December 2002

A Peer-to-Peer Associative Memory Network for Intelligent Information Systems

Asad Khan
Monash University

Vinod Ramachandran
Monash University

A Peer-to-Peer Associative Memory Network for Intelligent Information Systems

Asad I. Khan

School of Network Computing
Monash University
Melbourne, Australia
Asad.Khan@infotech.monash.edu.au

Abstract

The paper describes a highly-scalable associative memory network capable of handling multiple streams of input, which are processed and matched with the historical data (available within the network). The essence of the associative memory algorithm lies with in its highly parallel structure, which changes the emphasis from the high speed CPU based processing to network processing; capable of utilising a large number of low performance processors in a fully connected configuration. The approach is expected to facilitate the development of information systems capable of correlating multi-dimensional data inputs into human thought like constructs and thus exhibiting a level of self-awareness.

Keywords

Associative Memory, Content-addressable Memory, Parallel Processing Systems, Intelligent Agents, Spiking Neurons, Neural Networks, Artificial Intelligence, Nano-sensors, Quantum Computing

INTRODUCTION

A truly scalable *associative memory* would be an essential component in the design of software agents with human-like intelligence. The implementation of an *associative memory* system should be such that it may store information from a variety of sources. In the case of intelligent biological systems the inputs are in the form of somatic stimuli e.g. the sense of sight, hearing, smell, touch, and taste. These stimuli may be further classified into spatial and temporal based inputs. When designing an artificially intelligent system we have an opportunity to add to the inputs provided by nature and thus create systems with a higher level of awareness.

The earliest implementation of an *associative memory* system may be traced back to the Hopfield network. The network was conceptualised in terms of its energy and the physics of dynamic systems. Primary applications for this sort of network have included associative, or content-addressable, memories and a range of optimisation problems.

Improvements to the Hopfield model were investigated through the parallelisation of the code; albeit for the computationally intensive optimisation problems e.g. Di Blas *et al.* (2000). More recent work done on the spatiotemporal encoded, or spiking, neurons by Hopfield and Brody (1998) has drawn upon the properties of loosely-coupled oscillators. Izhikevich (1999) states though some new capabilities at differentiating between similar inputs have been revealed however there is no clear evidence to suggest their superiority over the classical Hopfield model in terms of an overall increase in the associative memory capacity. The Back-Propagation network provides a scalable associative memory however it is limited by the excessive computational cost required for adding new patterns. Also, the energy and error minimisation functions often get trapped inside the local minima for these networks. It is possible to devise an *associative memory* system that works with exact matches or utilises a nearest neighbour approach. However the computation cost would generally tend to increase non-linearly with the increase in the number of stored patterns for such implementations. Hence, most of the effort made into the emulation of some of the very basic biological memory functions has yet to produce comparable performances within the silicon-based systems. Either the pattern storage capacity does not scale-up too well or the computational cost becomes prohibitive.

Stapp (1995), while discussing the quantum-mechanical aspects of the consciousness, presented a highly-parallel model for processing information. The mechanics of vision discussed by Huth (2002) and the work done on the modelling of consciousness by Baars (1997) and Franklin (2001) also depict a highly parallel architecture for processing information.

Hence the objective of this research is to introduce the type of parallelism, which is present within the biological systems, for implementing a generic *associative memory* system.

VARIOUS ASSOCIATED MEMORY MODELLING TECHNIQUES

The Statistical Modelling Approach

The Ordination methods such as Principal Component Analysis (PCA) and Correspondence Analysis (CA), within the Multivariate Techniques, provide a mechanism for implementing an *associative memory* (AM) system. These methods rely on the correlation available within the data patterns to assist with the classification and the subsequent retrieval of these patterns. PCA requires an explicit and *a priori* knowledge about the correlated variables where as CA makes use of non-intuitive techniques such as the Eigen value analysis to determine the correlations. The issue with PCA is in its dependence on human judgement; to assist with the classification process. In the case of CA, the Eigen value computations become very costly for larger data sets.

