Autonomous SPY©: INTELLIGENT WEB PROXY CACHING DETECTION USING NEUROCOMPUTING AND PARTICLE SWARM OPTIMIZATION

Sarina Sulaiman, Siti Mariyam Shamsuddin, Fadni Forkan

Universiti Teknologi Malaysia, Johor, Malaysia
Soft Computing Research Group, Faculty of Computer Science and Information System
sarina@utm.my, mariyam@utm.my, fuelcon@gmail.com

Ajith Abraham

Norwegian University of Science and Technology, Trondheim, Norway
Centre for Quantifiable Quality of Service in Communication Systems
ajith.abraham@ieee.org

ABSTRACT

The demand for Internet content rose dramatically in recent years. Servers became more and more powerful and the bandwidth of end user connections and backbones grew constantly during the last decade. Nevertheless users often experience poor performance when they access Web sites or download files. Reasons for such problems are often performance problems which occur directly on the servers (e.g. poor performance of server-side applications or during flash crowds), problems concerning the network infrastructure (e.g. long geographical distances, network overloads, etc.) and a surprising fact is that many people tend to access the same piece of information repeatedly. Web caching has been recognized as the effective schemes to alleviate the service bottleneck, minimize the user access latency and reduce the network traffic. Consequently, in this paper we discuss an alternative way to tackle these problems with an implementation of Autonomous SPY tool. This tool is capable to self directed either to cache or not to cache the objects in a document based on the behavior of users' activities (number of object hits, script size of objects, and time to receive objects) in Internet based electronic services (e-services) for enhancing Web access.

1. INTRODUCTION

E-services offer great opportunities and challenges for many areas of services, such as government, education, tourism, commerce, marketing, finance, and logistics. They involve various online service providers, delivery systems and applications including e-government, e-learning, e-shopping, e-marketing, e-banking, and e-logistic. Hence, Soft Computing (SC) approach is introduced into Autonomous SPY to determine the type of users' Web request, and to optimize the performance on Web cache. Two methods are employed in this tool; Artificial Neural Network (ANN), and Artificial Life particularly on Particle Swarm Optimization (PSO). Moreover, the novel features of this tool are multi-platform (can be run on more than one computer or server platform), adaptive parameters (parameters change depend on the knowledge discovery data) and load reduction on origin server.

Additionally, the Autonomous SPY is vital to improve the latest Web caching technology by providing virtual client and administrator feedback; hence making Web caching technology practical, efficient and powerful. This tool provides guidance to the administrators or any Internet clients to select the popular parameters to be cached and they can recognize the parameters of data set in proxy caching accordingly.

The rest of the paper is organized as follows: Section 2 describes Web proxy caching, followed by Artificial Neural Network (ANN) and Particle Swarm Optimization (PSO) algorithm in Section 3. Section 4 discusses on Intelligent Web caching architecture, while Section 5 explains The Autonomous SPY tool. Finally, Section 6 concludes the article and future work of the study.

2. WEB PROXY CACHING

Web proxy caches (also known as forward proxy caching) work on the same principle, but at larger scale. Proxies serve hundreds or thousands of users in the same way; large corporations and ISP's (Internet Server Provider) often set them up on their firewalls. Due to large number of users behind proxy caches, they are very good at reducing latency and traffic, since popular objects are requested only once, and served to a large number of clients.

Most proxy caches are deployed by large companies or ISPs that want to reduce the amount of Internet bandwidth that they use. Since the cache is shared by a large number of users, there are a large number of *shared hits* (objects that are requested by a number of clients). Hit rates of 50% efficiency or greater are not uncommon. Proxy caches are a type of *shared cache*. Caches are often deployed as forward proxy caching, reverse proxy caching and transparent caching [1]. Figure 1 shows sample of proxy caching architecture.

