Applied Mathematical Sciences, Vol. 7, 2013, no. 1, 23 - 35

Developing Agent Based Modelling for Doing Logic Programming in Hopfield Network

Saratha Sathasivam and Ng Pei Fen

School of Mathematical Sciences, Universiti Sains Malaysia
11800 USM Penang, Malaysia
saratha@usm.my, ngpeifen@yahoo.com

Abstract

In recent studies on artificial intelligence, logic program occupies a significant position because of its attractive features. Neural networks are dynamic systems in the learning and training phase of their operation and convergence is an essential feature, so it is necessary for the researchers developing the models and their learning algorithms to find a provable criterion for convergence in a dynamic system. In this paper, an agent based modelling (ABM) was developed by using NETLOGO as a platform to carry out logic programming in Hopfield network. The developed model seems to illustrate the task of doing logic programming in a simpler and user friendly manner.

Keywords: Hopfield Network, NETLOGO, agent based modelling

1.0 INTRODUCTION

Neural Networks is a mathematical model or computational model that is inspired by the structure of biological neurons such as the brain process information. It can solve sophisticated recognition and analysis problems. It is because it composed of huge amount of interconnected neurons to solve specific problems. However in this paper, we are concentrated on Hopfield network. Hopfield network is a recurrent neural network [5] invented by John Hopfield, consists of a set of N interconnected neurons which all neurons are connected to all others in both directions. It has synaptic connection pattern which involving Lyapunov function, E (energy function) for dynamic activities.

Gadi Pinkas [2, 3] and Wan Abdullah [10, 11] defined a bi-directional mapping between propositional logic formulas and energy functions of symmetric neural networks. Both methods are interested in finding whether the solutions obtained are models a corresponding logic program. Both researchers interested with Hopfield network. Besides that, both approaches can handle non-monotonicity of logic. Pinkas introduced "preferred interpretation" in handling non-monotonicity. Meanwhile, Wan Abdullah's method hunts for the best solutions, given the clauses in the logic program, and the corresponding solutions may change as new clauses added. Even when clauses in the logic program are inconsistent, but yet by using Wan Abdullah's method, interpretation with least logical inconsistent can be obtained. Pinkas introduced "preferred interpretation" in handling non-monotonicity. Meanwhile, Wan Abdullah's method hunts for the best solutions, given the clauses in the logic program, and the corresponding solutions may change as new clauses added. Even when clauses in the logic program are inconsistent, but yet by using Wan Abdullah's method, interpretation with least logical inconsistent can be obtained. However the main different between these both approaches are: Pinkas's work revolves around first-order logic meanwhile Wan Abdullah's work revolves around propositional Horn clauses and learning ability of the Hopfield network. Our objective is to find models for the corresponding logic program.

In this paper, we develop agent based modelling (ABM) for doing logic programming in Hopfield network. Agent-based modeling is a powerful simulation modeling technique that has seen a number of applications in the last few years, including applications to real-world business problems. ABM is computer representation of systems that are comprised of multiple, interacting agents. We want to develop a user friendly approach to handle the task of logic programming. We limit our model to Horn clauses due to non-Horn clauses involve more computational complexity and triggered satisfiability problem.

This paper is organized as follows. In section 2, an outline of Hopfield network is given and in section 3, method of doing logic programming in neural network is described. Meanwhile in section 4 contain discussions regarding the NETLOGO and agent based modelling. Finally, section 5 and 6 occupy the simulation results and concluding remarks regarding this work.

2.0 HOPFIELD NEURAL NETWORK

The Hopfield model is a single layer recursive neural network where the output of each neuron is connected to the input of every other neuron. The energy function of the Hopfield network, which is a quadratic function, is associated with the objective function for minimizing the optimization problem.

A Hopfield network with n units has two versions: binary and continuously valued. Let v_i be the state or output of the i th unit. For binary networks, v_i is either +1 or -1, but for continuous networks, v_i can be any value between 0 and 1. Let w_{ij} be the synapse weight on the connections from units i to j. In Hopfield networks, $w_{ij} = w_{ji}$, $\forall i, j$ (symmetric networks), and $w_{ii} = 0$, $\forall i$ (no self-feed-back connections). The network dynamics for the binary Hopfield network are:

$$v_i = \text{sgn}\left(\sum_j w_{ij} v_j - threshold \right) \tag{1}$$

An energy function for discrete Hopfield network is given by:

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} v_i v_j - \sum_j w_i v_i \tag{2}$$

Discrete Hopfield network is shown in Figure 1, as an expanded form of a common representation of the Hopfield network. Hopfield [5] had stated that this network is useful for solving combinatorial optimization problems as a content addressable memory or an analog computer. Combinatorial optimization includes looking for the combination of choices from a discrete set which produces an optimum value for some related cost function.