The Neural Network Approach

The classical neural networks such as Hopfield and Back-propagation depend upon an energy or an error minimisation function. Generally the accuracy of recall tends to fall with the increase in the number of patterns. Also, these networks are sensitive to the number of input and output variables. More recent work with the temporal-spatial encoded (spiking) neurons has shown a remarkable increase in the pattern recognition capabilities of these networks. The significant improvement is in the ability of an individual neuron to respond to complex stimuli. This allows for a much finer grained evaluation, of the input pattern, as compared with the classical networks. Van Rullen *et al*. (1998; 1999) and Thorpe (2000) have shown how these neurons may be used to implement a face recognition algorithm that follows the same principles as the human vision. However there still remains a question mark regarding their ability to scale-up to large storage capacities and to deal with generic data inputs.

Some of the Other Approaches

Wagner and Stucki (2001 in press) have presented a novel approach; using the periodic unstable orbits of a chaotic attractor for storing content-addressable information. Watta *et al*. (1999) proposed the use of a Hamming network to increase the memory storage capacity.

THE GRAPH NEURON APPROACH

In this paper an AM network, named as the Graph Neuron (GN), is being presented. This approach models the parallelism available within the naturally occurring AM systems and thus bypasses the deficiencies present in some of the other contemporary approaches. It may also be noted that the GN algorithm is an inclusive technology; it may be extended to include spatiotemporal encoded neurons and the evolutionary optimisation techniques such as the genetic algorithms.

The Graph Neuron (Gn) Rationale

The saying that network is the computer has a distinct meaning when it comes down to modelling an AM system using the GN approach.

There are ample instances in nature where network architectures are employed within the evolved systems. Mattick (2001) asserts that the abundance of the un-coded RNA, within the genetic material, provides the network communication support for the coded DNA. Hence it is reasonable to assume that the man-made (silicon-based) networks should be able to provide a similar level of awareness as the natural networks do. In order to

understand why even the largest man-made network provides no such functionality would however require a comparison between the two types of networks. The cerebral cortex within an average human brain comprises 10 billion neurons according to Shepherd. Koch puts that to 20 billion neurons (Chudler, 2002). Some of the statistics for the human brain versus the largest man-made network, i.e. the Internet, have been compared in Table 1.

	Neural Data	**Internet Statistics**
Cerebral cortex neurons/ nodes	10-20 billion neurons	544 million online nodes
Number of synapses/ connections for a typical node	1,000-10,000	1
EEG frequency range/ typical connection speed	0.5 – 30 Hz	3,500 Hz/ 56 Kbps
Conduction velocity of action potential/ signal speed	0.6-120 m/s	A fraction of the speed of light
Processing capability of a neuron/ CPU	Simple and very low speed biological switching	A typical value of 500 MHz may be assumed for the contemporary Internet nodes

Table 1: A comparison of the human brain data with some of the equivalent Internet statistics

The following inferences may be made from the Table. The neural data comprises a very large number of very low performance processors. These processors are inter-connected with a very large number of direct (point-to-point) links. Each of these links supports a very low network bandwidth.

The Internet has fewer processing nodes but each of these nodes has a far superior processing capability, the network links are much faster but these connect a far fewer number of nodes directly, and the connections are of much higher bandwidth.

It is evident, from the comparison, that the man-made information processing network is heavy on the processing side and light on the network connectivity. Hence the network supports a substantially lower quantum of parallelism, owing to a fewer number of processing nodes, as compared to the human brain. The comparison again highlights the highly parallel and connected aspect of the naturally occurring networks. The above is based upon a cursory examination of the physical topology of the Internet; Hibbard (2001) discusses the impact of the network diameter on the self-awareness in a greater detail by taking into consideration the virtual topology of the web.