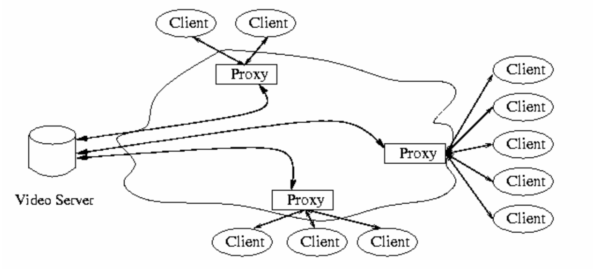


Figure 1. *Proxy caching.*

Some organizations may choose for proxy based caching solutions. They may try to overcome the problem of configuration Web browsers by forcing the use of browsers that provide auto-configuration. Large organizations may make use of network components such as routers and switches [2][3], and may also employ transparent caching. These organizations may prefer highly scalable solutions anticipating future needs. Organizations' Web site with high dynamic content may employ Active Cache [4] or may use Web server accelerators. Furthermore, the subject of measurement of performance is restricted not just to find the efficiency of a given Web caching solution but also to encompass evaluation of the performance of cache replacement algorithms, cache consistency protocols, the role of fundamental protocols such as HTTP and TCP and so on.

3. NEUROCOMPUTING AND PARTICLE SWARM OPTIMIZATION ALGORITHM

Artificial Neural Network (ANN) comprises of architecture and a learning algorithm, with the arrangement of the neurons within the network, i.e. how they are linked together. ANN learning algorithm is the execution of each neuron of the network to minimize the error function. Commonly, the calculations of artificial neurons are simply a summation of their input activations with defined activation function to generate the output value. 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. Unsuccessful attempt induces a change in the neurons internal states.

Back propagation (BP) is a mathematical technique for calculating errors in a complex mathematical system [5], such as ANN. It is one of a number of gradient descent algorithms, which are inversely similar to more traditional artificial intelligence 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). The major limitations of BP algorithm are the existence of temporary, local minima resulting from the saturation behavior of the activation function, and the slow rates of convergence. There are many studies have been done on the optimizations of standard BP algorithm to overcome these problems [14]. One such solution is the integration of Particle Swarm Optimization (PSO) in ANN learning.

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 [6]. PSO is being used to develop 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 algorithm, 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 [7]. 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.

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.

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

The velocity vector drives the optimization process and reflects the social exchange information. There are two main algorithms usually used in PSO, the local best algorithm (lbest) and the 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. Figure 2 illustrates the steps of Artificial Life Neural Network (PSO and ANN) for intelligent Web caching.

```
1. Read and get log data
2. Add log data to array
3. Assign number of hits to log array
4. Add log array to training array
5. Preprocess log data
        5.1 Calculate size
        5.2 Calculate time retrieval
        5.3 Calculate Number of hits
        5.4 Total = size + time+ hit
        5.5 if total > 0.5
                5.5.1 cache = 1
        5.6 else
                5.6.1 cache = 0
        5.7 Find minimum and maximum size
        5.8 Find minimum and maximum time
        5.9 Find minimum and maximum hit
6. Normalize data using min max value
7. Add normalize data to normal array
8. Display actual data
9. Display preprocess data
```

```
10. Display normalize data
11. Initialize input, hidden and output neuron
12. Initialize weights
13. Initialize input, hidden and output array
14. Initialize particles value
        14.1 Generate particle weights and velocity
             array
        14.2 Generate pbest weights and velocity array
        14.3 Generate gbest weights
        14.4 Generate initial fitness value using
             feedforward method
15. Train data
        15.1 if fitness < 0.005 or iteration >20000
                15.1.1 Save gbest weights
        15.2 Update position and velocity
16. Display minimum error/fitness
17. Test data
        17.1 Apply weights to network array
        17.2 Add data to input array
        17.3 Feedforward process
        17.4 Find accuracy
18. Display result
19. Simulation data
        19.2 Add data to input array
        19.3 Feedforward process
20. Display result
```

Figure 2. *Artificial Life Neural Network (PSO and ANN) algorithm.*

4. INTELLIGENT WEB CACHING ARCHITECTURE

Hammami [8] was the pioneer in investigating the possibility of using ANN in a new cache block placement. In his work, he adapted ANN in block placement strategy in computer cache memory widely known as Random Access Memory (RAM). However, 1990's was the setback of this approach due to the computational complexities 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.