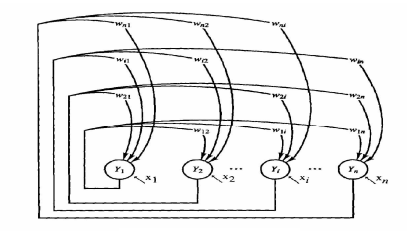


Figure 1: Discrete Hopfield network

As the network evolves according to the dynamics, the energy E can only decrease or stay unchanged at each update. This is because the change ΔE due to a change in the output can only be zero or negative. Eventually the network will converge to a (local) minimum energy state because E is bounded from below. The local minimum points in the energy landscape correspond to the prototype patterns stored in the storage phase. It means Hopfield network retrieval approximately equal to the target patterns.

3.0 LOGIC PROGRAMMING IN HOPFIELD NETWORK

Essentially, logic programming can be seen as a problem in combinatorial optimization and thus it can be carried out on neural network to obtain the desired solution. This is done by using the neurons to store the truth values

of the atoms. When all the clauses are satisfied, we are required to write a cost function which is minimized.

A logic program consists of a set of program clauses (at least one conclusion) and is activated by an initial goal statement, in Conjunctive Normal Form (CNF), the clauses contain one positive literal. Logic programming in Hopfield model can be formulated as a problem in combinatorial optimization. Due to that, it can be carried out in a neural network to obtain the desired solution. Our main objective is to find a set of interpretation (i.e., truth values for the atoms in the clauses which satisfy the clauses (which yields all the clauses true). In other words, we want to find 'models' for the corresponding logical clauses [8, 9].

The following algorithm summarizes on how a logic program can be done in a Hopfield network based on proposal by Wan Abdullah [10] known as *direct method*:

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, *H*.
vi) Let the neural networks evolve until minimum energy is reached. Checked whether the solution obtained is a global solution.

4.0 AGENT BASED MODELLING

After knowing the efficiency of doing logic programming in Hopfield network, we will used NETLOGO [1, 4] 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 is a multi-agent programming language and integrated modelling environment. There has been continuous development for connected learning and computer-based modelling. Furthermore, it is a well suited

method for modelling complex systems that can give instructions to hundreds or thousands of "agents" to operate independently like turtle. It is because it is fully programmable and formed by simple language structure. It can instruct mobile agents move over a grid of stationary agents.

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. 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.

4.1 Model of Neural Networks Simulator

Firstly, a simulator of Hopfield networks that using a conventional computer had created instead of every time build up a new network design or store a new set of memories. It saves lots of energies and times for the programmer to rebuild new system from time to time. Thus, a computer program which emulates exactly what the user want needs to construct in order to simulate the action of Hopfield Network. It wills easy the programmer to modify the program and store a new set of data. Thus, an agent based modelling had designed for the user to run the simulator.

Moreover, agent-based Modelling (ABM) [6, 7] which also called individual-based modelling is a new computational modelling paradigm which is an analyzing systems that representing the 'agents' that involving and simulating of their interactions. Their attributes and behaviours will be group together through their interactions to become a scale. Programmer can design ABM in NETLOGO by using button, input, output, slides and other functions that make ABM easy to understand and use. In addition, ABM reveals the appearance of the systems from low to high level outcomes and it make improvement by surpassing the traditional modelling limitations such as allowing agent learning and adaption, limited knowledge and access to information. It is because, the agent based modelling paradigm are commonly used in dynamics and complex communities such as telecommunication, health care and others that involving large populations which used explicitly capture social networks.

In general, when we build an ABM to simulate a certain phenomenon, we need to identify the actors first (the agents). We then need to consider the

processes (rules) governing the interactions among the agents. Figure 2 shows the layout of the ABM built and Figure 3 shows the flow chart for the ABM.