THE IMPLEMENTATION OF A GN ARRAY

The AM is implemented as a virtual network of processing nodes, where each node executes the same GN algorithm and thus provides a structure to support parallelism. The algorithm is best suited for immensely parallel systems such as the futuristic quantum computers. However the array has been implemented on a classical computer and hence some underlying assumptions have to be made in order to differentiate between the true capabilities of the array and the limitation imposed by the contemporary computer architectures and networks:

1. The current implementation sets each GN node as a Java object where each of these objects executes the same code, but gets instantiated with a different data set and port numbers. The objects simulate a SIMD processor array. It is thus assumed that a very large number of low-performance processors are available within the implementation.
2. The GN objects communicate with the outside world, and amongst themselves, using the standard network sockets and ports. Hence a GN object may contact any other GN object, located within the local computer's memory or anywhere on the network, directly by utilising the appropriate network and port addresses. It is therefore assumed that each of the objects within the array is capable of directly accessing any other object within the array.

3. The current implementation does not support parallel inputs yet. The outputs from the array are visually presented to the user and hence provide a basic form of parallelism. It is however assumed that the array functions with parallel inputs and outputs; where by each of the GN may be simultaneously accessed for input and output to/ from the array.

The overall topology of the array, which takes into consideration the above assumptions, is shown in Figure 1.

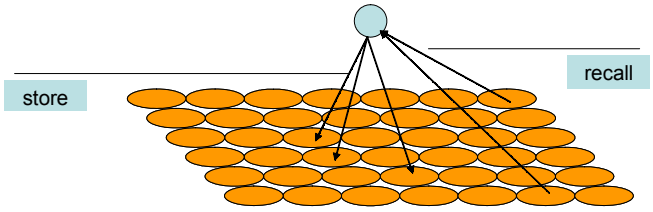


Figure 1: A GN array with parallel store (memorisation) and recall

The input to the array is done sequentially within the actual implementation; the array architecture however is perfectly suited for massively parallel input and output operations.

The proposed architecture draws upon the quantum-mechanical model proposed by Stapp (1995), and later expanded upon in a private communication. Stapp starts with an example of a classical-mechanical system and describes it as follows:

> "3.2 We introduced a grid of points in the brain. Let these points be represented by a set of vectors:
>
> x~i~,
>
> where i ranges over the integers from 1 to N. At each point x~i~ there was a set of fields:
>
> F~j~(x~i~),
>
> where j ranges from 1 to M, and M is relatively small, say ten. For each of the allowed values of the pair (i,j) the quantity F~j~ (x~i~) will have (at each fixed time) some value taken from the set of integers that range from -L to +L, where L is a very large number. There is also a grid of temporal values t~n~, with n ranging from 1 to T.
>
> 3.3 The description of the classical system at any time t~n~ is given, therefore, by specifying for each pair of value (i,j) with i in the set {1,2,...,N} and j in the set {1,2,..., M} some value of F~j~ (x~i~) in the set {-L, ..., +L}. We would consequently need, in order to specify this classical system at one time t~n~, N x M "registers", each of which is able to hold an integer in the range {-L, ..., +L}."

The GN data representation follows a very similar model to the one proposed by Stapp (*ibid*). The implementation of the GN algorithm further demonstrates that a generic thought/ concept may be discretely stored within the network by simply manipulating the adjacency information held within each node of the array.

THE GN DATA REPRESENTATION

The information presented to a GN is in the form of a value, position pair; representing a data point in a two dimensional space (for multi-dimensional patterns the number of values per position would increase in order to represent the additional information – the underlying principle would however remain the same).

The GN array converts the spatial/ temporal patterns into a graph representation and then compares the elements of the graphs for memorisation and recall operations. The advantage of having a graph-like representation is that it provides a mechanism for placing

the spatial/ temporal information in a context. Hence not only can we compare the individual data points but we may also compare the order in which these occur. The drawback to this approach is in the excessive number of comparisons required for matching a stored pattern with an incoming sequence – the search domain increases with the increase in the stored patterns. However this impediment only exists because of the nature of the contemporary computer architecture; which converts purely parallel operations into a sequential form and then emulates these operations in a pseudo-parallel mode using elaborate scheduling algorithms. The proposed algorithm on the other hand utilises the parallelism present within a processor array. The inter-processor message-passing is implementing using the communicating sequential process (CSP) model put forth by Hoare (1985).

