1 Solving the Assignment Problem Using Neural Networks

Current software systems suffer from an exponential increase in computational complexity when solving a quadratic assignment problem. This document considers the problem and proceeds to propose a solution using the inherent parallelism of a neural network to reduce computation times. A specific example is given, in this case a crossbar switch, onto which problem mapping is demonstrated and a solution given.

1.1 The Assignment Problem

As the complexity of modern communications and computational systems increases so does the need to develop new techniques which deal with common assignment problems ([4] and [5]) in situations such as:

- Network and service management.
- Distributed computer systems.
- Work Management systems.
- General scheduling, control or resource allocation problems.

The common assignment problem is essentially optimising task allocation to all available resources thus maximising throughput. In a distributed computer system this results in a many process computation being finished in the shortest possible time whereas, in a network management system, packets are routed to optimise throughput and minimise blocking.

This document examines specifically the assignment problem in a crossbar switch for packet routing [9]. These switches are present in many telecommunication systems and computer networks, one good example being ATM (Asynchronous Transfer Mode) networks.

1.2 Neural Network Implementation

The problem of packet routing in crossbar switches is known to be analogous to the travelling salesman problem (TSP). The TSP problem is a renowned NP complete problem [14] which means that although it can be solved by linear programming techniques, such as the Murnkes algorithm [15], it is computationally intensive and complexity grows exponentially as its order increases. Thus, a simple single processor solution will not provide satisfactory scalability.

One alternative is to apply a neural network to the TSP problem [6], [7]. The advantage of a neural net lies in the speed obtained through its inherent parallel operation, especially when dealing with large problems. Such an implementation will easily outperform any other method at higher orders of network size ([1], [2], [3], [4], [8], [12] and [13]) providing a very good, but not optimal, solution. It has been shown [4] that, at lower orders of network size, the average solution is within 3% of optimal. However, as the network size grows this figure improves slowly and begins to approach the optimal solution.

The problem which remains with any neural network solution is its adaptation to act as a controller for a crossbar switch.

1.3 Crossbar Switches and Notation

A crossbar switch can be simply abstracted as a set of n inputs and n outputs where each input can be switched to any output.

An example of this can be seen in figure 1 where, by simply closing the correct crosspoint switch, any input line may be connected to any output line. This system has the limitation that it is mutually exclusive: any input or output lines that are in use cannot be reused. Thus, two incoming requests for the same output line will result in one becoming blocked regardless of the routing algorithm which is used.

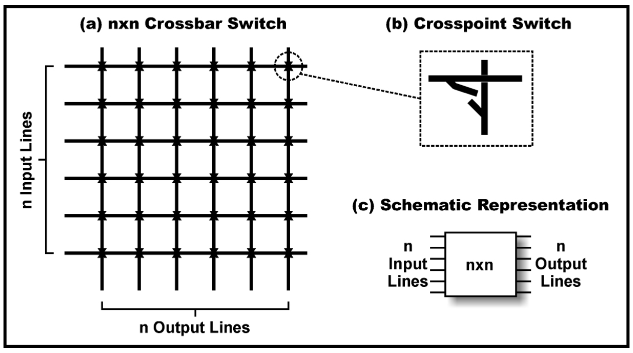

Figure 1

An NxN crossbar switch is shown here at various levels of detail.
(a) Shows an overall connection diagram for a typical crossbar switch.
(b) Details how each of the crosspoint switches work.
(c) Depicts a high level schematic of a crossbar switch.

To clarify the notation used throughout the rest of the document, please examine figure 2. This diagram details how a matrix may be mapped onto the crossbar switch, each crosspoint having a corresponding matrix element. A specific element in any matrix y can therefore be referenced using y_{ij}, where i is the input line and j the output line. Every element in the matrix can take on one of two values: 1 when there is a connection (or connection request) or 0 otherwise. The value and legality of the matrix is dependent on situation. Please examine the matrices shown in equations 1 and 2 overleaf.

