A Hybrid Optimization Algorithm combining Genetic Algorithm with Neural Network for Web Service Selection

K. Mohan and C. Kalaiarasan

1Department of AM & CS, PSG College of Technology, Coimbatore, Tamilnadu, India.
2Tamilnadu College of Engineering, Coimbatore, Tamilnadu, India.

A B S T R A C T

Web service composition provides a key technology to build loosely coupled, integrated and service oriented applications. The task of service selection that satisfies the user functional requirements is complicated. Due to the availability of the different services with same functionalities around the web, a structure must be designed that facilitates the consumer to select the required services with non-functionality characteristics such as Quality of Services (QoS). This paper shows the advantages of using the neural networks with evolutionary techniques to address the web service selection problem. The structure of the hidden layers in neural network is not modified since the system concerns only with calculation of weights. The genetic algorithm works on the weights of neural network and makes exchange between them using crossover and mutation operators to get advanced a network which produces user satisfied services as output. Experimental results prove that the proposed algorithm outperforms the results obtained by genetic algorithm and neural networks individually.

© 2014 AENSI Publisher All rights reserved.


INTRODUCTION

Web service provides a phenomenon by enabling the computer/human interactions via internet by providing a set of protocols that shares a common architecture to be available at variety of independently developed and deployed systems. As web services are in a dynamically changing environment the service oriented systems uses a mechanism called the late binding to overcome the dynamic problem. Web service composition provides a key technology to build the loosely coupled, integrated and service oriented applications. This composition may contain multiple services that are provided by various service providers with both functional specification (e.g. purchase, payment) and non functional specifications (e.g. reliability, cost, response time). When more than one web service provides same functionality, the selection is based on the user preference QoS values (L. Wang et al., 2012). Several evolutionary algorithms such as GA (Genetic Algorithm), PSO (Particle Swarm Optimization) and Memetic algorithms have been proposed for its robustness and ease of use as global optimisation method to help the user to find a suitable service with preferences for quality attributes. GA is most commonly used method for the selection of web services based on the QoS values. The performance and efficiency of the GA reduces as it has to do a large redundancy repeat because of the fact that it does not use adequate output information. Another drawback of GA is the fixed fitness function which cannot be modified for different composite services and it is a complete automatic process which would lead to miss suitable solution (C. Jin et al., 2008). PSO falls under the local search because the global search takes more iteration time and premature convergence is the main drawback (T. Geetha et al., 2013).

Memetic algorithm is not suitable to solve the non linear continuous multiobjieective optimization problems because the global search phenomena does not work in those cases (C.A. Coello et al., 2002). To overcome the problems caused by EA on efficiency, population diversity and premature convergence when applying to web service selection problem, we have proposed a hybrid algorithm which combines both the GA and Neural Networks. In recent years the evolutionary approach to artificial neural networks has been an emerging technology to solve combinatorial optimisation problems. Here the GA alters the weight values for Neural Networks using the crossover and mutation. The remaining of this paper is organised as follows. Next section describes the related work done by the researchers in this area followed by materials and methods and the hybrid algorithm. Final section gives the experimental results and the conclusion.

Corresponding Author: K. Mohan, Department of AM & CS, PSG College of Technology, Coimbatore, Tamil Nadu, India.
E-mail: mrmohan78@gmail.com
Related work:

In the past decades the researches in web services have focused in many challenging areas from service publication to service mining. Nowadays, QoS based service selection and service compositions have drawn attention of the researchers due to the rapid increase in the services available online and the similar functionalities which they offer. Consumer selects the services based on their preferences and non functionality requirements such as QoS factors. Many researches have been made to solve the web service selection problems. Selection and composition of the web services should be made not only with the functional properties but also with the transactional properties and QoS characteristics. A selection algorithm was proposed that considers the weights assigned to QoS as preferences and defines the transactional requirements (M Manouvrer et al., 2010). A multi objective optimal path selection is made with Ant colony optimization algorithms and is used for the dynamic service composition (Wei Zhang et al., 2008).

Neural Network identifies the services that belongs a variety of objects based on the notion of quality factors. The robustness and efficiency of this is tested with the QoS features in the training and testing phase (Al-Masri. E. et al., 2009). Using the Neuro Fuzzy model the Web service discovery and selection are made. The ranking and the selection of web services are made with the specifications of the neuro-fuzzy model (Abdallah Missaouei et al., 2010). Current Service selection mechanisms consider the static QoS parameters for the services it with little periodically and third drops the service request completely (Huang Baohua et al., 2011). The trustworthiness of the web services. This consists of three layers first drops the service randomly, second drops services it with little periodically and third drops the service request completely (Huang Baohua et al., 2006). Consumers can collect, analyze and make a report of the service providers based on their experience. This report may contain QoS and system context information. Using this client can reconfigure themselves to select best service in a dynamic environment (Zhengdong Gao et al., 2005).