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 ANN algorithm and PSO 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. Selection of best input variables, critical components and variables of affected server, contemporary caching approaches, and end users' need to be analyzed to obtain better solutions. The end user perspective is particularly important for online applications; while a perspective from a single monitoring server is adequate for most infrastructure applications. Several steps are involved to conduct the performance and evaluation of ANN algorithm and PSO in Web caching. Figure 3 depicts the workflow of the proposed intelligent Web caching architecture [9].

The proposed intelligent Web caching has been integrated into a workable prototype for better analysis and visualization, so-called Automonous SPY© tool.

5. THE AUTONOMOUS SPY© TOOL

Autonomous SPY is a tool to verify a request data either to cache or not to cache into the Web server. 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 in the future, it simply returns local cached data instead of searching over the Internet. On the other hand, this tool fills request from the Web server and determine which request should be stored locally using Soft Computing (SC) approach.

5.1. Preprocessing and normalize data

The preprocessing is the key component in Web cache. The log data (from November 1994 to May 1995) are obtained from Mosaic clients at the Boston University. Three common attributes have been identified in Web performance analysis [10][11]. Figure 4 shows the actual data prior to data preprocessing and figure 5 depicts the preprocess data. The attributes used in this study are:

1. Time - the counter that observes the time takes to receive a data in seconds (sec.).
2. Script Size – the size is expressed in bytes and kilobytes.
3. Numbers of Hit - the number of hits per data. Each completed request for a Web file will increase the Number of Hit for requested file.

Each attribute must be multiplied with defined Priority Value (PV) [12] 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 [12].

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, else it becomes 1 if the *expected_target* is equal to the threshold and greater as shown below:

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

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.

Normalization process (see figure 6) is done by determine the maximum and minimum value for each attribute. The end values are between 0 and 1 to improve training characteristics.

5.2. Training and testing

The training process is done to train the ANN to generate the desirable output. The process also finds the suitable weight in ANN so that it can generate output that is within the given minimum error. Furthermore, 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* [13]. 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. PSO parameters for web caching are assigned as:

Number of particle = 7
Global cognitive (C1) = 1.4
Local cognitive (C2) = 1.4
Time step (DT) = 0.1
Inertia Weight = 0.729844
Minimum error or stopping condition (Fitness error) = 0.005

Figure 7 illustrates the training process of PSO. It shows that with minimum iteration, the training process has met the stopping condition. The details of network architecture are as follow:

Input node = 3
Hidden node = 7
Output node = 1
Number of particle = 14

From the training, we find that the mean squared error (MSE) is 0.0049 and number of training iteration for PSO algorithm is 539.

Testing process is done to determine the accuracy of the output generated by the ANN and PSO if new or the existing value is used. The accuracy is done base on the difference result between the actual value and the generated value by the ANN and PSO. In this study, the accuracy is measured as follows:

$$Accuracy = \frac{\text{Number of correct data}}{\text{Total data}} \times 100\%$$

Based on this equation, the accuracy of this study is 98.6%.

5.3. Simulation result

Figure 8 shows that the process of Web caching to determine the total size that can be reached in the cache server. The total size of data web cache using PSO and Feedforward (FF) process (data after the training) is smaller compare to the actual data and BP. This is proof that the process in the intelligent cache server (with the PSO and FF) is faster compare to the original and BP cache server. Consequently, the extra space can be used for the other cache server process.

Moreover, Figure 9 depicts the second simulation for the total of time retrieval to fetch the data. As a result, the process to retrieve and send the data from the intelligent cache server (PSO and FF) to the client is faster compared to the original and BP cache server in getting the data from web.