Hence the data representation for a GN may be summarised as follows:

An input pattern vector *P{}* is represented as a set of *p*(*value, position)* pairs. These inputs are mapped on to a virtual array of processors by using the adjacency characteristic of the input. For example, alphabets and numbers would have their inherent adjacency characteristics. Similarly images would have the frequency bands, intensity, and spatial coordinates as the adjacency characteristics per pixel etc.

For an input domain *R,* the GN array represents all possible combinations of *P{}* in *R.* Hence each GN node is initialised with a distinct pair *p* from the input domain *R..*

Each GN keeps a record of the number of times it encounters a matching input pair; within its *bias* vector*.* Each element of the *bias{}* comprises a list of the adjacent GNs relating to a matched input pair*.* The *bias{}* counter is incremented for each new pair matched by the GN. A new pair is defined as the one which has a different set of adjacent GNs to the existing elements of the *bias{}.*

In order for this method to work successfully we need to have a priori knowledge regarding the size of the input data domain. Or alternatively we may chose our own limits and define the reality within those bounds. For instance, by defining an input domain which comprises all the characters in a natural language and the number of characters in the longest word occurring in that language would be sufficient to represent any word from the language. Alternatively we could set our own limits for discretising a continuous input domain for this purpose.

THE PARALLELISM WITHIN THE REPRESENTATION

A Graph Neuron (GN) array may be created where each GN is initialised to a value, position pair *p* for every possible position and value within the input domain. The incoming data pairs simply get mapped to their appropriate locations within the array. For instance a four lettered word with a choice of two alphabets, say X and O for each position, would require eight GNs for representing every conceivable combination. It's easy to show that the total number of possible combinations in this case would be 2*2*2*2 = 2^4 = 16.

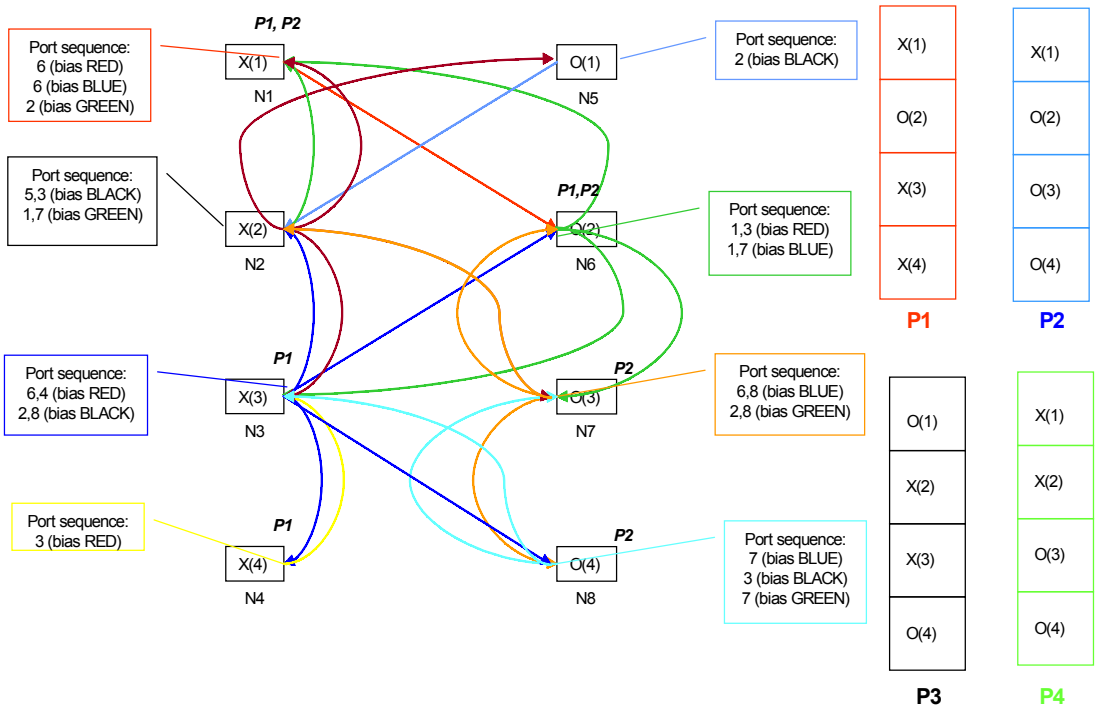
This effectively means that we are assigning a separate search domain for each set of the possible values of the alphabets and thus halving the search domain in this case. E.g. if we get a letter X in the first position of the word then letter O can never occur at this position for this particular word and vice versa. The halved (adjacency) search domains are processed concurrently, thus the total time for the search is that for one half of the domain in this case.