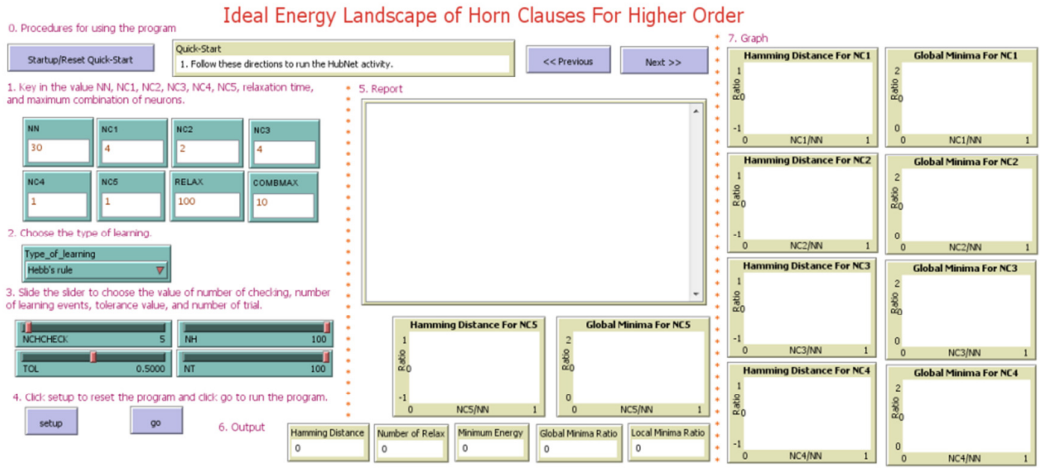
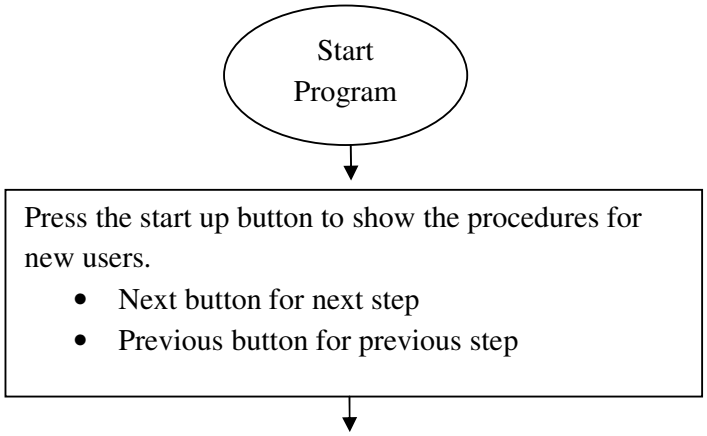