6. CONCLUSIONS AND FUTURE WORK

In this study, an integration of PSO and ANN (FF) in Web caching technology is promising in alleviating the congestion of Internet access mainly for diverse e-services. Therefore, this study has proven that the intelligent cache (PSO and FF) server is smarter contrast to the original and BP cache server. Hence, this situation will affect the size of data in the cache server and time to retrieve the data from the cache server. In the future, we will evaluate the performance analysis of other hybrid soft computing techniques to the Web caching technology.

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 cooperation in making this study a success.

8. REFERENCES

[1] Web Caching. *Caching Tutorial for Web Authors*. 2008. Available from http://www.web-caching.com/mnot_tutorial/intro.html.
[2] Nagaraj, S. V., *Web Caching and Its Applications*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2004.
[3] Krishnamurthy, B. and Rexford, J., *Web Protocols and Practice: HTTP 1.1, Networking Protocols, Caching and Traffic Measurement.* Addison Wesley, 2001.
[4] Cao, P., Zhang, J., and Beach, K., Active Cache:Caching Dynamic Contents on The Web. *Distributed Systems Engineering*, 6(1): 1999, pp. 43-50.
[5] 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.

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

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

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

[9] Sulaiman, S., Shamsuddin, S.M., Forkan, F. and Abraham, A., *Intelligent Web Caching Using Neurocomputing and Particle Swarm Optimization Algorithm*, Second Asia International Conference on Modeling and Simulation, AMS 2008, IEEE Computer Society Press, USA, 2008, pp. 642-647.

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

[11] 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.

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

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

[14] S. M. Shamsuddin, M. N. Sulaiman, & M. Darus. 2001. An Improved Error Signal of Back propagation Model for Classification Problems. *International Journal of Computer Mathematics*, UK, 76(2):297-305, Taylor and Francis Group.

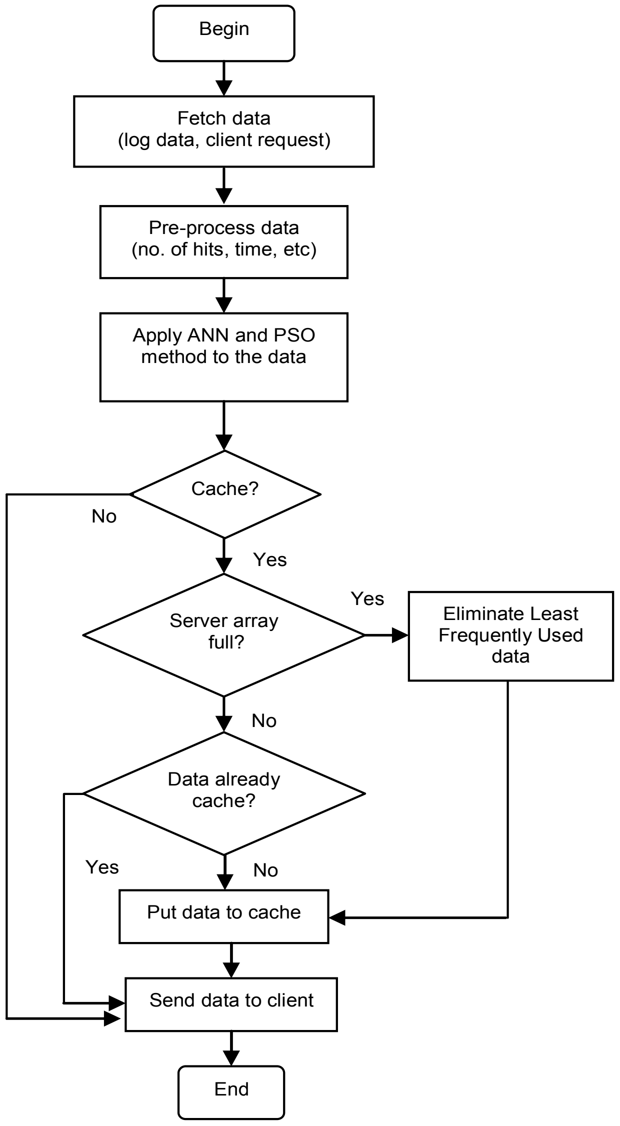
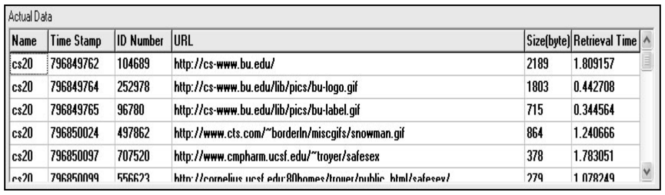


