External Examiner Allocation using Artificial Intelligence

Samrat Bachhav
University of Pune
Sinhgad Academy of Engineering.
Kondhwa, Pune-48

Swapnil Patil
University of Pune
Sinhgad Academy of Engineering.
Kondhwa, Pune-48

Hanumant Jagtap
University of Pune
Sinhgad Academy of Engineering.
Kondhwa,Pune-48

Nikhil Nere
University of Pune
Sinhgad Academy of Engineering.
Kondhwa,Pune-48

ABSTRACT
The idea of this paper is to implement the automated external allocation system using knowledge of Artificial Intelligence (AI) and Java programming language. We are developing this project to reduce the work load of Universities. The scope of the work of this paper is to develop an automatic system which assists Universities in allocating the External examiner to colleges. The system will provide online registration facility for colleges and externals. This proposed system will also be able to trace location of colleges and external's home location and will also have an automatic mailing facility. Graphical User Interface (GUI) for the system will be developed using Java (net Beans 7.0 IDE).The tasks handled by GUI are to store details on information of externals and colleges.

General Terms
Ant algorithm, Artificial neural network

Keywords
AI, automatic mailing, automated external allocation, GUI

1. INTRODUCTION
The External examiner Allocation System is the complete solution for universities that will be useful to allocate external examiners to colleges according to externals' specialization and college project domains. This system is essential because university may contain hundreds of colleges and thousands of people may register as external. This should be automated otherwise it is very time consuming and also requires lot of manpower.

There is no existing automated system for this purpose. The existing process includes manual registration of external via printed forms and required documents, sorting of documents according to domains. The allocation of externals to colleges is tedious manual process.

Drawbacks of Existing Process

- Requires more manpower.
- More time Consuming.
- Requires large volume of paper work.
- No automated notification facility.
- External may get project outside his DOMAIN of interest.

To eliminate these drawbacks, an automated system needs to be implemented.

ANNs (Artificial Neural Networks) is part of Artificial intelligence which is used for solving complex problems such as 'Travelling Salesman Problem'. A complex system may be decomposed into simpler elements, in order to be able to understand it.

Networks are one approach for achieving this. There are a large number of different types of networks, but they all are characterized by the following components: a set of nodes, and connections between nodes. The nodes can be seen as computational units. They receive inputs, and process them to obtain an output. This processing might be very simple (such as summing the inputs), or quite complex (a node might contain another network...). The connections determine the information flow between nodes. They can be unidirectional, when the information flows only in one sense, and bidirectional, when the information flows in either sense.

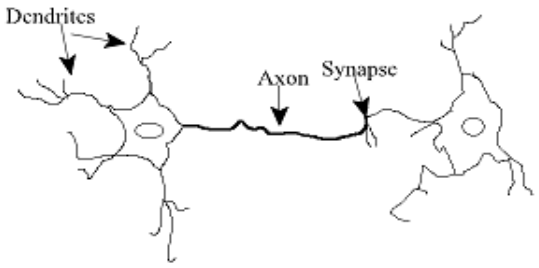


Figure 1 Natural Network

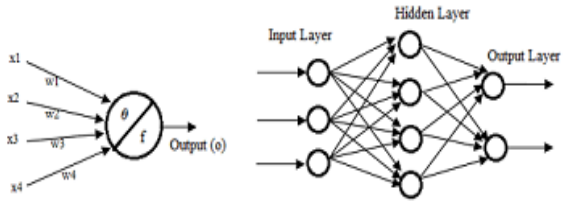
Fig 1 shown above illustrates Natural Networks; Natural neurons receive signals through *synapses* located on the dendrites or membrane of the neuron. When the signals received are strong enough (surpass a certain *threshold*), the neuron is *activated* and emits a signal though the *axon*. This signal might be sent to another synapse, and might activate other neurons.

2. ABOUT NEURAL NETWORK'S

2.1 Background

2.1.1 Introduction

A typical artificial neuron and the modeling of a multilayered neural network are illustrated in Figure 3.



(a) Artificial Neuron. (b) Multilayered artificial neural network.

Figure 3 Architecture of artificial neuron and a multilayered neural network

Referring to Figure 3, The signal flow from inputs $x_1, \ldots, x_n$ is considered to be unidirectional, which are indicated by arrows, as is a neuron's output signal flow (O). The neuron output signal O is given by the following relationship:

$$O = f(net) = f\left(\sum_{j=1}^{n} (Wj)(Xj)\right) \quad \ldots\ldots\ldots\ldots (1)$$

where w_j is the weight vector, and the function $f(net)$ is referred to as an activation (transfer) function. The variable net is defined as a scalar product of the weight and input vectors,

$$net = w^T x = w_1 x_1 + \cdots + w_n x_n \ldots\ldots\ldots\ldots (2)$$

where T is the transpose of a matrix, and, in the simplest case, the output value O is computed as

$$O = f(net) = \begin{cases} 1 & \text{if } w^T x \geq \theta \\ 0 & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots (3)$$

where θ is called the threshold level; and this type of node is called a *linear threshold unit*.