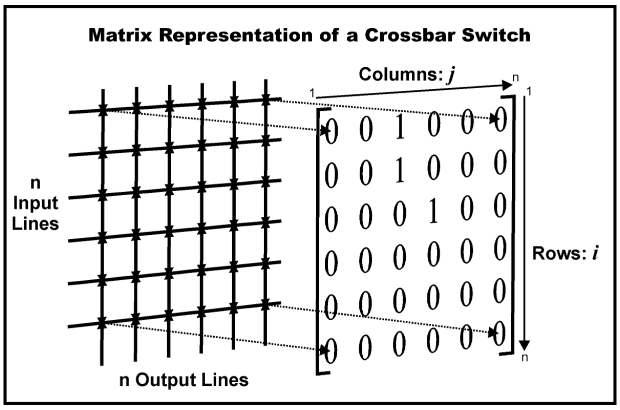

Figure 2

This diagram shows how a matrix can be mapped onto the crossbar switch thus aiding representation.

These matrices represent the crossbar switch in figure 2 but from different points of view. Equation 1 represents a set of desired connections where three input lines have requested connection to two different output lines: one

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Equation 1

This matrix shows a set of requested connections. Input i=1 has requested a connection with output j=3 and both inputs i=2 and i=3 have requested a connection to output j=4.

$$y = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Equation 2

This matrix shows a solution or response to the request in equation 1. It is legal because there are no other connections on the input rows and output columns which have been selected.

request is obviously going to have to wait. Such a matrix is legal regardless of the combination of zeroes and ones. Equation 2 shows a sample response. One request has been discarded in favour of another since only one input line can be connected to one output line at a time. A response is considered to be legal if there are no other closed switches on the same lines, i.e. all other elements in the same row and column as the active element must be zero.

The real optimisation problem comes in when you start to consider a system which has buffered input In such systems there can be multiple packets waiting on a single input line for various output lines, as can be seen in equation 3. Requests for multiple connections can be seen in the left matrix and the only optimal solution which maximises throughput on the right. This request matrix proves useful for testing crossbar control systems.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Equation 3

The left matrix shows a request and the right the only optimal response. This matrix is useful for testing a system.

As an enhancement to packet systems, each element could be converted to an integer value representing the number of packets waiting on each connection.

1.4 The Neural Network

The key to utilising the parallelism of a neural network is matching the network as closely as possible to the problem. For more information please refer to references [10], [16], [17], [18], [19] or [20].

1.4.1 The Neuron or Node

A neural network consists of a large number of processing elements called

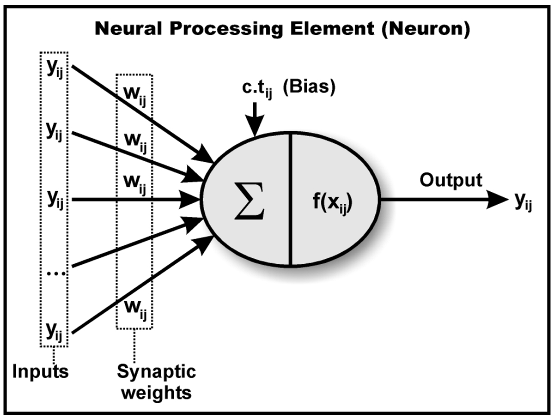

Figure 3

The building block of any neural network: the neuron.

neurons (see figure 3 or references [11] and [20]) which are highly interconnected to each other in a specific fashion. Neurons are the basic building blocks of neural networks and are an approximation of the neuron found in nature. A neuron takes inputs from other neurons' outputs y_{ij} (referenced by ij) and multiplies their strengths by a scalar weight w_{ij} known as the synaptic weight.

All inputs are summed by the neuron along with a specific bias to find x_{ij}. The neuron's output y_{ij} can then be determined using a monotonic activation function $f(x_{ij})$, as shown in equation 4. Here β is used to control the gain of the sigmoid function, a higher value resulting in a steeper transition, and o_{min} and o_{max} determine the minimum and maximum output values for y_{ij} respectively.

$$y_{ij} = f\left(x_{ij}\right) = o_{\min} + \frac{o_{\max} - o_{\min}}{1 + e^{-\beta x_{ij}}} \qquad \textbf{Equation 4}$$

The exact form of $f(x_{ij})$ is not particularly important and in fact any appropriate non-linear monotonically increasing function could be used. The preferred embodiment is, however, the sigmoid function.