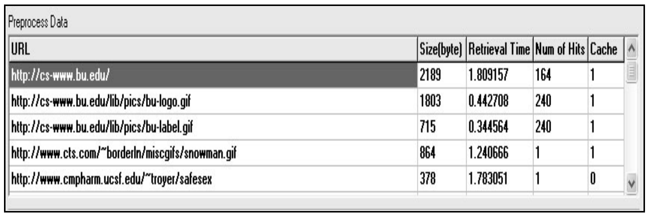
Figure 3. *Workflow of intelligent Web caching.*



Actual Data

Name	Time Stamp	ID Number	URL	Size(byte)	Retrieval Time
cs20	796849762	104689	http://cs-www.bu.edu/	2189	1.809157
cs20	796849764	252978	http://cs-www.bu.edu/lib/pics/bu-logo.gif	1803	0.442708
cs20	796849765	96780	http://cs-www.bu.edu/lib/pics/bu-label.gif	715	0.344564
cs20	796850024	497862	http://www.cts.com/~borderln/miscgifs/snowman.gif	864	1.240666
cs20	796850097	707520	http://www.cmpharm.ucsf.edu/~troyer/safesex	378	1.783051
cs20	796850099	556623	http://cornelius.ucsf.edu:80homes/troyer/public_html/safesex/	279	1.078249

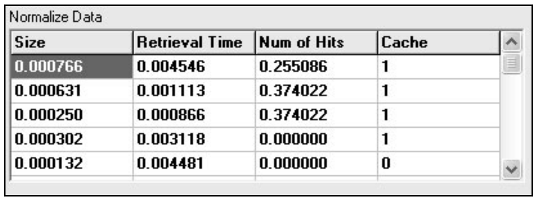
Figure 4. *Example data.*



Preprocess Data

URL	Size(byte)	Retrieval Time	Num of Hits	Cache
http://cs-www.bu.edu/	2189	1.809157	164	1
http://cs-www.bu.edu/lib/pics/bu-logo.gif	1803	0.442708	240	1
http://cs-www.bu.edu/lib/pics/bu-label.gif	715	0.344564	240	1
http://www.cts.com/~borderln/miscgifs/snowman.gif	864	1.240666	1	1
http://www.cmpharm.ucsf.edu/~troyer/safesex	378	1.783051	1	0

Figure 5. *Preprocess data.*


Normalize Data

Size	Retrieval Time	Num of Hits	Cache
0.000766	0.004546	0.255086	1
0.000631	0.001113	0.374022	1
0.000250	0.000866	0.374022	1
0.000302	0.003118	0.000000	1
0.000132	0.004481	0.000000	0