The number of elements in the *bias{}* increase with the number of patterns being presented to the array. However the number of *bias{}* elements does not increase in proportion to the number of stored patterns since pairs with the same set of adjacent GNs are treated as recalls (and thus do not get stored). The store operation requires an increment in the *bias{}* index counter.

The process of searching through the bias entries within each GNs takes place concurrently. This map and search process is broadly illustrated in Figure 2.

The Figure outlines the process of storing patterns *P1, P2, P3, and P4* on an array comprising 8 GNs (labelled as *N1, N2, ... N8)*. Each pattern comprises 4 pairs where the *values* may alternate between *X* and *O* for each of the four *positions*.

Assuming *P1* is mapped first in this instance. Each GN would records the responses from the other GNs to form its port sequence list of the adjacent GNs and would allocate an entry within the *bias{}* for these pairs (the GNs are adjacent if their *position* differs by 1 in this example). Hence, *N1* will store the port number, *6,* in its *bias{}* for *N6* after encountering *p(X,1). N6* will store the port numbers *1 and 3,* for *N1* and *N3,* in its *bias{}* after encountering *p(O, 2).* The process gets repeated for the encounters with the remaining pairs in the pattern i.e. *p(X, 3),* and *p(X, 4)* . The entire process is repeated each time for storing *P2, P3*, and *P4*. The *bias{}* entries for each of the GN are shown in Figure 2. The GN algorithm may thus be summarised as follows.



Note: The colouring scheme for interconnects is separate from the scheme used for the patterns

Figure 2: An eight node GN array is in the process of storing patterns P1 (RED), P2 (BLUE), P3 (BLACK), and P4 (GREEN)

THE GRAPH NEURON ALGORITHM

All GNs have exactly the same logic and code. These are implemented as copies of a self contained message-passing application. Each instance of the application is initialised to a distinct *p(val, pos)* and port values. Hence the GN array keeps all possible values and all possible positions, for a particular data domain *R,* mapped as unique *p(val, pos)* pairs on each GN.

The patterns are presented as sets of *p(val, pos)* pairs to the array. Adjacencies are calculated independently by each *GN* within the array as part of the store/ recall operations.

A GN on receipt of a *p*(*val, pos)* pair checks with all other G*Ns* for adjacent values and notes the port sequence for that particular pair. The GN then compares the previously stored port sequences within the *bias{}* and returns a high bias if a match is found, otherwise the sequence is added as a new element to the *bias*{} (partial matches may result in low confidence bias matches, however this function has not been implemented yet).

Only a single *value* may be found at a particular *position* within the array. Thus knowing the adjacent GN's number is sufficient to determine the pair it has been programmed to respond to.

The Input Operation

Incoming stimuli (the whole pattern/ sequence) should be sensed by every GN (akin to an Ethernet broadcast on a shared LAN). Only the GNs with matching values should initiate action. Doing this would however require interfacing the array to the spiking neurons or a form of sensory mechanism. Alternatively the use of a multicasting protocol may be considered.

