Bio-Genetics Journal
Vol. 5, No. 1 (2017): 101-113
Research Article
Open Access

ISSN: 2347-7407

Use of Genetic Algorithm for Hopfield Neural Network to do Logic Programming

Shehab Abdulhabib Saeed Alzaeemi*, Salaudeen Abdulwaheed Adebayo, Mohd Shareduwan M. Kasihmuddin, Mohd Asyraf Mansor and Saratha Sathasivam

School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang Malaysia.

* Corresponding author: Shehab Abdulhabib Saeed Alzaeemi; e-mail: shehab_alzaeemi@yahoo.com

Received: 15 December 2016 Accepted: 18 January 2017 Online: 03 February 2017

ABSTRACT

Genetic algorithms (GAs) are one of the guided random search techniques that use evolutionary ideas of natural selection as an inspiration for solving computational problems. The basic idea behind the study of evolutionary systems is to develop a robust and adaptive search technique. GAs can be used as an optimization tool for engineering problems and other real world complex problems. In this paper, we carried out computer simulations using a developed agent based modelling (ABM) from NETLOGO as the platform to demonstrate and verify the ability of doing logic programming in Hopfield network. GAs was also incorporated into the developed agent based modelling (ABM) using specific procedures to optimize neuron states and energy in the Hopfield network. We then analyzed the GAs performance by comparing the results of global minima ratio, computational time and hamming distance of the GAs with the previous method proposed by Wan Abdullah. We assumed that this is due to the fact that GAs is less susceptible to been trapped in local optima or in any sub-optimal solutions. Hence, it is observed that GAs provide better solutions in finding optimal neuron states and thus, enhance the performance of doing logic programming in Hopfield network.

Keywords: Hopfield network, Genetic Algorithms, Agent Based Modelling, Logic Program.

I. INTRODUCTION

The basic principle of natural selection has been considered as the main evolutionary tool. As generations progress, biological organism evolves based on natural selection the fittest organisms survive and reaches some remarkable forms of accomplishment, while the organisms with poor fitness find it hard to survive and go into extinction, Genetic Algorithms (GAs) were first invented by John Holland in the 1960s and he was referred to as the father of genetic algorithms[1]. Genetic algorithms (GAs) were invented to mimic the process of natural evolution and selection, they were also invented with a view to use this power of evolution to optimize solution in problems[2]. In general, GAs are search algorithms that begin from a set of potential solution / population of "chromosome". (For the purpose of this research strings of +1 and -1 will be used) to a new population (more optimal solutions) by using a kind of natural selection together with the genetics operators of crossover, mutation, and inversion[20].

However, within the sample set, poor solutions are prone to die out, better solutions tend to propagate their good trait using the survival of fittest phenomenon and introducing solutions that has greater potential to the set while the population of the sample set remains constant. GAs have been widely explored on optimization techniques and engineering applications [20].

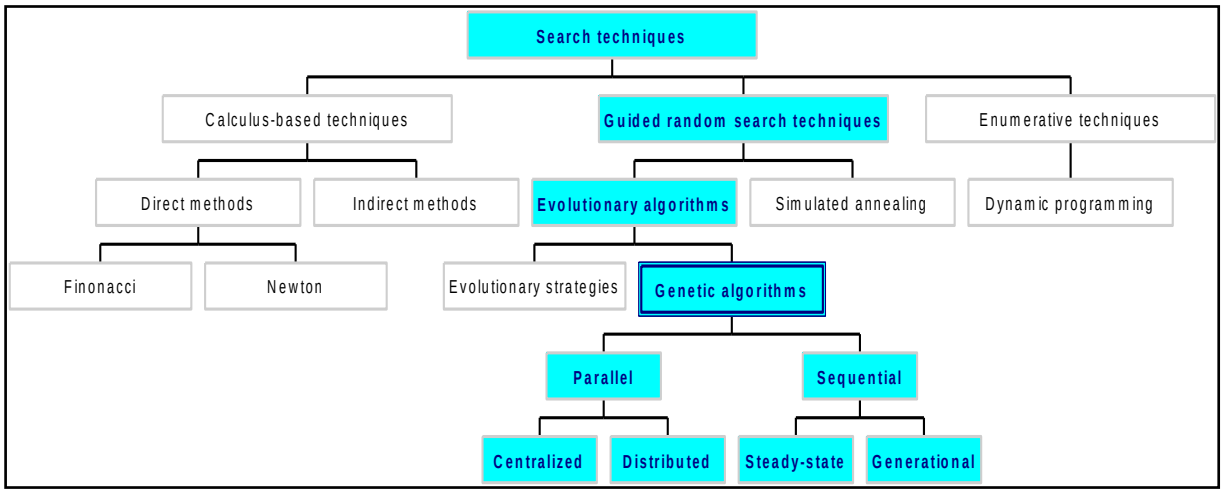

Figure 1 shows classes of search techniques available including Gas [20]

There are many papers including journals, articles and books that discuss the methods of implementing GAs in neural networks. Some of the best available papers were reviewed. The first paper reviewed was a paper by Pires *et.al* [27] he used Hopfield network and GAs to solve shortest problem. The authors introduce modified energy function of Hopfield network, where the energy minimization is conducted in two stages namely energy confinement, E_{conf} and energy optimization, E_{op}. Next paper was from Imada and Araki [2, 3],describe GAs using real-encoded chromosomes which successfully evolves over-loaded Hebbian synaptic weights to function as an associative memory. The authors examined the robustness of Hopfield network by applying more patterns than the saturation level and modified the synaptic weights adaptively using GAs. Small perturbation is given to the Hebbian synaptic weights.

Paper by Garcia *et.al* [8] introduces hybrid Hopfield network-GAs for solving the lights-up puzzle. The proposed algorithm uses Hopfield network to solve part of the puzzle constraints, while the GAs manage other set of constraint as it looks for good quality solution in term of the puzzle's objective function. Somesh and Manu Pratap [9] describe the implementation of GAs to evolve the population of weight matrices for storing and recalling the patterns in a Hopfield network. Using the GAs, efficient set of optimal weight matrix for pattern recall is obtained. The chromosome is represented in a string of parent weight matrix. The authors introduced population generation technique, where the population of the weight matrices is generated using equation that modifies the original weight matrix. In the process of recalling the stored pattern which is a noisy prototype input pattern, Hopfield energy function is used to evaluate the fitness value of each chromosome[29]. Weight matrices with zero fitness value will be selected as the optimal weight matrices for recalling an input pattern efficiently. The authors claim that the GAs is a better searching technique for recalling noisy prototype input patterns compared to Hebbian rule. Standard GAs is used which applies the basic operators namely: selection, fitness evaluation using roulette wheel, crossover, and mutation. In this paper, the Hopfield network acts as a filter for obtaining feasible solutions after the GAs operators, where the Hopfield network repairs the solutions obtained after crossover and mutation operation[27]. In this way, The GAs will always work with feasible solutions.