A heuristic approach to the selection problem has been proposed by using the Bees algorithm which inherits the behavior from bees. This technique reduces the search time, produces accurate results, satisfies consumer requirements and considers the QoS properties (Achraf Karray et al., 2013). Neural Network is used to improve the trustworthiness of the web services. This consists of three layers first drops the service randomly, second drops services it with little periodically and third drops the service request completely (Huang Baohua et al., 2006). Consumers can collect, analyze and make a report of the service providers based on their experience. This report may contain QoS and system context information. Using this client can reconfigure themselves to select best service in a dynamic environment (Julian Day et al., 2004).

Two different approaches namely GA and Memetic algorithm have been used to match the consumer’s service request with the service providers based on the QoS attributes is proposed and was compared with the Munkre’s algorithm (Simone A. Ludwig et al., 2011). The composition of web services is done using the genetic algorithm optimized with neural networks which minimizes the neurons with an assumption that the selection is a multistage decision making problem (Yang L et al., 2005). This work provides a motivation for our research since the method of combining genetic algorithm and neural networks, QoS evaluation, service composition and service selection. The drawbacks faced in the above method are optimizing the QoS model and the weights adjustment according to the user preferences. We have tried to solve the above two problems in the following sections.

Hybrid optimisation algorithm:

We have considered a travel plan domain which includes Flight reservation, Transportation and Hotel booking services. Quality of Service (QoS) factors for each web services are expressed by the providers. The user may specify their preferences towards quality attributes by assigning weights to them. For example, a person may show interest in choosing a transport at low cost and hotel at higher cost for luxury.

QoS Evaluation Model:

Quality is an important metric in case of the service selection. Here we consider five quality attributes namely Reliability, Availability, Reputation, Response Time and Cost. Let us assume that there are n similar services in a service repository, each with the above mentioned quality attributes. Then the quality matrix is given by

\[
Q = \begin{bmatrix}
q_{11} & q_{12} & \ldots & q_{15} \\
q_{21} & q_{22} & \ldots & q_{25} \\
\vdots & \vdots & \ddots & \vdots \\
q_{n1} & q_{n2} & \ldots & q_{n5}
\end{bmatrix}
\]

Score of the matrix is given by the sum of products of weight and QoS.

\[
\text{Score} = \sum_{j=1}^{n} w_j \times \sum_{i=1}^{N} (\sum_{j=1}^{3} \frac{q_{ij} - q_{ij}^{min}}{q_{ij}^{max} - q_{ij}^{min}} + \sum_{i=4}^{5} \frac{q_{ij}^{max} - q_{ij}}{q_{ij}^{max} - q_{ij}^{min}})
\]  

(1)
Where $w$ represents weights and $q_{ij}$ represents the attribute matrix, $q_{ij}^{\min}$ refers to the minimum value present in the quality matrix ($Q$) and $q_{ij}^{\max}$ refers to the maximum value in the quality matrix ($Q$). The score is calculated such that the reliability, availability and reputation are maximized and cost, response time is minimized. It is clear from the above function that score depends on weight and quality factors. A web service with highest value is considered to be the optimal service. Sample services for the above scenario in registry are given in Table 1 and Table 2.

Table 1: Sample services in service registry.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Service ID</th>
<th>Service Name</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Flight Booking Service</td>
<td><a href="http://localhost:3006/airindia.asmx">http://localhost:3006/airindia.asmx</a></td>
<td>1001 Air India</td>
<td>1500</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://localhost:3006/spicejet.asmx">http://localhost:3006/spicejet.asmx</a></td>
<td>1002 Spice Jet</td>
<td>1200</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://localhost:3006/jetairways.asmx">http://localhost:3006/jetairways.asmx</a></td>
<td>1003 Jet Airways</td>
<td>3000</td>
</tr>
<tr>
<td>A2</td>
<td>Hotel Booking Service</td>
<td><a href="http://localhost:3006/thepark.asmx">http://localhost:3006/thepark.asmx</a></td>
<td>H01 The Park</td>
<td>5000</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://localhost:3006/royalgrande.asmx">http://localhost:3006/royalgrande.asmx</a></td>
<td>H02 Royal Grande</td>
<td>4800</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://localhost:3006/residency.asmx">http://localhost:3006/residency.asmx</a></td>
<td>H03 Residency</td>
<td>6300</td>
</tr>
<tr>
<td>A3</td>
<td>Cab Providers</td>
<td><a href="http://localhost:3006/srs.asmx">http://localhost:3006/srs.asmx</a></td>
<td>C1 SRS</td>
<td>4000</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://localhost:3006/kpn.asmx">http://localhost:3006/kpn.asmx</a></td>
<td>C2 KPN</td>
<td>4500</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="http://localhost:3006/goodwills.asmx">http://localhost:3006/goodwills.asmx</a></td>
<td>C3 Goodwill</td>
<td>4650</td>
</tr>
</tbody>
</table>

Table 2: Sample services after service composition.