2.1.2 Architecture

The basic architecture consists of three types of neuron layers: input, hidden, and output layers. In feed-forward networks, the signal flow is from input to output units, strictly in a feed-forward direction. The data processing can extend over multiple (layers of) units, but no feedback connections are resent. Recurrent networks contain feedback connections. Contrary to feed-forward networks, the dynamical properties of the network are important. In some cases, the activation values of the units undergo a relaxation process such that the network will evolve to a stable state in which these activations do not change anymore. In other applications, the changes of the activation values of the output neurons are significant, such that the dynamical behavior constitutes the output of the network.

There are several other neural network architectures (Elman network, adaptive resonance theory maps, competitive networks, etc.), depending on the properties and requirement of the application. A neural network has to be configured such that the application of a set of inputs produces the desired set of outputs. Various methods to set the strengths of the connections exist. One way is to set the weights explicitly, using a priori knowledge. Another way is to train the neural network by feeding it teaching patterns and letting it change its weights according to some learning rule.

The learning situations in neural networks may be classified into three distinct sorts. These are supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, an input vector is presented at the inputs together with a set of desired responses, one for each node, at the output layer. A forward pass is done, and the errors or discrepancies between the desired and actual response for each node in the output layer are found. These are then used to determine weight changes in the net according to the prevailing learning rule. The term *supervised* originates from the fact that the desired signals on individual output nodes are provided by an external teacher.

2.2 Choosing initial weights

The learning algorithm uses a steepest descent technique, which rolls straight downhill in weight space until the first valley is reached. This makes the choice of initial starting point in the multidimensional weight space critical. However, there are no recommended rules for this selection except trying several different starting weight values to see if the network results are improved.

3. NEURAL NETWORK LEARNING

Learning was based on the modification of synaptic connections between neurons. Specifically, when an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased. The principles underlying this statement have become known as *Hebbian Learning*. Virtually, most of the neural network learning techniques can be considered as a variant of the Hebbian learning rule. The basic idea is that if two neurons are active simultaneously, their interconnection must be strengthened. If we consider a single layer net, one of the interconnected neurons will be an input unit and one an output unit. If the data are represented in bipolar form, it is easy to express the desired weight update as

$$wi(new) = wi(old) + xio, \ldots\ldots\ldots\ldots (4)$$

where o is the desired output for

$i =$ 1 to n(inputs).

Unfortunately, plain Hebbian learning continually strengthens its weights without bound (unless the input data is properly normalized).

3.1 Perceptron learning rule

The perceptron is a single layer neural network whose weights and biases could be trained to produce a correct
target vector when presented with the corresponding input vector. The training technique used is called the *perceptron learning rule*. Perceptrons are especially suited for simple problems in pattern classification. Suppose we have a set of learning samples consisting of an input vector x and a desired output $d(k)$. For a classification task, the $d(k)$ is usually +1 or −1. The perceptron learning rule is very simple and can be stated as follows:

1. Start with random weights for the connections.
2. Select an input vector x from the set of training samples.

3. If output *yk != d(k)* (the perceptron gives an incorrect response), modify all connections *wi* according to: $\delta wi = \eta(dk - yk)xi$; ($\eta$ = learning rate).

4. ANT ALGORITHM

Ant algorithms were inspired by the observation of real ant colonies. Ants are social insects, that is, insects that live in colonies and whose behaviour is directed more to the survival of the colony as a whole than to that of a single individual component of the colony.

While walking from food sources to the nest and vice versa, ants deposit on the ground a substance called *pheromone*, forming in this way a pheromone trail. Ants can smell pheromone and, when choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations. The pheromone trail allows the ants to find their way back to the food source (or to the nest). Also, it can be used by other ants to find the location of the food sources found by their nestmates.

It has been shown experimentally that this pheromone trail following behaviour can give rise, once employed by a colony of ants, to the emergence of shortest paths. That is, when more paths are available from the nest to a food source, a colony of ants may be able to exploit the pheromone trails left by the individual ants to discover the shortest path from the nest to the food source and back. It has been shown experimentally that this pheromone trail following behaviour can give rise, once employed by a colony of ants, to the emergence of shortest paths. That is, when more paths are available from the nest to a food source, a colony of ants may be able to exploit the pheromone trails left by the individual ants to discover the shortest path from the nest to the food source and back.