Figure 6. *Normalize data.*

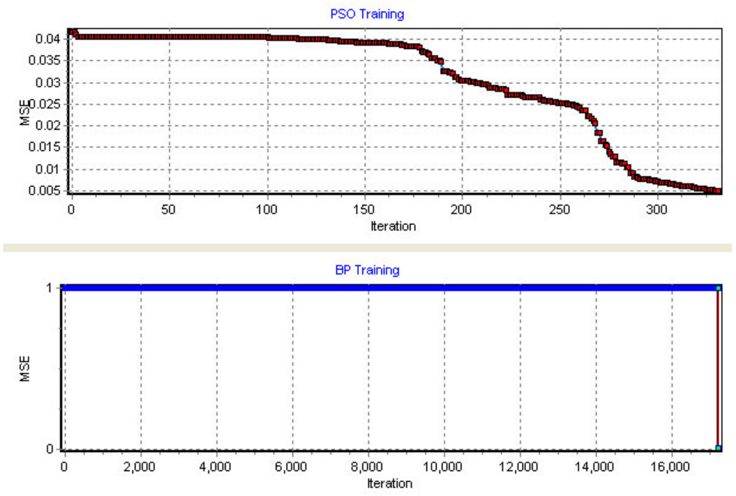


Figure 7. *Training process of PSO and BP data.*

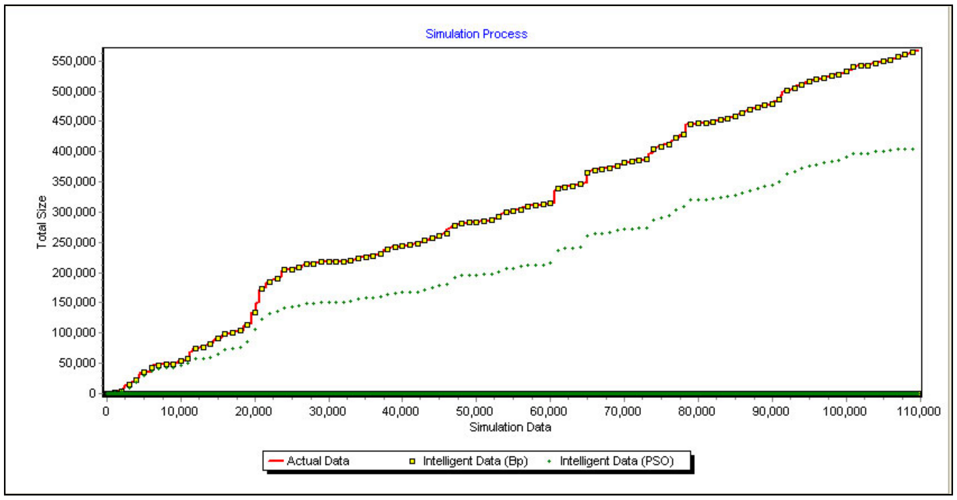


Figure 8. *Simulation result of total document size.*

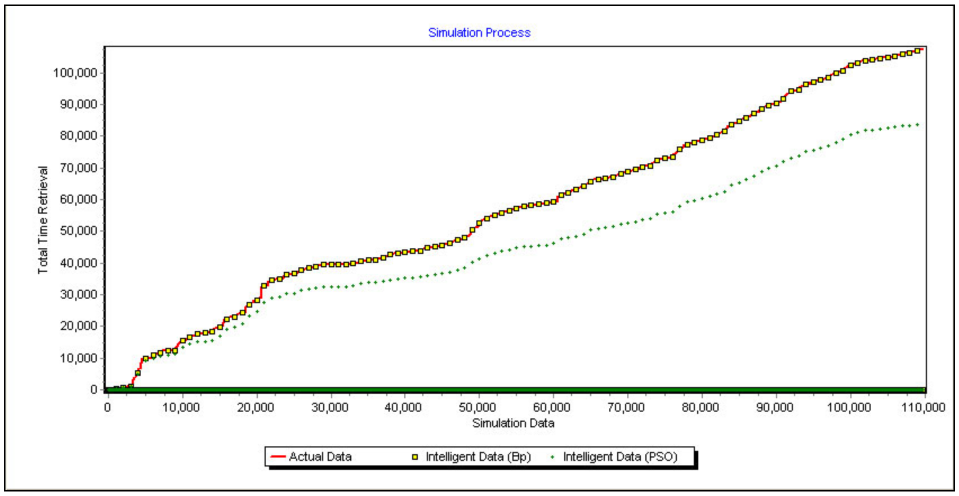


Figure 9. *Simulation result of total time retrieval.*