In this paper, we will focus on Genetic Algorithms (GAs) usage logic programming in neural network. In finding optimal neurons state, we will use GAs method as opposed to the original method of doing logic programming in neural network proposed by Wan Abdullah[10, 11].

The rest of the paper is organized as follows: section II contains Hopfield neural network. In section III, we briefly discussed logic programming in Hopfield network, IV we talk about implementation of logic program in Hopfield network, Section V we discuss the genetic algorithms, Section VI implementation of genetic algorithms in Hopfield network, section VII discuss the agent based modeling, section VIII implementation of agent based modeling, section IX simulation and discussion, section X to conclude the study and in section XI recommendation.

II. HOPFIELD NEURAL NETWORK

Hopfield network was developed by John Hopfield[4]. They are constructed from artificial neurons. There artificial neurons have N inputs, associated to each neuron is a weight denoted by w_i, they also have output. The state of output remain constant until the neuron is updated, updating neurons include the following operations

- The value of each input, x_i is determined and the weighted sum of all inputs, that is $\sum_{i=1}^{N} w_i x_i$ is calculated
- The output state of the neuron is set to +1 if the weighted input sum is greater than or equal to 0, It is set to -1 otherwise
- If the output is denoted by *O,* then the value of *O* is either +1 or -1.

A Hopfield network[6, 5]is a network of *N* interconnected artificial neurons, which are fully connected. The connection weight from neuron *j* to neuron *i* is represented and denoted by the number w_{ij}, in general, the number w_{ij} is symmetric, that is $w_{ij} = w_{ji}$ and in Hopfield network $w_{ii} = w_{jj} = 0$ (no self connection), the set of all such numbers is represented by the weight matrix *W*, whose elements are w_{ij}. Hopfield network are used as a "black box" to calculate some output obtained from a certain self-organization due to the network. Like any other neural network architectures, Hopfield network offer the following usage in most applications[2, 3].

1) Associative memories: the Hopfield network has the capacity to memorize some states, in form of patterns.
2) Combinatorial optimization: if the problem is correctly modeled, the network can give some minimum and some solutions but rarely find the optimal solution.

Hopfield network is also used in the following areas in image processing, signal processing, data de-convolution, pattern matching, solving equations and optimizing functions, travelling salesman, scheduling and resource allocation to mention nut few.

The Hopfield network has demonstrated the interesting features [13, 15, 29]:

1) Distributed representation: A memory is stored as pattern of activation across a set of processing elements, memories can be superimposed upon one another, and different memories can be represented by different patterns over the same set of processing elements.
2) Distributed asynchronous control: Each processing element makes decisions based only on its own local situation. All of these local actions add up to a global solution.
3) Content addressable memory: The network can store a number of patterns. It can retrieve a pattern if only a portion of the pattern is specified. The network will automatically find the closest match.
4) Fault tolerance: The network can still function properly, even if few of the processing elements misbehave or fail completely.

Hopfield network as a *N* interconnected neurons update their activation values asynchronously and independently of other neurons. Each neuron has both input and output which make it different from the classical feed-forward neural network, where the feed-forward network consists of input layers, hidden layers and output layers. Each layer consists of certain number of neurons.

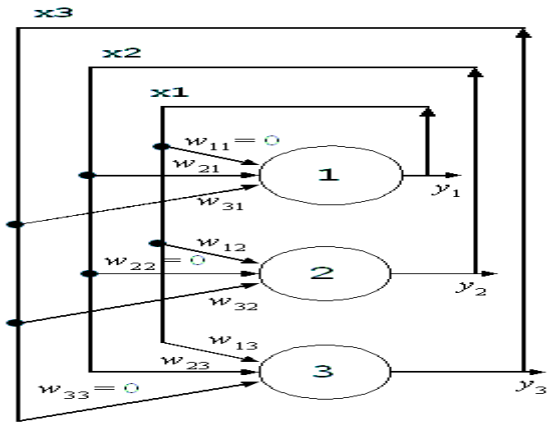

Figure 2: Discrete Hopfield network[14]

From the fact stated earlier that each neuron in Hopfield network produces an output of either +1 of -1, each neuron can only updates its state one at a time , that was the reason why Hopfield network is termed as a discrete- time asynchronous update system which updating rules shown below:

$$S_i(t+1) = f\left(\sum_{i \neq j}^{N} w_{ij} S_i(t)\right) \quad (1)$$

where $S_i(t)$ denoted the state of the *i*th neuron at time t, w_{ij} denoted the connection weight between neuron *j* and neuron *i* , the equation (1) above $f(z) = 1$ if $z \geq 0$ and -1 otherwise where $z = \sum_{i \neq j}^{N} w_{ij} S_j(t)$.

A main application of Hopfield network is an associative memory[2, 3, 14, 12]. The weights have to be set in order for the states of the system which correspond to the pattern to be stored in the network are stable. When the network is prompted with a noisy or incomplete test pattern not in the memory, it will render such data by iterating to a stable state which is near to the prompted pattern which is stored in memory. Hopfield specified the w_{ij} by using the Hebbian rule, as follows

$$w_{ij} = \sum_{\mu=1}^{p} x_i^{\mu} x_j^{\mu} \quad (i \neq j); \quad w_{ii} = 0 \qquad (2)$$

where p is the number of bipolar pattern namely, $X^{\mu} = (x_1^{\mu}, x_2^{\mu}, \ldots x_N^{\mu})$, $\mu = (1, 2, \ldots, p)$, to be stored in associative memory. We can then define an energy function for the discrete Hopfield network by

$$H = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij}(t) S_i(t) S_j(t) \qquad (3)$$

The presence of the Energy function otherwise known as *Lyapunov* function in the equation (3) guarantee convergence into a stable state of activations, however to have to store a set of bipolar patterns in Hopfield network, the weight matrix $W = \{w_{ij}\}$ is given by

$$w_{ij} = \sum_{p} S_i(t) S_j(t) \qquad (4)$$

Therefore equation (3) becomes

$$H = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} S_i^2(t) S_j^2(t) \qquad (5)$$