1.4.2 The Updating Rule

Adapting a neural network to any problem requires that an updating rule is defined and thereby the network interconnection structure. The updating rule determines the next value that a neuron will take with respect to time based upon the previous outputs of other neurons, as shown in equation 5:

$$\frac{dx_{ij}}{dt} = i_{ij}\left(-\lambda_{ij} x_{ij} - \sum_{k \neq j}^{n} w_{ik} y_{ik} - \sum_{k \neq i}^{n} w_{kj} y_{kj} + c.t_{ij} \right) \qquad \textbf{Equation 5}$$

where:

x_{ij}: is a summation of all inputs to the neuron referenced by ij including the bias.

i_{ij}: determines whether a neuron referenced by ij is allowed to evolve: it can take a value of either 0 or 1.

λ_{ij}: time constant for neuron referenced by ij.

w_{ij}: synaptic weight for neuron input ij.

y_{ij}: output from neuron referenced by ij.

$c.t_{ij}$: Neuron specific threshold or bias.

and x_{ij} is related to y_{ij} using equation 4.

To illustrate this rule further, figure 4 overleaf shows an interconnection diagram for the modified system. Here the neuron marked with output y_{ij} has inputs from all the other neurons in the same row $-w_{2j}.y_{2j}$ and column - $w_{i2}.y_{i2}$. The important point to note here is that the neural network works in an inhibitory fashion so any active input will inhibit y_{ij}. $c.t_{ij}$ describes the external bias supplied to each neuron which is not inhibitory.

The idea behind this interconnection strategy is that any active neuron will try and turn all the others off, eventually resulting in only one of the requests remaining active in each row and column. However, to demonstrate its ability to find an optimal solution, the example in figure 4 needs to be extended slightly, as in equation 6. The left matrix here represents a request and the right its best case solution with y_{22} switched off. Careful consideration leads us to conclude that the network must converge to the solution shown here since both y_{24} and y_{42} are inhibiting y_{22}, thus resulting in it being switched off before the others and essentially losing. If y_{22} had won in this case then it would have resulted in a poor solution since y_{24} and y_{42} would be off: obviously not maximising potential throughput.

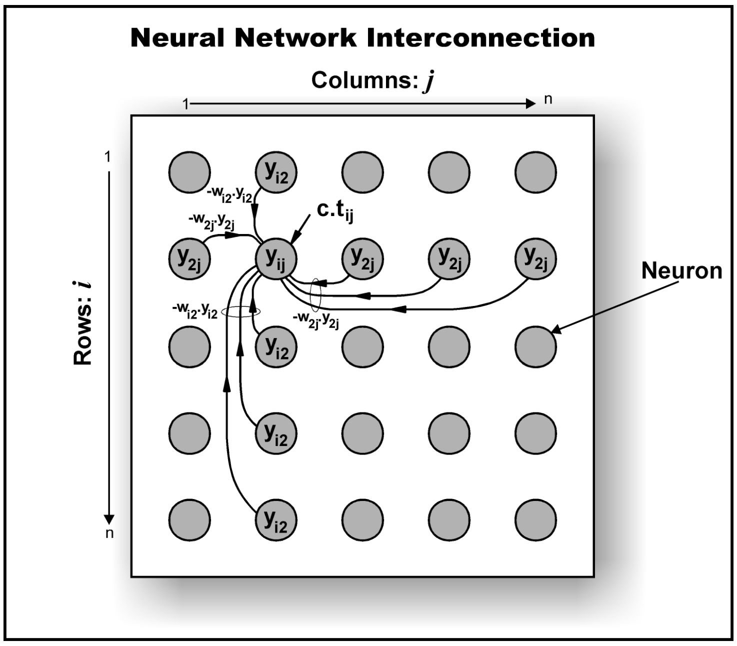

Figure 4

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Equation 6

The left matrix is a request and the right its solution.

It has been shown by Hopfield that with symmetric connections and a monotonically increasing activation function $f(x)$, the dynamical system described by the neural network possesses a Lyapunov (energy) function which continually decreases with time. The existence of such a function guarantees that the system converges towards equilibrium which is often referred to as a ‘point attractor’.

1.4.3 Local Minima

In any system with a continually reducing energy function, there is always a risk that the system will become trapped in a local minima. In this system, a local minima can be represented as a solution which satisfies the switching constraints but is not a global optimal solution. The best way round this problem is to introduce noise into the system by varying β. This alteration in the activation curve’s gradient is significant enough to provide successful convergence to a global minimum during network simulation.