Figure 2: The layout of the ABM



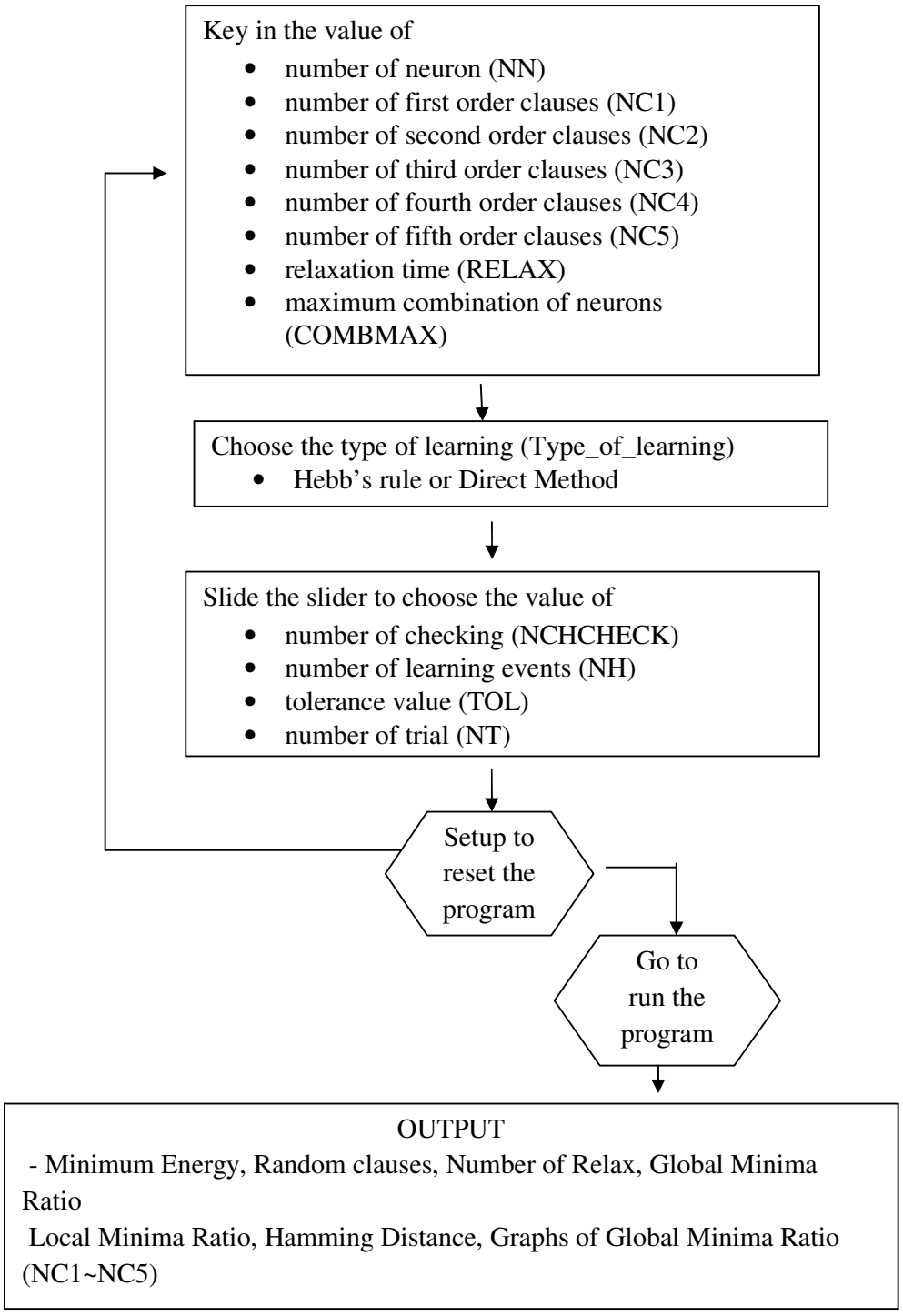


Figure 3: Flow chart of the ABM

Explanation for the flow chart:

PHASE 1: Entering 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) Later, key in NN, NC1, NC2, NC3, NC4, NC5, RELAX and COMBMAX.
a) The maximum of NN is 80. NC1, NC2, NC3, NC4, NC5 is related to the value of the NN that had entered. The maximum number of RELAX and COMBMAX is 1000. 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 [10, 11].
4) Then, slide the slider 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 value had been set, press the setup button to fix and set those values in the program.
6) Then, press go button to run the program.
a) The program will generate random program clauses. Example, if user declared NC1 as 3, then 3 first order clauses will be generated.

PHASE 2: Training

7) Initialize initial states for the neurons in the clauses
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= Number of global solutions/ Number of iterations). The relaxation will run for 1000 trials and 100 combinations of neurons
9) Lastly, the system will print out the output for each run.

5.0 EXPERIMENTAL RESULTS AND DISCUSSION

From the graphs obtained, we observed that the ratio of global solutions is consistently 1 for all the cases, although we increased the network complexity by increasing the number of neurons (NN) and number of literals per clause (NC1, NC2, NC3). Due to we are getting similar results for all the trials, to avoid graphs overlapping, we only presented the result obtained for the number of neurons (NN) = 40. Besides that, error bar for some of the cases could not be plotted because the size of the point is bigger than the error bar. This indicates that the statistical error for the corresponding point is so small. So, we couldn't plot the error bar.

Most of the neurons which are not involved in the clauses generated will be in the global states. The random generated program clause relaxed to the final states, which seem also to be stable states, in less than five runs. Furthermore, the network never gets stuck in any suboptimal solutions. This indicates good solutions (global states) can be found in linear time or less with less complexity.

Since all the solutions we obtained are global solution, so the distance between the stable states and the attractors are zero. Supporting this, we obtained zero values for Hamming distance. Figure 4- Figure 9 illustrate the graphs obtained for ratio of global solutions and final hamming distances. From Figure 4-Figure 6, we can observed that the error bar increased simultaneously with the ratio of number of clauses to number of neurons. This is because, when the number of clauses increased, the probability for the same neuron involved in more than one clause also increased. So, the statistical error also increased as shown in the figures.