<table>
<thead>
<tr>
<th>Flight ID</th>
<th>Flight Name</th>
<th>Hotel ID</th>
<th>Hotel Name</th>
<th>Cab ID</th>
<th>Cab Name</th>
<th>Reliability</th>
<th>Availability</th>
<th>Reputation</th>
<th>Response Time</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Air India</td>
<td>H03</td>
<td>Residency</td>
<td>C1</td>
<td>SRS</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
<td>0.7</td>
<td>4300</td>
</tr>
<tr>
<td>1003</td>
<td>Jet Airways</td>
<td>H01</td>
<td>The Park</td>
<td>C1</td>
<td>SRS</td>
<td>0.4</td>
<td>0.6</td>
<td>0.5</td>
<td>0.2</td>
<td>3600</td>
</tr>
<tr>
<td>1002</td>
<td>Spice Jet</td>
<td>H02</td>
<td>Royal Grande</td>
<td>C3</td>
<td>Goodwill</td>
<td>0.1</td>
<td>0.9</td>
<td>0.4</td>
<td>0.1</td>
<td>5700</td>
</tr>
<tr>
<td>1001</td>
<td>Air India</td>
<td>H03</td>
<td>Residency</td>
<td>C2</td>
<td>KPN</td>
<td>0.3</td>
<td>0.3</td>
<td>0.6</td>
<td>0.4</td>
<td>3000</td>
</tr>
<tr>
<td>1003</td>
<td>Jet Airways</td>
<td>H01</td>
<td>The Park</td>
<td>C3</td>
<td>Goodwill</td>
<td>0.4</td>
<td>0.5</td>
<td>0.3</td>
<td>0.5</td>
<td>2600</td>
</tr>
</tbody>
</table>

When a request is made the services in service registry are grouped together according to the request and presented as service composition. The QoS values namely Reliability, Availability, Reputation and Response Time are randomly generated whereas the cost is the average of flight, hotel and transport cost.

**Genetic Algorithm:**

Genetic algorithm is used to find approximate solutions to search problems. This works by creating a set of initial population randomly. Fitness is evaluated for each individual with the help of fitness function. The individuals are ranked based on their fitness value. The optimization criteria are checked for the termination.

![Fig. 1: Flow chart of Genetic Algorithm.](image-url)
If the best individual is found the algorithm terminates with the best solution, if it is not found in the current generation, the genetic algorithm continues to modify the population at the subsequent generations. Selection is used to identify which chromosomes in the population are used to produce offspring for the next generation. Tournament selection is the popular selection technique applied in the genetic algorithm because of its efficiency and easy implementation (D. E. Goldberg et al., 1991). In a tournament selection the n individuals are randomly selected from the population and made to compete with each other. Individual that wins is promoted to take part in the next generation. Number of individual in the competition decides the tournament size. Most commonly the tournament size is set to 2. After the selection is process is over, the selected individuals are subjected to crossover. Crossover simulates the generation of offspring from the parents. This takes a part of the string from one parent and other part from other parent and hence a new offspring is formed from these two parents. Three kinds of crossover exists are one-point, two-point and uniform crossover.

Example for one point, two point and uniform crossover is given below (Leung FHF et al., 2001).

\[
\begin{align*}
11001010 + 11011111 &= 11001110 \text{ (One-point)} \\
11001010 + 11011111 &= 11011110 \text{ (Two-point)} \\
11001011 + 11011101 &= 11011111 \text{ (Uniform)}
\end{align*}
\]

GA produces better solution through mutation. In mutation randomly selected bits are flipped is shown below

\[
\begin{align*}
11001000 &\rightarrow 1000100
\end{align*}
\]

Genetic Algorithm is of two types Binary Coded Genetic algorithm and Real Valued Genetic algorithm. Binary Coded Genetic Algorithm takes much time for encoding and decoding but produce optimal solution. When the parameters are too large and when the number of bits needed to represent them is more the coding and encoding process slows down the speed of the system (Nazif. H et al., 2012). Real Valued Genetic Algorithm does not have this problem and most of the optimization problems use real valued parameters. By using this it is convenient to manage the parameters and also improves the efficiency by eliminating the time required for encoding and decoding process. The structure of the genetic algorithm is shown in the Fig 1.