Basic processing unit for the Hopfield network is the formal neuron of McCulloch and Pitts, where the neuron has two states determined by the level of the induced local field acting on it. For bipolar version, output S_i = +1 indicates that the state of neuron *i* is "on" or "firing", whereas S_i = -1 denotes that the state of neuron *i* is "off" or quiescent". For a network made up of *N* neurons, the state of the network is defined as:

$$S = [S_1, S_2, \ldots, S_N]^T \qquad (6)$$

And the induced local field *vj* of neuron *j* is defined by:

$$v_j = \sum_{i=1}^{N} W_{ij} S_i + b_j \qquad (7)$$

where b_j is fixed bias (negative of the threshold) applied externally to neuron *j*. Therefore, the state S_j of neuron *j* is modified according to the deterministic rule:

$$S_j = \begin{cases} +1 & if \ v_j \geq 0 \\ -1 & if \ v_j < 0 \end{cases} \qquad (8)$$

Thus equation (8) can be written in compact form using lignum function (sgn): $S_j = \mathrm{sgn}[v_j]$ (9)

Precisely, the operation of a discrete Hopfield network to manipulate memories involves two phases, namely storage and retrieval [18]. In other words, Hopfield network forms a content addressable memory (CAM), where the network can reconstruct input patterns that may have been corrupted, and match to the original patterns stored in the network. Details of the two phases are described as follows.

1. Storage Phase. During the storage phase, the network learns the weights after presenting the training examples. The training examples are a set of N-dimensional vectors denote by { ξ_{μ}=1, 2,.., M), where M vectors are called also as fundamental memories, representing the patterns to be memorized by the network. Let ξ, i represent the *i*th element of the fundamental memory $\xi\mu$, for the class μ = 1,2, ..., M. Then, Hebbian learning method is used to compute the synaptic weight from neuron *i* to neuron *j* as defined by equation (4).

2. Retrieval Phase. During the retrieval phase, an N-dimensional vector ξ_{probe} called a probe is imposed on the Hopfield network as its state. The elements of the probe vector are either +1 or -1. Usually, this vector has incomplete or corrupted version of a fundamental memory of the network. Information retrieval then proceeds in accordance with a dynamic rule, in which each neuron *j* of the network randomly, but at some fixed rate, examines the induced local field v_j (including any nonzero bias b_j) applied to it. If, at that instant of time, v_j is greater than zero, neuron *j* will switch its state to +1 or remain in that state if it is already there. Similarly, neuron *j* will switch its state to -1 or remain in that state if it is already there, if *vj* is less than zero. If *vj* is exactly zero, neuron *j* is left in its previous state, regardless of whether it is on or off. The selection of a neuron to perform the updating is done randomly, while the states update from one iteration to next iteration is determinable. The asynchronous updating procedure is continued until there are no further changes to report. That is, starting with the vector ξ_{probe}, the network finally produces a time-invariant state vector y whose individual elements satisfy the following condition for stability:

$$S_j = \mathrm{sgn}[v_j] \qquad j = 1,2 \ldots, N \qquad (10)$$

The state vector that satisfies the condition for stability is called stable state or a fixed point of the state space of the system. Therefore, Hopfield network with

asynchronous dynamics, will always reach to a stable state at local minimum of the energy function.

III. LOGIC PROGRAM HOPFIELD NEURAL NETWORK

A logic program consists of program clauses and is activated by an initial goal statement. States the logic program provides a natural way for problem-solving[7]. The fact that logic programming has gained ground as a vital application in computer science and artificial Intelligence (AI) is undisputable, it provides helpful programming languages for computer to express facts in which intelligent system can comprehend[12]. Logic programming is easy to understand, easier to verify and change compare to other programming where deeper knowledge of understanding matters a lot, thus an inexperienced users and database users might find it very easy to use, Moreover, Logic programming are high level language, easy to prototype, and offer shorter and more readable programs that suits many AI applications[13, 16].

IV. IMPLEMENTATION OF LOGIC PROGRAM IN HOPFIELD NETWORK

Pinkas and Wan Abdullah defined a bi-directional mapping between propositional logic formulas and energy functions of symmetric neural network[11, 21]. Both approaches can handle non-monotonicity logic. Introduced "preferred interpretation" in handling non-monotonicity[21]. Meanwhile, Wan Abdullah's method hunts for the best solutions may change as new clauses added. For example, interpretation with inconsistent logic program can be obtained by using Wan Abdullah's method. Identified a symmetry mapping between cost functions of neural networks and propositional logic formulas[12]. He is interested in using Hopfield network that can handle non-monotonicity of logic to model and solve combinatorial optimization problems. Wan Abdullah's effort revolves around propositional Horn clauses and learning ability of the Hopfield network.

Based on Wan Abdullah's method, the following algorithm summarizes how a logic program can be done in a Hopfield network[12]:

i) Given a logic program, translate all the clauses in the logic program into basic Boolean algebraic form.
ii) Identify a neuron to each ground neuron.
iii) Initialize all connections strengths to zero.
iv) Derive a cost function that is associated with the negation of all the clauses, such that $\frac{1}{2}(1 + S_x)$ represents the logical value of a neuron *X*, where S_x is the neuron corresponding to *X*. The value of S_x is define in such a way that it carries the values of 1 if *X* is true and -1 if *X* is false. Negation (neuron *X* does not occur) is represented by $\frac{1}{2}(1 - S_x)$; a conjunction logical connective is represented by multiplication whereas a disjunction connective is represented by addition.
v) Obtain the values of connection strengths by comparing the cost function with the energy, *E*.
vi) Let the neural networks evolve until minimum energy is reached.
vii) Checked whether the solution obtained is a global solution.

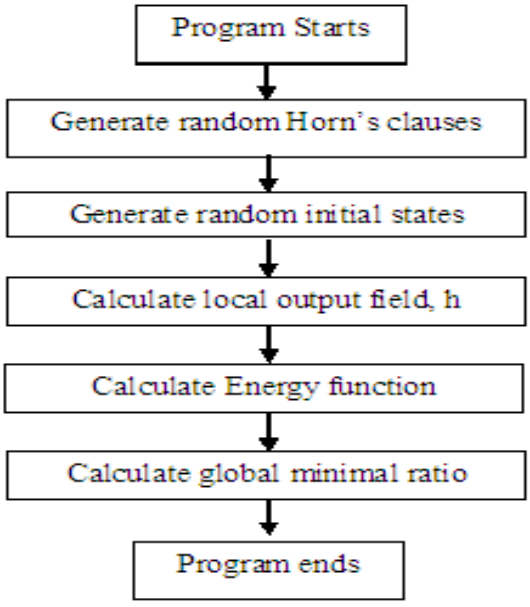

Figure 3: Flow diagram for the Wan Abdullah's Method [13].