Pattern Store and Recall Operations

Assuming such an input mechanism is in place, each GN listens on the port that matches its own unique identity number to store or recall. There is no order as to how a pattern gets distributed amongst the GNs. The commit to memory operation is done on first-come-first-served basis. Each GN communicates with the other GNs to identify its adjacent GNs. The commit to memory operation is only performed if there is no recall within the GN. Hence for each input pair, a GN checks with its neighbours to decided whether to treat the incoming pair as a store or to as a recall operation.

The Graph Neuron PDU

An input to the array is in the form of Protocol Data Units (PDUs) comprising the pattern. Each pattern in-turn comprises a set of value and position pairs. The structure of a PDU is shown in Figure 3, where 'pos' could be a timing relationship or it could be a vector in its own right comprising contextual values associated with each 'val'.

In the current implementation the 'vals' and 'pos' pair determines the contact port and the direction of search; using the adjacency characteristics of a two-dimensional array. It is important to note that the data type is only for the human consumption. As far as the array is concerned, the data type has no bearing on its store and recall operations. The array only deals with the internal representations, associated with the inputs, in terms of its connectivity with other the nodes within the array. The connectivity information is kept within the *bias{}* vector.

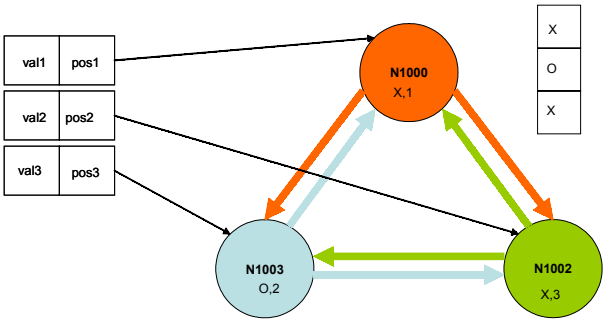


Figure 3: A text string comprising characters 'X' and 'O' being mapped to the appropriate nodes within the array using the input PDUs made up of 'val' and 'pos' pairs

Extending the Array for Storing Concepts

The GN array in its simplest forms provides a Yes or a No answer to the question posed to the array in the form of an input pattern. If the answer is No then the array will memorise this pattern for a future reference; otherwise a Yes answer will be returned. However more meaningful responses may be obtained by simply connecting the arrays in a recursive manner. The inputs, which are at the lowest level of correlation, progressively get correlated as the information is passed to the higher-level arrays. The GN algorithm preserves the path history whilst it concentrates the information through a process of conceptualisation. This is where the algorithm differs from the statistical and the traditional neural network approaches. These approaches tend to lose the path history while reducing the dimensionality of the input information. The GN arrays however maintain the complete path history of the transitions. The recurrent processing of inputs is shown in Figure 4.

The first array is initialised by the user to respond to the input pairs. The outputs from this array are fed into the second array. This process leads to a collapse in the dimensionality of the input data. Hence a set of such arrays could store all the words, sentences, paragraphs, and pages of a book as a single *p (val, pos)* pair within a top level GN – a thought-like construct? As stated earlier, the nature of data is not important. The arrays may store textual information or information collected from a myriad of sources; using a similar mechanism.