**Neural Networks:**

Neural Networks (NN) finds applications in every field to solve the classification or approximation problem. Each node in the network contains an input and activation function. The activation function is always a summing function. Common activation functions are linear activation and sigmoid activation function. An artificial neural network (ANN) contains an input layer, hidden layer and output layer. Nodes in the input layer denote the number of parameters present in the dataset, output nodes differ according to the domain. Hidden layers are added between the input and output layer to make the network more accurate. The idea behind the neural networks is a chain reaction (i.e) when input layers create output the same will be provided as the input to another input layer and so called BackPropagation (L. Almeida et al., 2008). Training and learning algorithms are used to adjust the weights. These algorithms include Gradient descent, Hebbian learning, LVQ, Widrow-Hoff and Kohonen. Out of these, gradient descent learning is the most commonly used algorithm for training a feed forward network. The flow chart of ANN with gradient descent learning is shown in the Fig 2.

![Fig. 2: Flow chart of ANN with Gradient descent Learning](image-url)
target values. This process continues until the specified level of accuracy is achieved to minimize the predicted error. A cycle of forward-backward pass and weight adjustments using input and output in dataset is called as epoch or iteration. If the network is overtrained it will lose the ability to generalize. Hence, the training or learning method should be chosen suitably. Learning methods in neural network is of three types Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Most common method of unsupervised learning technique used in feed forward networks is delta rule or backpropogation rule. Normally, output of the nodes is function of its inputs. The inputs to the node are products of output of preceding nodes with their corresponding weights. These are summed and passed through activation function before it is sent out from nodes. This can be written as

\[ X_i = \sum w_{ij} a_i \]  
\[ a_j = f(X_i) \]

Here \( X_i \) is the sum of products of weights and outputs obtained from the previous layer i, \( w_{ij} \) is the weight connecting layer i and layer j, \( a_i \) represent the activations of the nodes from the previous layer i, \( a_j \) is the activation of the current node and f is the function which represents activation. The error function is given by,

\[ E_{TE} = \frac{1}{2} \sum_n (t_{jn} - a_{jn})^2 \]

Where \( E_{TE} \) represent the total error of training patterns and \( \frac{1}{2} \) is applied to simplify the derivative of function, n is the output nodes, \( t_{jn} \) represents the target value for the node n in the output layer j and \( a_{jn} \) represents the actual activation node. Error over the entire training pattern is the sum of \( E_{TE} \)

\[ E = \sum_p E_{TE} \]

Where E represents the total error and p represents training patterns. The mean square error is given by

\[ \text{MSE} = \frac{1}{2PN} \sum_p \sum_n (t_{jn} - a_{jn})^2 \]

Here P and N are the total number of training patterns and output nodes. The aim of the gradient descent function is to minimize the MSE (equation (6)). Delta rule is given by

\[ \Delta w_{ij} = -\varepsilon \frac{\delta E}{\delta w_{ij}} = \varepsilon \delta a_i \]

From the equation (7) it is clear that the change in a weight of a particular node is equal to the product of learning rate epsilon (\( \varepsilon \)), difference between target and actual activation function, activation of the input node associated the particular weight during calculation. Each iteration will result in a slight reduce of error. Hence more iteration is required to minimize the error. But this gradient descent learning gets trapped in to local minima. GA enables the learning rules to escape from local minima to avoid the backpropogation algorithm to premature convergence. Neural networks usually use inductive learning (which requires examples) while genetic algorithm uses deductive learning (requires objective function). GA is used in neural networks for three major functions train the weights, design the structure and find optimal learning rule. The algorithm for traditional ANN is shown in Fig 3.

| Step 1: | Randomly initialize weights for ANN. |
| Step 2: | Repeat the process until |
| Step 3: | For each weight \( w_{ij} \) set \( \Delta w_{ij} = 0 \) |
|       |   • Set input units from training data. |
|       |   • Compute value of output units |
|       |   • For each weight \( \Delta w_{ij} \) set \( \Delta w_{ij} = \Delta w_{ij} + \frac{\delta E}{\delta w_{ij}} \) |
| Step 4: | For each weight \( w_{ij} \) set \( w_{ij} = \Delta w_{ij} + \varepsilon \Delta w_{ij} \) |

Fig. 3: Algorithm for Traditional ANN.
Training Neural Networks With Genetic Algorithm:

Many researches have been done in the past decade in using genetic algorithms to adjust the weights of neural networks. Initially this combination found to be slow and now due to the improvements both the algorithm have gained speed and popularity at finding unknown variables (i.e.) the parameters which represent the neural network (Philipp Koehn et al., 1994). In this paper we have used genetic algorithm instead of gradient descent algorithm to overcome the problem of local minima and premature convergence. The flow chart for the GANN is shown in the Fig 4.

