APPLICATION OF NEURAL NETWORKS

TO FIND THE SHORTEST ROUTING PATH

By

BERNARD MONTEBELLO

June 13, 2014

A DISSERTATION SUBMITTED IN PARTIAL

FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE (HONORS) IN COMPUTER NETWORKS

Declaration of Authorship

I, Bernard Montebello (0137493M), declare that this dissertation entitled *"APPLICATION OF NEURAL NETWORKS TO FIND THE SHORTEST ROUTING PATH*" and the work presented in it is my own.

This dissertation is based on the results of research carried out by myself, is my own composition, and has not been previously presented for any other certified or uncertified qualification.

The research was carried out under the supervision of Mr. Alan Gatt

Signature: _____ Date: _____

Copyright Statement

In submitting this dissertation to the MCAST Institute of Information and Communication Technology (ICT) understand that I am giving permission for it to be made available for use in accordance with the regulations of MCAST and the College Library.

Signature: _____ Date: _____

Contact Address: 46, ‘The Clover, Triq il-Mitjar, LQA 1191 Luqa (Malta).

Acknowledgements

Special thanks also go for my family and friends who patiently listened to me and forever being a source of courage during the most trying times and for their continued support, unconditional love and determination which have enabled me to complete such dissertation.

Special thanks go to my mentor, Mr. Alan Gatt, for sharing with me his ideas and opinions, he has been an endless source of inspiration ever present since the very beginning. His enthusiasm, dedication and guidance have inspired me and kept me going.

Contents

Table of Equations

List of Tables

Table of Figures

Abstract: An application of Hopfield Neural Networks algorithms in present routing communication networks in a Local Area Network (LAN) is suggested. Hopfield neural networks offer potential advantages such as the possibility of working with uncertain data, learning and flexibility. Hopfield Neural Networks Algorithms were compared to popular non-neural algorithms such as Dijkstra, Bellman-Ford and Floyd-Warshall each of the latter presenting protocols such as Routing Information Protocol (RIP) and Open Shortest Path First (OSPF). Under appropriate assumptions the packet-switched communication networks may also be considered as a shortest path problem and an optimization problem. This optimization problem and shortest path problem can be solved using a Hopfield Neural Networks as a solution. The applied Hopfield Neural Network algorithms were designed to simulate and find the optimal path within a network considering link metrics representing the number of hops, the speed of the path, and packet loss on the link, latency, path reliability, path bandwidth, throughput and load. The applicability of such simulation is shown through various computer simulations which include various link costs and different topologies which include routers (nodes) with bi-directional traffic capabilities.

Chapter 1 – Literature Review

1.1 Introduction

The traditional model of routing in computer networking is that of having a packet switched network which uses different routing protocols. These routing protocols will then enable communication between two nodes (routers) in computer networks.

A different approach to this traditional model of routing in computer networks will be taken. This different approach will consist of identifying whether neural networks can be used as an alternative solution to the existing algorithms used for shortest route problem in computer networks. A subset of neural networks, specifically Hopfield Neural Network (HNN) algorithms will be presented in this research.

This research will be a means to investigate different algorithms that can be used further for adaptive routing and be implemented on routers which can be designed with various adaptive learning mechanisms and aided by application integrated circuits. These neural network algorithms may also offer the possibility of shaping traffic differently than the present model of networking by identifying shortest paths based on specific criteria by which packets can be forwarded giving network engineers and administrators faster networks and flexibility as traffic conditions may rapidly vary in time.

There are a number of algorithms based on Hopfield Neural Networks (HNN) these include Ali & Kamoun Algorithm, Park & Choi Algorithm, Ahn & Ramakrishna Algorithm, and Park & Keum Algorithm. These algorithms will be individually examined and compared against each other and

against other non-neural network algorithms such as Dijkstra Algorithm (uses Open Shortest Path First Protocol), Bellman–Ford Algorithm (uses RIP Protocol), and Floyd–Warshall Algorithm.
The comparison between the Hopfield Neural Network Algorithms will be done mainly on convergence rate and optimality of the algorithms as they are simulated on different network topologies and also having various random metrics that represent various metrics. While the comparison between Hopfield Neural Networks and Non-neural Algorithms will be based on the total metric cost which will represent the shortest path.

The shortest path test will be carried out on using all neural and non-neural algorithms. Eventually the best neural network algorithm can be selected from both the Neural Networks and the non-neural algorithms. This will depend on the total 'link cost' which is used as a metric (an individual link cost identifies the speed of the medium in terms of a communication line). This will enable us to identify the best optimal neural network in various controlled 20 node, 40 node and 180 node scenarios that will simulate asymmetrical traffic with random weights.
The simulated environments are specifically made to be presented as a Local Area Network (LAN), this was done so that the results are taken from a more controlled environment and thus achieve a better qualitative result reflecting the true results of applying both neural and non-neural algorithms. Wide Area Networks (WAN) are more complex and as in a WAN topologies are not pre-defined and can have unknown structures.