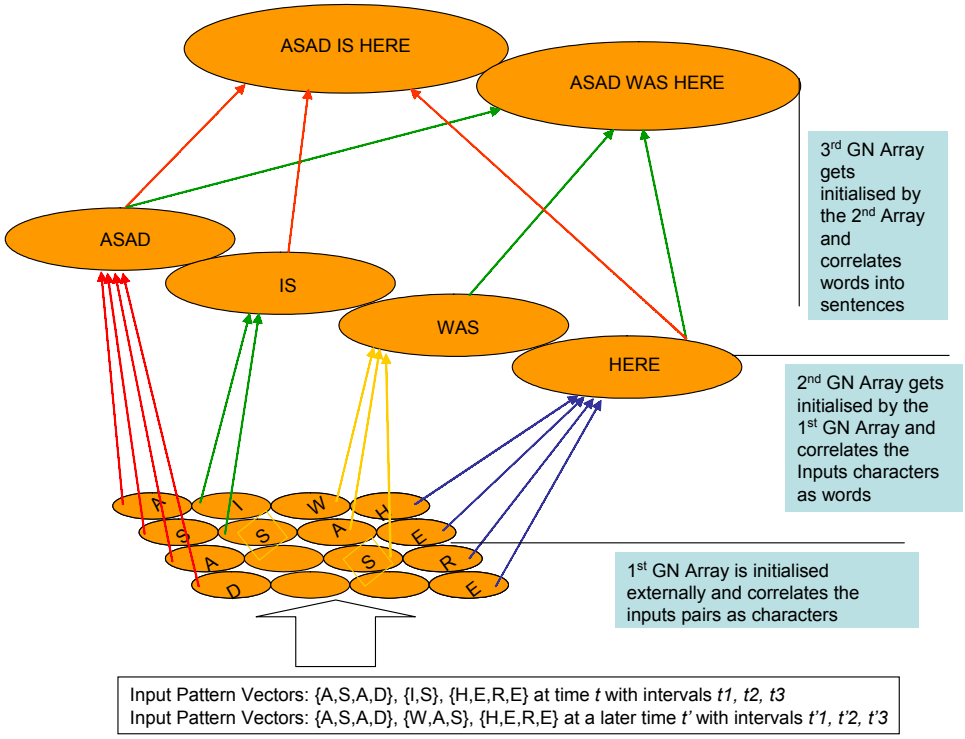


Figure 4: A set of GN arrays linked to store temporal-encoded inputs as discreet concepts

REMARKS AND CONCLUSIONS

The initial mapping of the pairs to the GNs requires that each GN must contact every other GN, at the appropriate locations, within the array. The process requires a high degree of connectivity between the GNs or the availability of adequate bandwidth if shared connections are being used.

The communications among the GNs, with adjacent pairs, take place in parallel. The communication in the previous step would also occur in parallel if a spiking neuron interface or a multicast protocol was used to input the information.

Each new pattern sequence results in an increase in storage within some of the GNs. Thus the search domain does not increase proportionately whilst the total memory capacity keep increasing until it's exhausted for the particular topology.

There are no constraints or overheads for over estimating the size of the array; in this case not all the GNs would get utilised.

The current implementation of the GN array assimilates newer patterns and continues to become more knowledgeable over time.

The later part where several arrays are combined together, in a hierarchical manner to store concepts, is yet to be implemented in software by the author.

SIGNIFICANCE

The proposed AM system is expected to assist with the development of intelligent information systems capable of handling complex concepts and being able to act autonomously. The examples of such systems may include:

- Autonomous software agents that interact amongst themselves to complete the tasks assigned by their owners.
- The visualisation of very large structures or impossible to view perspectives. For instance simultaneously viewing a hollow object from inside and outside or creating a near 360-degrees spherical vision.
- Conceptualisation of multi-dimensional inputs e.g. where the electromagnetic and the sonar data sources are combined to form the higher dimensional inputs.
- Interpretation of information gathered through a multitude of nano-sensors.

The above are some of the outcomes this technology may produce in a bid to create systems that match or even exceed the human perception. More significant but not fully realisable at this stage is its potential in relation to the nano-technologies; a very large number of low performance nano-sensors may be deployed in unprecedented ways to collect and conceptualise information.

REFERENCES

Baars, B. J. (1997) In the Theatre of Consciousness – Global Workspace Theory, A rigorous scientific theory of consciousness, *Journal of Consciousness Studies*, 4, No. 4, 292-309.

Chudler, E. H. (2002) Brain Facts and Figures, http://faculty.washington.edu/chudler/facts.html, Accessed 05-May-2002.

Di Blas, A., Jagota, A., Hughey, R. (2000) Parallel Implementations Of Optimizing Neural Networks, *Proc. of ANNIE 2000 Conf.*, 153-158.

