91.53	98.96	-
Total Cache Size (kb)	102398.4	102398.4	147668.7
January 1995			
Error	0.052	0.0049	-
Total Iteration	67980	533	-
Accuracy Test	89.5	99.441	-
Total Cache Size (kb)	54089.219	54091.349	74779.29
February 1995			
Error	0.047	0.005	-
Total Iteration	57960	488	-
Accuracy Test	90.65	99.68944	-
Total Cache Size (kb)	72084.8	72084.8	84963.98

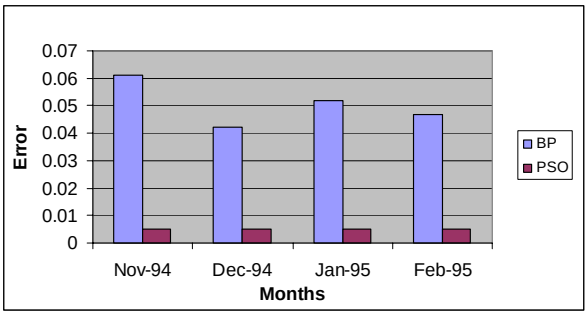

Figure 5. Mean squared error

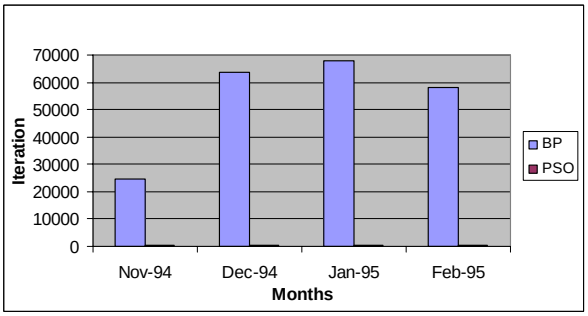

Figure 6. Total training iteration

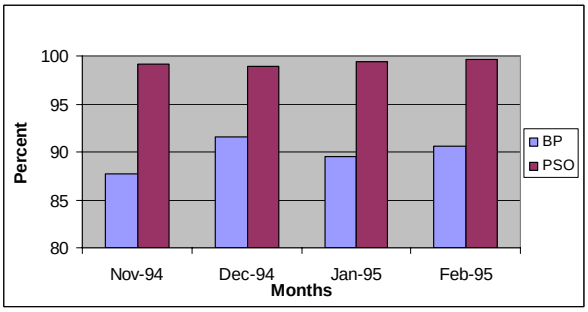

Figure 7. Accuracy test

6. Conclusion

We presented two approaches to optimize the use of Web caching in server. Although further extensive benchmarking is required, there is a strong indication that these approaches enhance the performance of Web cache.

Direction for further research to include the benchmarking with log data from various web and proxy server, as well as different protocol also assigned within the training phase (ftp, gopher etc) to fully optimize the use of intelligent Web cache.

7. Acknowledgement

This work is supported by MOSTI and RMC, Universiti Teknologi Malaysia, MALAYSIA. Authors would like to thank *Soft Computing Research Group*, Faculty of Computer Science and Information System, UTM for their continuous support and fondness in making this study an accomplishment.

8. References

[1]Brain, D. Davison, *Web Caching Overview*, http://www.Web-caching.com/welcome.html.

[2]Kennedy, J., and Eberhart, R. C., Particle Swarm Optimization. *Proc. IEEE Int'l Conf. on Neural Networks IV*, Piscataway, 1995, pp.1942 – 1948.

[3]Eberhart, R. C., and Shi Y., *Particle Swarm Optimization: Development, Application and Resources*, Evolutionary Computation, Volume 1, 2001, pp.27-30.

[4]Hammami, O., *Towards Self Organizing Cache Memories Using Neural Networks.* University of Aizu, under Fukushima. Grant, Japan, 1996, pp.965-80.

[5]Rousskov, A., and Soloviev, V., *On Performance of Caching Proxies*, Short version appears as poster paper in ACM SIGMETRIC'98 Conference, 1998.

[6]Liu M., Wang F. Y., Zeng D., and Yang L., *An Overview of World Wide Web Caching*. IEEE International Conference on. Systems, Man, and Cybernatics, Volume 5, 2001, pp.3045-3050.

[7]Koskela, T., *Neural Network Method in Analysing and Modelling Time Varying Processes*, PhD dissertation, Helsinki University of Technology, 2004.

[8]Sulaiman, S., Shamsuddin S. M., and Abraham, A., *An Implementation of Rough Set in Optimizing Mobile Web Caching Performance*, Tenth International Conference on Computer Modeling and Simulation, UKSiM/EUROSiM 2008, Cambridge, UK, IEEE Computer Society Press, USA, 2008 (in press).

[9]Forkan, F., Sulaiman, S., and Mohammed, F., *Artificial Life and Artificial Neural Network – A Comparison Study*, The 3rd International Seminar on Information and Communication Technology, ICTS, Surabaya, Indonesia, Faculty of Information Technology, Sepuluh Nopember Institute of Technology, 2007, pp.181-185.

[10]Lippmann, R.P., *An Introduction to Computing with Neural Net*, IEEE ASSP Magazine, April, 1987, pp.4-22.

[11]Mohamed, F., *Intelligent Web Caching Architecture*, Master thesis, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, 2007.