V. GENETIC ALGORITHMS

Genetic Algorithms (GAs) were first introduced by John Holland (1962)[1]. Genetic Algorithms (GAs) are adaptive heuristic search algorithm that is based on the evolutionary ideas of natural selection and genetics mostly used in estimating a set of possible solutions to a problem. As such they represent an intelligent exploitation of a random search used to solve optimization problems[20]. Although randomised, but in real sense GAs are by no means random, however, they exploit historical information to fashion the search towards the region of better performance within the search space. GAs are designed to simulate processes in natural systems necessary for evolution, especially those that stick to principles first laid down by Charles Darwin of "survival of the fittest."[22]. GAs are better than conventional AI, because they are more robust[28]. Unlike older AI systems, they do not break easily even if the inputs changed slightly, or in the presence of reasonable noise, GAs may offer significant benefits over more typical search of optimization techniques for instance linear programming, heuristic, depth-first, breath-first, and praxis[20]. The GA aims to use selective `breeding' of the solutions to produce `offspring' better than the parents by combining information from the chromosomes. In GAs, chroiomosome refers to a candidate solution to a problem, often encoded as a bit string. The *genes* are either single bits or short blocks of adjacent bits that encode a particular element of the candidate solution. An *allele* represents either 0 or 1 in a bit string, whereas *crossover* exchanges parts between two single chromosomes (haploid). *Mutation* operator flipped the bit at a random position in the chromosome[19]. Particularly, most applications of GAs employ haploid chromosome. The simplest form of GAs consists of three types of operators[27, 28, 19]:

i) Selection: This operator selects chromosomes in the population for reproduction by giving preference to better individuals in the population, allowing better individuals to pass on their genes to the next generation. The goodness of each individual depends on its fitness value or score this can be determined by an objective function or by a subjective judgement.

ii) Crossover: This operator randomly chooses two individuals from the population using the selection operator, a locus along the bit strings is randomly chosen the subsequence of the two strings are exchanged up to the locus for instance if S_1 =000000 and S_2 =111111 and the crossover point is 2 then S_1'=110000 and S_2'=001111, The two new offspring created from this mating are put into the next generation of the population, by recombining portions of good individuals, this process is likely to create even better individuals.

iii) Mutation: This operator randomly flips some of the bits in a chromosome. Mutation tends to maintain diversity within the population and inhibit premature convergence. It induces a random walk through the search space and mutation and selection (without crossover) create a parallel, noise-tolerant, hill-climbing algorithm, For instance consider, the string S_1 = 10101011 might be mutated in its second position to yield S_1' = 11101011. Mutation can occur at and bit position in a string with very small probability (0.001). Elitism was first introduced by De Jong[19] as a useful and important selection operator that forces the GAs to retain certain number of best individuals at each generation. The fitter chromosomes can get lost if they are not selected to reproduce or if they are destroyed by crossover or mutation which tends to produce an offspring different from the parent. Elitism operator helps to improve the GAs performance in finding solutions because the best chromosomes will be available in each population of chromosomes. A fitness function is a particular type of objective function that prescribes the optimality of a solution (that is, a chromosome) in a genetic algorithm so that that particular chromosome may be ranked against all the other chromosomes. In order for the network to function as associative memory, the instantaneous network state must be similar to the input pattern. The fitness function is then defined as[22]:

$$f = \frac{1}{t_0 p} \sum_{t=1}^{t_0} \sum_{v=1}^{p} m^v(t) \qquad (11)$$

The fitness score 1 implies all the p patterns are stored as fixed points, while fitness less than 1 includes many other cases. Therefore, in this example, fitness value 1 represents the best solution for selection. Another important of GAs operator is selection or reproduction. After assigning fitness value to each chromosome, selection operator will be used to identify the best candidates to reproduce. Selection is based on the concept natural selection and it is one of the main three operators used in genetic algorithm [22].

VI. IMPLEMENTATION OF GENETIC ALGORITHMS IN HOPFIELD NETWORK

Genetic Algorithms (GAs) has been used in finding optimal states in Hopfield Neural Network (HNN) using logic programming in Hopfield network by WAN Abdullah's method[15]. A population of initial states are generated randomly with population of n, where n is equal to the number of generations. In this research, n is equal to 10, the value which is obtained by try and error technique. The chromosomes are represented in form of a string of 1 and -1 allele instead of convectional representation of 1 and 0. In each generation (iteration), the neuron states were modified through the two most important genetic operators (crossovers and mutations) and fitness values of each chromosome which varies for applications are calculated. Below shows the general procedures of implementing of GAs in Hopfield network[15, 16].

1.Random n population of l-string chromosomes is generated. Note that, l is the number of neurons, where each chromosome consists of 1 and -1 alleles.

2.The network is allowed to evolve until minimum energy is reached. The final state is tested (state after the neuron is relaxed/minimum energy is reached) obtained whether it is a stable state by running five times. If the states remain the same for five runs, then it is considered stable state [14].

3. Fitness value for each of chromosome is calculated, for this research work the fitness function is calculated using the below relation[22]. The fitness evaluation of chromosome *j*, *f(j)* is as follows:

$$f(j) = \frac{1}{2N}\sum_{i=1}^{N}\left[N_i^I(j) - N_i^F(j)\right] \times \frac{100\%}{1} \qquad (12)$$

where N is the number of neurons, N_i^I and N_i^F are the initial states and final states of neuron i respectively.

4. Steady state selection method is used to choose the best chromosomes based on their fitness value. This is done by sorting the chromosomes, and the best two chromosomes are crossed- over to form two new offspring. The crossover is chosen at random point based on user preference.

5. The new off spring is inserted in the current population, and the chromosome of lowest fitness value is die out/ removed.

6. One randomly chosen chromosome, k is mutated at random point, t with probability pm = 0.001 and the resulting chromosome is added into the current population, replacing the chromosome k, this done to create chromosome that might be difficult to obtain through crossover[32].