1.5 Conclusions

Simulation of the system has proven not only that the system works but that it is highly scaleable and has underlined two important points:

- Noise plays a very significant role in this model. As the noise level increases, the time taken for network stabilisation decreases. However, when the noise value reaches unity the network becomes unstable and does not provide a valid or steady solution.
- Network size plays an important role in convergence to a solution: the larger it is, the longer it takes to converge.

What makes this system so interesting is its diversity: switching is only one of its many applications. Essentially, this system could be used to solve any quadratic assignment problem where time is of the essence. Its ability to handle larger order problems without serious performance degradation emphasises the contribution such systems could make to the field of computing.

2 Bibliography

[1] Peter W. Protzel, Daniel L. Palumbo and Michael K. Arras, *“Performance and Fault-Tolerance of Neural Networks for Optimisation”*, IEEE Transactions on Neural Networks, volume 4, number 4, July 1993.

[2] C. Bousoño-Calzón and M. R. W. Manning, *“The Hopfield Neural Network Applied to the Quadratic Assignment Problem”*, BT Labs paper, Martlesham Heath, Ipswich, IP5 7RE, publication date unknown.

[3] Joydeep Ghosh, Ajat Hukkoo and Anjun Varma, *“Neural Networks for Fast Arbitration and Switching Noise Reduction in Large Crossbars”*, IEEE Transactions on Circuits and Systems, volume 38, number 8, August 1991.

[4] M. R. W. Manning and M. Gell, *“Evaluation of the Hopfield Neural Network for Service Assignment”*, BT Labs paper, Martlesham Heath, Ipswich, IP5 7RE, publication date unknown.

[5] W. J. Wolfe, J. M. MacMillan, G. Brady, R. Mathews, J. A. Rothman, D. Mathis, M. D. Orosz, C. Anderson and G. Alaghband, *“Inhibitory Grids and the Assignment Problem”*, IEEE Transactions on Neural Networks, volume 4, number 2, March 1993.

[6] J. J. Hopfield and D. W. Tank, *“‘Neural’ Computation of Decisions in Optimisation Problems”*, Biological Cybernetics, volume 52, pages 141-152, 1985.

[7] R. D. Brandt, Y. Wang, A. J. Laub and S. K. Mitra, *“Alternative Networks for Solving the Travelling Salesman Problem”*, IEEE International Conference on Neural Networks, 24th to 28th Feb. 1998, San-Diego.

[8] T. X. Brown, *“Neural Networks for Switching”*, IEEE Communications Magazine, November 1989.

[9] T. X. Brown, *“Chapter3: Controlling Circuit Switching Networks”*, Extract from T. X. Brown’s Thesis from CalTech.

[10] J. J. Hopfield, *“Neural Networks and Physical Systems with Emergent Collective Computational Abilities”*, Proc. Natl. Acad. Sci. USA, volume 79, pages 2554-2558, April 1982.

[11] J. J. Hopfield, *“Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons”*, Proc. Natl. Acad. Sci. USA, volume 81, pages 3088-3092, May 1984.

[12] A. Marrakchi and T. Troudet, *“A Neural Net Arbitrator for Large Crossbar Packet Switches”*, Circuits and Systems Letters, IEEE Transactions on Circuits and Systems, volume 36, number 7, July 1989.

[13] S. B. Aiyer, M. Niranjan and F. Fallside, *"A Theoretical Investigation into the Performance of the Hopfield Model"*, IEEE Transactions on Neural Networks, volume 1, number 2, June 1990.

[14] M. R. Garey and D. S. Johnson, *"Computers and Intractability"*, New York, W. H. Freeman, 1979.

[15] J. Munkres, *"Algorithms for Assignment and Transportation Problems"*, J. Soc. Ind. Appl. Math., 5, 32-8, 1957.

[16] Robert L. Harvey, *"Neural Network Principles"*, Prentice Hall International Editions, 1994.

[17] James A. Anderson, *"An Introduction to Neural Nets"*, IT Press, 1995.

[18] Simon Haykin, *"Neural Networks"*, Macmillan Publishing Company, 1994.

[19] Clifford Lau, *"Neural Networks: Theoretical Foundations and Analysis"*, IEEE Press, 1991.

[20] J. Hertz, A. Krough and R. G. Palmer, *"Introduction to the Theory of Neural Computation"*, Addison-Wesley, 1991.