Franklin, S. (2001) Conscious Software: A Computational View of Mind, Soft Computing Agents: New Trends for Designing Autonomous Systems, ed. V. Loia, and S. Sessa. Berlin: Springer (Physica-Verlag), 1 - 46. http://www.msci.memphis.edu/~franklin/bbs_target_2.html, Accessed 05-May-2002.

Hibbard, B. (2001) Network Diameter and Emotional Values in the Global Brain*,* From Intelligent Networks to the Global Brain Evolutionary Social Organization through Knowledge Technology, The First Global Brain Workshop (GBrain 0), 3-5 July 2001, Brussels, Belgium. http://www.ssec.wisc.edu/~billh/gbrain0.html, Accessed 05-May-2002.

Hoare, C. A. R. (1985) Communicating Sequential Processes, Prentice Hall, London, UK.

Hopfield, J. J., Brody, C. D. (1998) What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration*, Proc. Natl. Acad. Sci. USA,* 98**,** 1282-1287. http://www.cshl.org/labs/brody/Papers/Brody/hopfield_brody2.pdf, Accessed 05-May-2002.

Huth, G. C. (2002) A New Model For Light Interaction With The Retina Of The Human Eye And The Vision Process, http://ghuth.com/A%20new%20Model.htm, Accessed 05-May-2002.

Izhikevich, E. M. (1999) Weakly Pulse-Coupled Oscillators, FM Interactions, Synchronization, and Oscillatory Associative Memory*,* http://math.la.asu.edu/~eugene/publications/html/pco/pco/pco.html, Accessed 5-May-2002.

Mattick, J. S. (2001) Molecular genetic networks and the architecture of biological complexity, HPC Asia 2001, 25-28 September 2001, Gold Coast, Australia. http://www.gu.edu.au/conference/hpcasia2001/content4.html#keynote, Accessed 05-May-2002.

Stapp, H. P. (1995) Why Classical Mechanics Cannot Naturally Accommodate Consciousness but Quantum Mechanics Can, *PSYCHE, 2(5),* May 1995. http://psyche.cs.monash.edu.au/v2/psyche-2-05-stapp.html, Accessed 05-May-2002.

Thorpe, S. J., Delorme, A., VanRullen, R., Paquier, W. (2000) Reverse engineering of the visual system using networks of spiking neurons. *Proceedings of the IEEE 2000 International Symposium on Circuits and Systems*, IEEE press. IV: 405-408.

VanRullen, R., Gautrais, J., Delorme, A., Thorpe, S. (1998) Face processing using one spike per neurone. *BioSystems*, 48 (1-3) pp 229-239. http://www.klab.caltech.edu/~rufin/OriginalPapers/VanRullen98BioSystems.pdf, Accessed 05-May-2002.

VanRullen, R., Thorpe, S.J. (1999) Spatial attention in asynchronous neural networks. *NeuroComputing* , 26-27 pp 911-918. http://www.klab.caltech.edu/~rufin/OriginalPapers/VanRullen99NeuroComputing.pdf, Accessed 05-May-2002.

Wagner, C., Stucki, J. W. (2001) Construction of an Associative Memory Using Unstable Periodic Orbits of a Chaotic Attractor*, J. Theor. Biol. 2001*, in press. http://www.cx.unibe.ch/~jstucki/papers/upo.pdf, Accessed 05-May-2002.

Watta, P., Ikeda, N., Artiklar, M., Subramanian, A., Hassoun, M. (1999) Comparison Between Theory and Simulation for the Two-Level Decoupled Hamming Associative Memory, *International Joint Conference on Neural Networks (IJCNN99)*, CD ROM Proceedings paper number JCNN0337, Washington D.C. July, 1999.

ACKNOWLEDGEMENTS

The author would like to acknowledge the information provided by Henry Stapp at Lawrence Berkley National Laboratory regarding the quantum aspect of Mind.

COPYRIGHT