7. Step 2 is repeated for n generations/iterations.

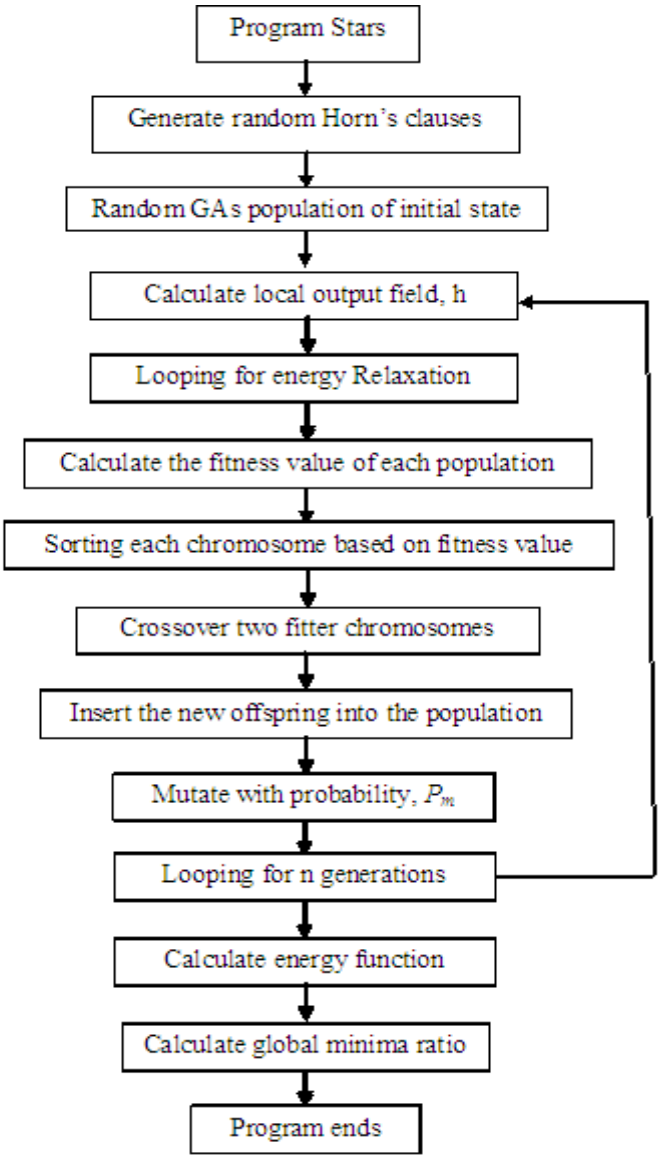

Figure 4: Flow diagram for Genetic Algorithms Method

VII. AGENT BASED MODELLING

After knowing the efficiency of doing logic programming in Hopfield network, we will used NETLOGO as a platform to develop agent based modelling (ABM). We will designed an agent based modelling to implement the Hopfield network in doing logic programming. Netlogo was designed and authored by Uri Wilensky, director of Northwestern University's Center for Connected Learning and Computer-Based Modelling [23, 24, 25]. Agent-based modeling (ABM) is a technique increasingly used in a broad range of social sciences[26]. It involves building a computational model consisting of "agents," each of which represents an actor in the social world, and an" environment"[31]. Netlogo is an agent based programming language and integrated modeling environment. Netlogo was designed, in the spirit of the Logo programming language, to be "low threshold and no ceiling". It teaches programming concepts using agents in the form of *turtles*, *patches*, *links* and the *observer*. Netlogo was designed for multiple audiences in mind, in particular: teaching children in the education community and domain experts without a programming background to model related phenomena. Many scientific articles have been published using Netlogo.

While for the link agents, they connect the mobile agents to make networks, graphs and aggregates which can let the users get more understanding on output of system. Moreover, its runs are exactly reproducible cross-platform [17, 30].

The model can be viewed in either 2D or 3D form. Programmer can choose any interesting agent shapes to design the agent based modelling, and some interface builders like buttons, sliders, switches, choosers, monitors, notes, output area in the agent based modelling can be developed too. Those interface builders are ready to use and programmer do not need to write more programming language from them. Later in the next section will carry out the definition and more explanation of simulator and benefits of agent based modelling in NETLOGO [17].

VIII. IMPLEMENTATION OF AGENT BASED MODELLING

In this paper work, we have proposed a new Genetic Algorithm (GA) with Logic programming in Hopfield network method which is used to find the optimal neuron states. The performance of the proposed new GA was later compared with the original method of doing logic programming in Hopfield network proposed by Wan Abdullah subsequently referred to as Wan Abdullah's method.

The experiment was run for various numbers of neurons (NN) from 10 up to 80 and number of literals per clause (NC1, NC2, NC3). The results shown in Figures 6,7,8 represent NN=20, 30, 40, 50, 60, 70 and 80. The network complexity increases as the number of neuron (*NN*) increases. The development of agent based modeling was done using Microsoft Window 7 professional 64-bit, with the following specification (500GB hard disk, 4096MB RAM and processor 3.40GHz) since the computer specification play a significant role in the performance of the Agent Based Modelling (ABM). The developed ABM was design in such a way that latest version of NETLOGO (5.3.1), tools and techniques were utilized. The interface of the ABM was designed in such a way the programmer/ user see the series of procedures and stages involves so he/she has flexibility to adjust input parameters at the beginning of the programme and see the results of the global minima, hamming distance, computational time to mention but few at the end of the simulation. Furthermore, all simulations were done using computers (8 units) with the same specification in order to guarantee consistency in the results of global solutions, running time and hamming distance for all calculation.