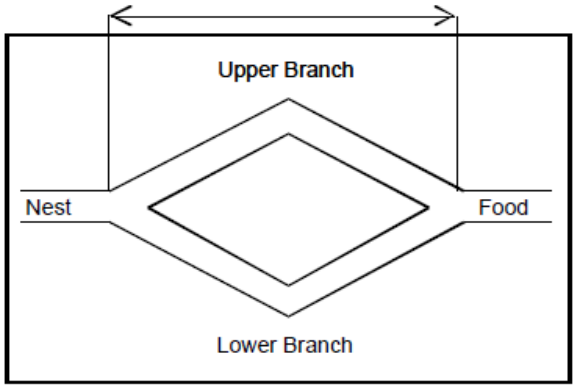


Figure 4 Experimental Setup

In the above experiment initially there is no pheromone on the two branches, which are therefore selected by the ants with the same probability. Nevertheless, random fluctuations, after an initial transitory phase, cause a few more ants to randomly select one branch, the upper one in the experiment shown in Figure 4, over the other. Because ants deposit pheromone while walking, the greater number of ants on the upper branch determines a greater amount of pheromone on it, which in turn stimulates more ants to choose it, and so on.

The probabilistic model that describes this phenomenon, which closely matches the experimental observations, is the following. We first make the assumption that the amount of pheromone on a branch is proportional to the number of ants that used the branch in the past. This assumption implies that pheromone evaporation is not taken into account. Given that an experiment typically lasts approximately one hour, it is plausible to assume that the amount of pheromone evaporated in this time period is negligible. In the model, the probability of choosing a branch at a certain time depends on the total amount of pheromone on the branch, which in turn is proportional to the number of ants that used the branch until that time. More precisely, let *Um* and *Lm* be the numbers of ants that have used the upper and lower branch after *m* ants have crossed the bridge, with *Um* +*Lm* = *m*. The probability *PU*(*m*) with which the (*m*+1)-th ant chooses the upper branch is

$$P_U(m) = \frac{(U_m + k)^h}{(U_m + k)^h + (L_m + k)^h}$$

while the probability *PL*(*m*) that it chooses the lower branch is *PL*(*m*) = 1−*PU*(*m*).

5. PROJECT DEVELOPMENT

Figure 5 shows plan of our project. The designing of GUI will be done using net Beans IDE in Java programming language. Colleges' and Externals' registration will be done initially and administrator will authenticate the data. The authentication will lead to database development.

Algorithm will then trace the location and will start bucket generation. After that Administrator will initiate the AI for allocation process which results in an automatic allocation and a mail will be sent to External. The reply from external will decide whether to add or remove him from next process of allocation.

The terminating condition will arrive only when all buckets are empty

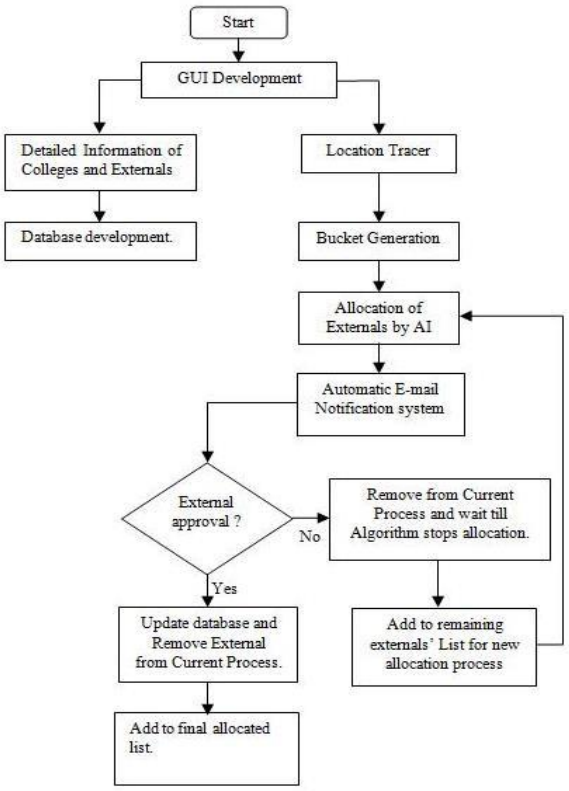


Figure 5 Project Development Algorithm

6. ACKNOWLEDGEMENT

We hereby take this opportunity to record our sincere thanks and heartily gratitude to **Prof. A. S. Shitole** for his useful guidance and making available to us his intimate knowledge and experience for this paper. We are also thankful to HOD **Prof. G.Gandhi** of our Computer department.

7. CONCLUSION

The project plan discussed in the previous section gives a clear perspective that ANN's technology will give the better benefits over the traditional manual allocation system. The system built using Ant algorithm will generate the same result as manual system but in less time and will add more functionalities to existing system making the system automated and thereby reducing human workload.

8. REFERENCES

[1] Artificial Neural Networks Ajith Abraham Oklahoma State University, Stillwater, OK, USA;

[2] Ant Algorithms for Discrete Optimization Marco Dorigo and Gianni Diaro IRIDIA, Universit´e Libre de Bruxelles Brussels, Belgium;

[3] Artificial Neural Networks for Beginners Carlos Gershenson C.Gershenson@sussex.ac.uk;

[4] Ant Algorithm by Claudia Lorenz lorenzc@ee.ethz.ch and Patrizia Mottl mottlp@ee.ethz.ch

[5] http://www.wikipedia.com