![Flow chart of ANN with Genetic Algorithm](image)

**Fig. 4:** Flow chart of ANN with Genetic Algorithm.

An algorithm for the working of GANN is shown in the Fig 5.

![Algorithm for GANN](image)

**Fig. 5:** Algorithm for GANN.

In order to train the system with genetic algorithm first the services has to be represented as chromosomes as shown in the Fig 6. The initial weights are randomly generated which is different from the backpropagation model, since backpropagation uses the probability distribution between -1.0 and 1.0. The focus in this paper is to use real valued genetic algorithm. The fitness function in equation (8) is defined such that the performance of the neural network is improved.

\[
\text{Fitness Function} = \sum_{i=1}^{N} (\sum_{j=1}^{3} \frac{q_{ij} - q_{ij}^{\text{min}}}{q_{ij}^{\text{max}} - q_{ij}^{\text{min}}} + \sum_{i=4}^{5} \frac{q_{ij}^{\text{max}} - q_{ij}^{\text{min}}}{q_{ij}^{\text{max}} - q_{ij}^{\text{min}}})
\]

(8)

To evaluate the fitness function each chromosome QoS values are assigned as weight in the links to the network.

![Encoding of chromosomes](image)

**Fig. 6:** Encoding of chromosomes.
After encoding the chromosomes and performing crossover and mutation operations, the new weights are assigned to the network is shown in Fig 7. The network is then subjected to run with the training samples with the updated weights, this process returns the sum of square of errors. The square error is the difference between the output value obtained and the target data. The error is then back propagated to the genetic algorithm. By doing so the genetic algorithm finds a set of optimized weights which minimizes the error function of network. In evolutionary neural learning the genetic algorithm is used to find the optimal set of weights in a network which minimizes the error function.

**Experimental results:**

To test the efficiency of genetic algorithm in optimizing the weights in neural network, a comparison between gradient descent, a traditional learning algorithm and genetic algorithm was done. Training phase makes the input of the network to produce the desired output. Training a network can be of many types. A popular way to train the network is by adjusting the weights depending on the error values propagated through other layers. Network should works well even in the unknown situations. Overfitting adjusts the training dataset and loses generalisation capability. Hence to overcome this problem the dataset is divided in to two parts: 70% is used for training and remaining 30% for testing. Again the testing data is divided in to 15% of the samples for new training and a validation set consisting of remaining 15%. In the proposed method the fitness of chromosomes are not verified with reference to training but with separate test data set. When this result in an error rate below threshold, the training process is interrupted (Lishoba P.J et al., 2006). A 150x5 matrix was formed with 150 web services with four quality attributes that are generated randomly and used as dataset. Out of 150 samples 70% of the samples were taken for training and 15% is for validation and 15% for testing. The genetic algorithm uses two operators namely mutation and crossover. Here we have considered that two iterations of genetic algorithm is equivalent to one cycle of backpropagation. The reason is that back propagation learning makes forward propogation and calculates errors with outputs produced. These errors are then backpropagated and weights are adjusted.

![Fig. 7: Operation of Operators.](image)

![Fig. 8: Performance Comparison between Gradient Descent and Genetic algorithm.](image)
For this weight adjustment process more computation requires in networks whereas the genetic algorithm performs mutation and crossover at very little computation. We have used learning rate ($\varepsilon$) as 0.5 and different number of training neurons. In all these experiments maximum training epoch is set to 100. The performance comparison of both the algorithms in terms of performance and fitness values are given in the Fig. 8 and Fig. 9.

![Fitness comparison between Gradient Descent and Genetic algorithm](image)

**Fig. 9:** Fitness comparison between Gradient Descent and Genetic algorithm.

**Conclusion:**

The genetic algorithm is proposed to train the neural network for web service selection problem and it is proven to be effective. Both the algorithms have their own strengths and weakness. Back propagation is enough if one adapts fire and forget mentality towards training. But in some cases where the data sets are more complicated the genetic algorithm becomes best. Back propagation works well if the data domain has missing values on the other hand if the search space is huge the genetic algorithm yields desirable results. Our future work is to try different coding schemes to increase efficiency and extend the search space by adjusting the chromosome length and population size. To avoid the overfitting problem the fitness function of genetic algorithm can be improved. Stopping criteria can be designed such that the mean square error reaches a threshold.

**REFERENCES**