Meanwhile, from Figure 7-Figure 9, we can observed that the error bars for Hamming distances are almost similar. This is because, in all the cases, the stable states we obtained are global solutions. So, the distance between the stable states and global states are almost zero. Due to this, we obtained similar statistical error for all the cases.

By using agent based modelling for doing logic programming in Hopfield network computer we verified we can obtains models for the logic program.

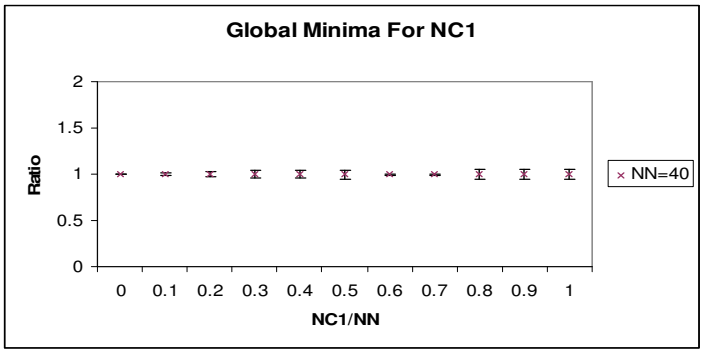


Figure 4: Global Minima Ratio for NC1

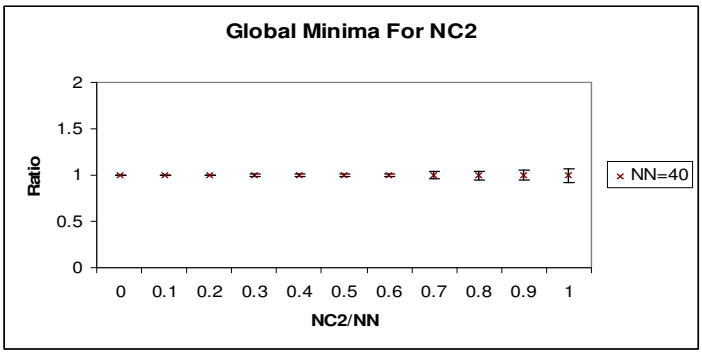


Figure 5: Global Minima Ratio for NC2

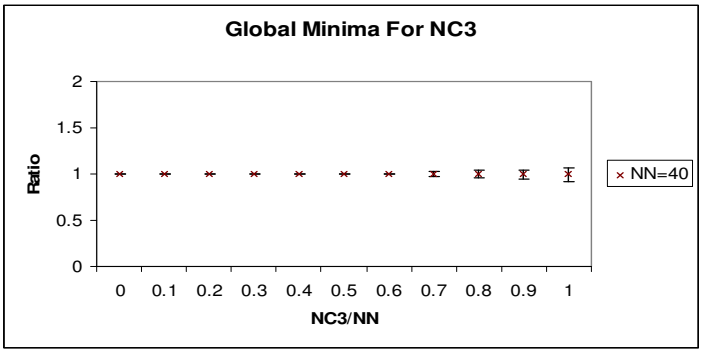


Figure 6: Global Minima Ratio for NC3

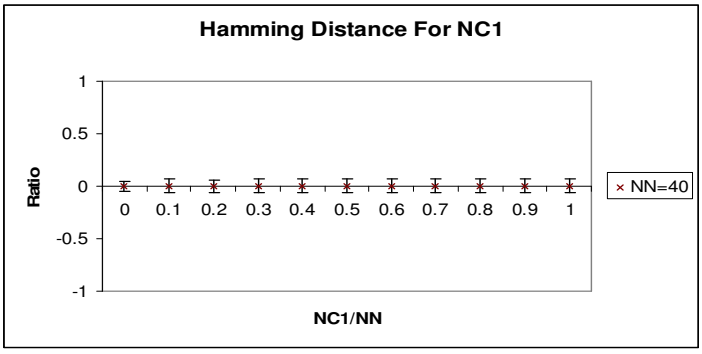


Figure 7: Hamming Distance for NC1

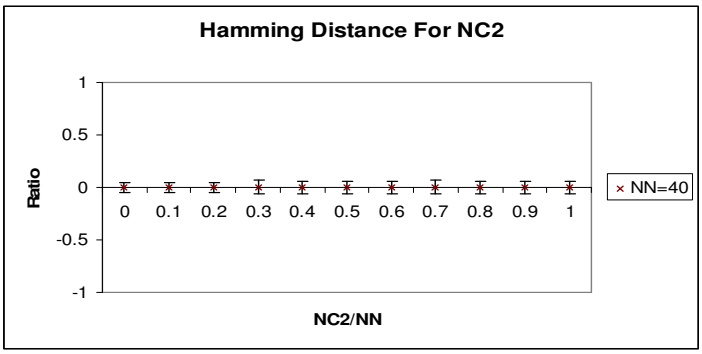


Figure 8: Hamming Distance for NC2

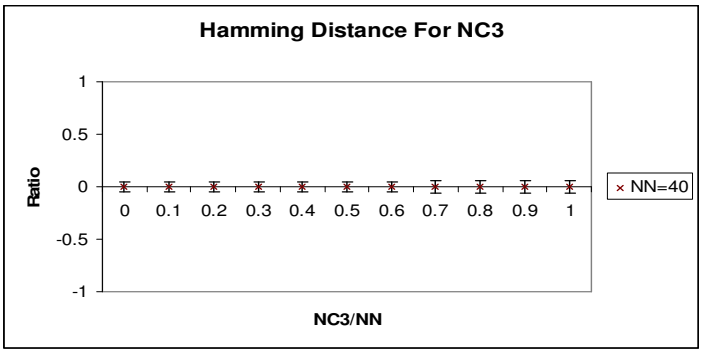


Figure 9: Hamming Distance for NC3

6.0 CONCLUSION

In this paper, we had developed agent based modelling to carry out logic programming in Hopfield network by using NETLOGO as platform. The agent based modelling that was developed was user friendly. The benefits of ABM over other modelling techniques can be captured in three statements:

(i) ABM captures emergent phenomena: able to produce model for the set of logic programs

(ii) ABM provides a natural description of a system: able to integrate/link logic program and Hopfield network

(iii) ABM if flexible: we can change the training parameters according to the user

As a conclusion, the ABM develop model the process of doing logic programming in Hopfield network efficiently.

ACKNOWLEDGEMENT