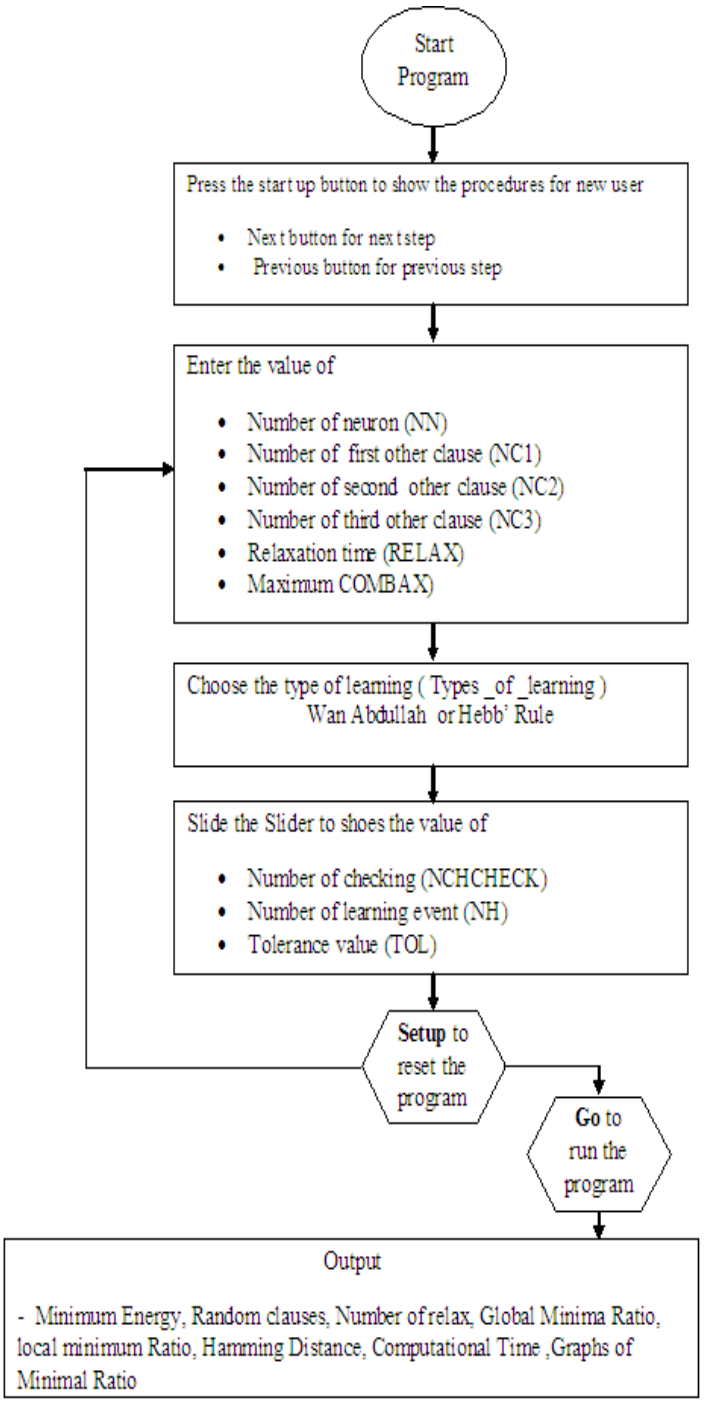

Figure 5 Flow chart of the ABM [17].

This is simply the explanation of the procedures in the flow chart above, the procedures can be classified into two (2) different stages namely, the entering of values phase and training phase

Phase 1: Entering of values

1) Press the start up / Reset Quick-Start button for the new user.

a) User can press next button to go for the next step and previous button to go for the previous step.

2) After that enter the values of NN, NC1, NC2, NC3, RELAX and COMBMAX.

a) The maximum of NN is 80, the value of NN, NC1, NC2, NC3 takes values from 1 to 80, but for any trial, the number of NN must greater than or equal to the value of NC1, NC2, and NC3 that is (NN $\geq$ NC1, NC2, NC3). For any value of the NN that had been entered The maximum number of RELAX and COMBMAX is 100. All these values are chosen by try and error technique.

3) Choose type of learning which is either Hebb's rule or Wan Abdullah's method for this research all results were obtained using Wan Abdullah's as the learning type.

4) the slider is slide such that to choose NCHCHECK, NH, TOL and NT.

a) The maximum of NCHCHECK, NH and NT are 100 while the maximum of TOL is 0.01.

5) After all the values had been set, Click on the setup button to fix and set those values in the program.

6) Then, click go button to run the program.

a) The program will generate random program clauses. Example, if user declared NC1 as 2, then 2 first order clauses will be generated.

Phase 2: Training

7) Initial states for the neurons in the clauses are initialized

a) Next, based on the idea for the Hopfield network that originated from the behaviour of neurons (particles) in a magnetic field, every particles will 'communicates' to each other to a completely linked forms with each of them are trying to reach an energetically favourable state that means a minimum of the energy function. This state is known as activation. Thus, all neurons in this state will rotate and thereby encourage each other to continue the rotation. So, let the network evolves until minimum energy is reached. The minimum energy corresponds to the energy needed to reach global minima values.

b) Test the final state (state obtained after the neurons relaxed). The system will determine the final state as a stable state if the state remains unchanged for more than five runs.

c) Following this, calculate corresponding final energy for the stable state.

d) If the different between the final energy and the global minimum energy is within tolerance value, then consider the solution as global solution or else go to step 1.

8) Finally, calculate global solution and also calculate ratio of global solutions ratio: $Ratio = \frac{Number\ of\ global\ solutions}{Number\ of\ iterations}$, The relaxation will run for 100 trials and 100 combinations of neurons.

9) Lastly, the system produce will the output for each of the run, the steps are repeated until all the desired results are obtained, the output with will include the following:

Minimum Energy, Random clauses, Number of relax, Global Minima Ratio, Local minimum Ratio, Hamming Distance, Computational time and Graphs of Minimal Ratio NC1- NC3 where NC1 is one literal per clause, NC2 is two literals per clauses and NC3 is three literals per clauses.

IX. SIMULATION AND DISCUSSION

1. Global minima ratio:

Due to the similarity in the graphs of NC1, NC2 and NC3 for global minimum ratio and also from the fact that values of NC1, NC2 and NC3 were choosing in such way that NC1= NC2 = NC3. The values (NC1= NC2 = NC3 and NN) were varied such that the ratio NC1/NN give the ratio (0.1, 0.2, 0.3 . . ., 1.0) for all value of NC1= NC2 = NC3 less than 80 and *NN* range from 20 to 80.

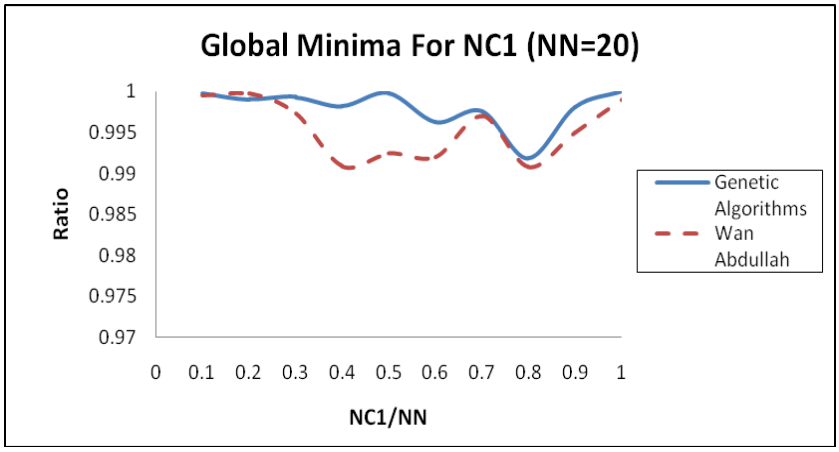

Figure 6: Global minima ratio for NC1=NC2=NC3 and neurons = 20

The results from the figure 6, showed that the ratio of global solutions for GAs method produces better results compared to Wan Abdullah's method. The GA method showed consistency for various value of number of neuron ranging from (20 to 80). The GAs method shows that the global solutions obtained are nearly or 1 for all values of NN, even though the network become more complex by increasing the NN and the NC1, NC2, and NC3, this does not affect the results significantly. However, the results of global minima ratio obtained, for GAs is more stable and consistent even as NN increases.

2. Running time

The following figure (7) show the results of computational time for NC1= NC2 = NC3 with NN = (20, 30,...,80). All computational time were computed in seconds.