This research is partly financed by Universiti Sains Malaysia's short term grant (304/ PMATHS/6312053) and E-Science grant (305/PMATHS/613142) from the Ministry of Higher Education, Malaysia

REFERENCES

[1] E. Bonabeau, Application of Simulation to Social Sciences, eds. Ballot, G. & G. Weisbuch, (Hermès Sciences, Paris), (2000), pp. 451–461.

[2] G. Pinkas, Energy minimization and the satisfiability of propositional calculus, Neural Computation, **3**, (1991), pp 282-291.

[3] G. Pinkas, Propositional nonmonotonic reasoning and inconsistency in symmetric neural networks, Proceedings of the 12th International Joint Conference on Artificial Intelligence, (1991), pp 525-530.

[4] http://ccl.northwestern.edu/netlogo/docs/. Accessed on 04/06/2012.

[5] J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proceedings National Academy of Science USA, Vol. 79, (1982), pp. 2554-2558.

[6] J.O. Kephart, J. E. Hanson, & A. R. Greenwald, Computer Networks, 32, (2000), pp731-752.

[7] M.J. Berryman & S.D. Angus, Software Tools for Analysis and Modelling of Complex Systems, In R. L. Dewar & F. Detering (Eds.) Complex Physical, Biophysical and Econophysical Systems: Proceedings of the 22nd Canberra International Physics Summer School. Sydney: (2010), World Scientific Publishing.

[8] Saratha Sathasivam & W.A.T. Wan Abdullah, Logic Mining in Neural Network, Computing, Volume 91, Issue, (2011), pp 119-133.

[9] Saratha Sathasivam, Upgrading Logic Programming in Hopfield Network, Sains Malaysiana, 39(1), (2010), pp115-118.

[10] W.A.T. Wan Abdullah, Logic Programming on a Neural Network, Int.J. Intelligent Sys, **7**, (1992), pp 513-519.

[11] W.A.T. Wan Abdullah, Neural Network logic, Proc of the workshop held in Elba International Physics Center, Italy, (1991), pp 135-142.

Received: September, 2012