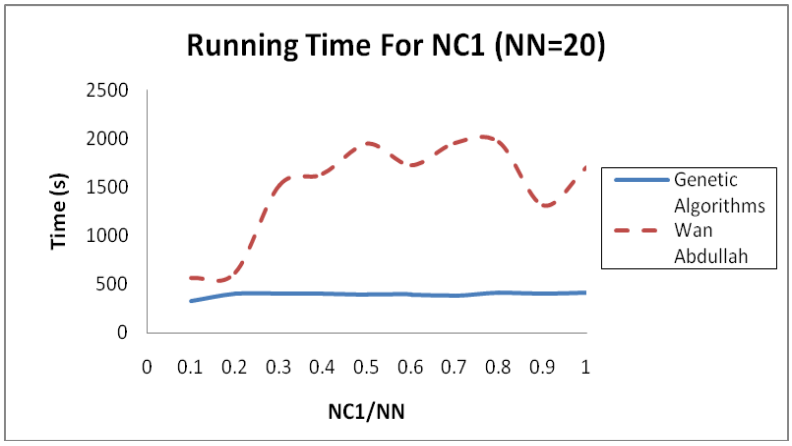

Figure 7: Running time for NC1= NC2= NC3 and neurons = 20

Figure 7 shows the computational time or simulation time for the GAs method and Wan Abdullah's method. For clarification and simplicity, all of the results collected are retracted to maximum of three (3) days of running time. Therefore from figure7, it can be observed that the proposed GAs method is more efficient when dealing with higher ratio of $\frac{number\ of\ literals\ per\ clause\ (NC)}{number\ of\ neuron\ (NN)}$, compared to Wan Abdullah's method. Initially, the difference between the computational times for the two methods seems to very small, the result of Wan Abdullah's method shows a bearable performance as the GAs method. However, as the network become more complex the GAs performance is better than the Wan Abdullah's method whose time for computation got wider than what could be taken as a good one. The behaviour of the two methods thus remains consistent as the network complexity is increased from NN = 20 until NN = 80.

Precisely, it was observed that as NN increase, the running time of Wan Abdullah's method becomes much slower and the collection of results becomes lesser. In addition, the difference in the computational time between GAs method and Wan Abdullah's method turn out to be more visible as the network complexity increase. These results justify consistency and stability of GAs method since they are less susceptible to get stuck at the local minima as compared to Wan Abdullah's method.

3. Hamming distance for genetic algorithms

The below figure (8) shows the results of hamming distance for GAs, the values NC1=NC2= NC3 were chosen to preserve the ratio (0.1,0.2, . ., 1.0) while the value NN = 40 was chosen, there is no any reason for chosen NN=40, it was chosen as such due to similar results obtained for the hamming distance for GAs which is approximately zero to 4 significant values. This situation certify that all the solutions obtained are global solutions, as the distance between the stable states and the global state are approximately zero, in other word, the statistical error are similar in all the cases.

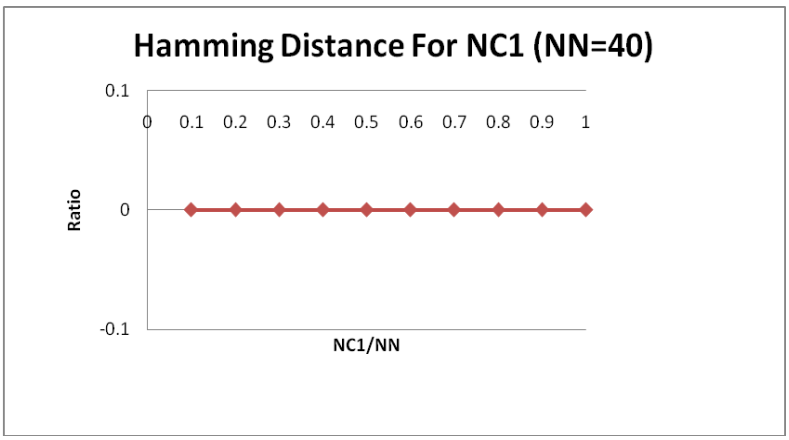

Figure 8 Hamming distance for NC1= NC2= NC3 and neurons = 40

Figure 8 shows hamming distance for proposed GA, whose values are nearly zero. We can deduce that this is due to the fact that GAs hardly get trapped in the local minima or any sub-optimal solutions and always search for optimal solutions which correspond to the global solutions this is an indication of global states (solutions). However, since all solutions obtained are global solution, the distance between the stable states and the attractor are zero. Supporting this, we obtained zero (0) values for the hamming distance as shown in the figure 8.

Furthermore, in GAs, the operators of crossover and mutation can be seen as ways of moving a population around the fitness echelon defined by the fitness function. The population evolves until the optimum fitness is reached. After some number of generations, all of the individuals in the population becomes similar (the fitness variance is low) and the evolution will come to halt.

Therefore, from our analysis, initial population of GAs is responsible for the solutions to converge towards global minima. This is because, at initial generation of GAs, the fitness variance in the population is high, and small numbers of individual (chromosome) are much fitter than the others. This paper ascertained that rate of convergence to the global minima depends on the fitness value of the individuals in the population. Since our GAs method start with random initial population, we determined the appropriate number of the GAs population using trial and error technique.

From the results obtained, we have shown and proved that GAs method is an efficient method in solving optimal problem or finding most appropriate model for a set of data. They efficiently search the model space; therefore they are more likely to converge towards global minima[33].
By using agent based modelling for doing logic programming in Hopfield network computer we verified that we can obviously obtain models for logic programming [17]. This however shows that we can implement genetic algorithm in logic programming in Hopfield network model through a specific procedures.

X. CONCLUSIONS

The research utilized the genetic algorithms (GAs) in finding optimal neuron states, which portrayed an improved method of doing logic programming in Hopfield network as compared to the previous method proposed by Wan Abdullah. The performances of these two methods were compared based on global minima ratio, running time and hamming distance. Results obtained indicate that the GAs method improved the efficiency in finding global solutions. Besides, the computational time using GAs method is better than the Wan Abdullah's method. Furthermore, this poses an indication that GAs method is well adapted to a complex network compared to Wan Abdullah's method, whose effect is obviously seen as the complexity of the network increases simply increases the number of neuron (NN). The results obtained also certified that GAs method always converges to the optimal solution or nearly to the optimal solutions and maintains the population of the candidate solutions for the problem to be solved. In addition to this, GAs is less prone to get trapped in the local optima or in any sub-optimal solutions. In contrast, Wan Abdullah's method exhibits slow convergence to the desired solutions (global solutions) and takes longer computational time as the network gets larger and complex. Thus, from the simulation results obtained, it can be concluded that GAs is a promising method for solving optimization problems and is a useful technique when dealing with a large and complex search space, while the results of the hamming distance mostly zero (0) in genetic algorithms strengthening the assertion of global solutions in the process.

XI. RECOMMENDATION

GAs can be apply to improve the efficiency of doing logic programming in Hopfield network, for instance, grouping genetic algorithm (GGA), ant-colony optimization (ACO), and particle swarm optimization (PSO). Also genetic algorithms produce a promising technique for calibrating ABM parameters, one need to be very careful in the choice of calibration measure. Different parameters provide varying levels of efficiency e.g. fitness functions might varies the results and performance, Thus model analysts should be conscious and try several different calibration measures when attempting formulating fitness function for model for better performance and results.

XII. REFERENCES

1. J.H. Holland, "Adaptation in Natural and Artificial Systems (Cambridge, 1992).
2. A. Imada, , & K. Araki, (1995, July). Genetic Algorithm Enlarges the Capacity of Associative Memory. In *ICGA* (pp. 413-420), 1995.
3. A. Imada, & K. Araki, . Random Perturbations to Hebbian Synapses of Associative Memory Using a Genetic Algorithm, *Proceedings of the 4th International Work-Conference on Artificial Neural Network.***1240**,pp 398-407, 1997.
4. J.J. Hopfield. Neural Networks and Physical System with Emergent Collective Computational abilities. Proc.Natl. Acad. Sci. USA, **79**,pp 2554-2558, 1982.
5. J.J. Hopfield. Neurons with Graded Response Have Collective Computational Properties like Those of Two-State Neurons.Proceeding. Natl. Acad. Sci. USA. 81(10), pp 3088-3092, 1985.
6. J.J. Hopfield. Neurons with Graded Response Have Collective Computational Properties like Those of Two-State Neurons. Proceeding. Natl. Acad. Sci. USA. , 81(10), pp 3088-3092, 1984.
7. R.A. Kowalski. Logic Programming for Problem Solving, New York: Elesvier Science Publishing Co, (1979) .
8. E. G. Ortiz-Garcia, S. Salcedo-Sanz, A. M. Perez-Bellido, & A. Portilla-Figueras, (2007, September). A hybrid hopfield network-genetic algorithm approach for the lights-up puzzle. In *2007 IEEE Congress on Evolutionary Computation* (pp. 1403-1407), 2007.
9. Somesh, Manu Pratap Singh and Kumar. "Pattern recalling analysis of English alphabets using Hopfield model of feedback neural network with evolutionary searching." *International Journal of Business Information Systems* 6.2 :pp200-218, 2010.
10. W.A.T.Wan Abdullah, Logic Programming on a Neural Network. *International Journal of Intelligent System*, **7**,pp 513, 1992.
11. W. A. T. Wan Abdullah, Neural Network logic. In O. Benhar et al. (Eds.),Neural Networks: *From Biology to High Energy Physics*. Pisa: ETS Editrice, pp. 135-142, 1991.
12. W. A. T. Wan Abdullah . Logic Programming in Neural Networks. *Malaysian Journal of Computer Science*, 9(1), pp. 1-5, 1996.
13. W.A.T. Wan Abdullah, Logic Programming on a Neural Network. Malaysian Journal of computer Science, 9 (1), pp. 1-5, 1993.
14. S. Sathasivam. Logic Mining in Neural Networks. PhD. Thesis. University of Malaya, Malaysia , (2006).
15. S. Sathasivam. ,Energy Relaxation for Hopfield Network With the New Learning Rule. *International Conference on Power and Optimazation*, (2009), Bali.

16. S. Sathasivam, & W.A.T. Wan Abdullah. Logic Learning in the Hopfield Networks, *Modern Applied Science,* **2**(2), pp 57-62, 2008.
17. S. Sathasivam, Upgrading Logic Programming in Hopfield Network, Sains Malaysiana, 39, pp. 115-118, 2010.
18. S. Sathasivam, & N. P. Fen. Developing agent based modeling for doing logic programming in hopfield network. *Applied Mathematical Sciences*,*7*(1), pp 23-35, 2013.
19. S.S. Haykin, *Neural Networks and learning machines.* New Jersey: Prentice Hall, 2009.
20. K. DeJong, An analysis of the behavior of a class of genetic adaptive systems. *Ph. D. Thesis, University of Michigan*, 1975.
21. M. Gen, and C. Runwei, *Genetic algorithms and engineering optimization*. pp. 7. John Wiley & Sons, 2000.
22. G. Pinkas, Propositional non-monotonic reasoning and inconsistency in symmetric neural networks, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*. pp525-530, 1991.
23. T.P. Patalia, & G.R. Kulkarni, Behavioral Analysis of Genetic Algorithm for Function Optimization, *Proceedings of the International Conference on Computational Intelligence and Computing Research*. 1. 2010.
24. U. Wilensky, W. Rand, and Kornhauser, D "Visualization tools for agent-based modeling in NetLogo." *Agent2007, Chicago, November* , pp 15-17, (2007).
25. U.Wilensky, Paradox, programming, and learning probability: A case study in a connected mathematics framework. *The Journal of Mathematical Behavior*, 14(2), pp. 253-280, 1995.
26. U. Wilensky, NetLogo Simulation Software northwestern. edu/netlogo Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL, 2008.
27. G.N. Gilbert. Agent-based models (No. 153). Sage, 2008 .
28. M.G. Pires, I. Silva, & F.C. Bertoni, Solving Shortest Path Problem Using Hopfield Networks and Genetic Algorithms, *Proceedings of the Eight International* Conferences *on Hybrid Intelligent Systems*. pp 643-648, 2008.
29. Chughtai (1995). *Determining Economic Equilibria using Genetic Algorithms*. Unpublished Thesis.
30. S. Kumar, & M.P. Singh, Pattern Recall Analysis of the Hopfield Neural Network with a Genetic Algorithm, *Computers and Mathematics with Applications*, **60**, pp 1049-1057, 2010.
31. M. J. Berryman. Review of software platforms for agent based models. Technical Report DSTO-GD-0532, Defence Science and Technology Organisation, Edinburgh, Australia (April, 2008).
32. Tisue, Seth, and Uri Wilensky. "Netlogo: A simple environment for modeling complexity." *International conference on complex systems*. pp 16-21. 2004.
33. M. Mitchell, *An Introduction to Genetic Algorithms.* United States: MIT Press[2001].
34. G. Monslave, *Introduction of Genetic Algorithms for Geophysical Applications.*Term Paper, University of Colorado Boulder(2006).

*****