The performance of each algorithm will be viewed when routes with various costs will be generated over a number of simulations. This will identify the most suitable algorithm for each scenario with the given parameters.

1.2 Network Optimization

The concept of network optimization is an evolutionary progress. In many networks, optimization is a need, since it affects the design, implementation, the efficiency and effort of any system. Optimized routing is important as it can help to find the optimal path between source and destination nodes (routers in a network). However, optimal routing is not very easy as "traffic is characterized by huge amount of source-destination pairs, high variability (burstiness), nonlinearity and unpredictability, the routing policy is a very hard task." **(Nenad Koji´c, 2006)** Under appropriate assumption neural networks can be implemented and achieve positive results which also include high computational speeds in a changing network scenario environment. **(Nenad Koji´c, 2006)** Moreover, reliability quality and efficiency in networking rely on constant optimization. This optimization will finally benefit and give advantages to the end users using such networks, services or systems.

1.3 Algorithms to Solve Shortest Route Problem

The shortest path problem is first defined in graph theory[1] by Lonard Euler, data is mainly represented in mathematical and graphical format and according to Keijo Ruohonen, and one of the uses of graph theory is to give "a unified formalism for many very different-looking problems". Graph theory, hence gives a visual approach to solving problems. **(Ruohonen, 2013).** According to Ravindra K. Ahuja, "The shortest path problem arises when trying to determine the shortest, cheapest, or most reliable path between one or many pairs of nodes in a network." **(Ravindra K. Ahuja*, n.d.)**

Various approaches were developed over time to tackle the shortest route problem; these include a number of algorithms such as the Dijkstra **(Dijkstra, 1959)**, Bellman–Ford **(Bellman, 1958)** , A* **(Hart, n.d.)** And Floyd–Warshall **(Floyd, 1962)**.

Graphs in terms of shortest route problem can be categories as directed graphs with non-negative weights, directed graphs with arbitrary weights, directed unweighted graphs and directed acyclic graphs. These are used by the previously mentioned algorithms to ultimately solve the shortest route problem in different scenarios.

[1] Mathematical and computer science branch.

1.4 Routing Protocols and Routing Algorithms

1.4.1 Routing Protocol Routing Information Protocol (RIP)

The routing information protocol (RIP) is a protocol based on the distance vector algorithm such as Bellman-Ford algorithm. The routing information protocol (RIP) also uses hop count[2] as a routing metric.

The Routing Information Protocol (RIP) works by implementing a variety of the distance-vector algorithm. A router under the RIP works by advertising the cost of reaching other networks instead of reaching a particular node. The cost is defined between a router and a network in which the destination node is located. The network on which the source node is situated is not counted and does not use a forwarding table. Therefore, by advertising the number of networks, the hops can be identified and therefore the message can be sent from the source node to the destination node. **(Forouzan, 2013)**

Routers that use the RIP protocol and an autonomous system[3] use forwarding tables on each router. These forwarding tables include the destination network IP address, the next router IP address and the count in hops. These forwarding tables, identify the destination network/node and the hop count with the least cost to the destination node.

[2] Hop Count represents the total number of devices a data packet passes through.
[3] An autonomous system is one network or sets of networks under a single administrative control.

A RIP message has two types of messages: a request and a response. The RIP message request is sent to ask about a specific router in the network, while the response is the answer to the request message. **(Forouzan, 2013)**

A limitation of this RIP protocol is that it is limited to few networks which means few hops from any router. This makes it an autonomous system in terms that it is contained and used in a managed network such as ISP networks were routers are administrated and each hop (router) is carefully managed. This makes the size of network limited and some routes may be considered unreachable.

Another limitation of the RIP protocol is that it uses fixed metric and costs to compare alternative routes. According to the RFC 1058[4], this makes the protocol not appropriate for situations where routes need to be given real-time parameters or changing metrics and such introduction will make instabilities which such protocols are not designed to handle. **(Hedrick, June 1988)**

[4] RFC 1058 is the Request for Comments for Routing Information Protocol (RIP)

1.2.1.2 Routing Protocol - Open shortest path first (OSPF)

The open shortest path first (OSPF) is an open protocol based on the link-state routing algorithm such is the Dijkstra algorithm. In the open shortest path first (OSPF) protocol the cost of reaching the destination is also calculated from the source router (node). OSPF can assign weights based on throughput, delays, reliability and each link in the infrastructure. The hop count can also be taken as the cost in any infrastructure when using OSPF. The OSPF also creates a forwarding table for each router according to the 'weights', metrics given in the scenario.

OSPF protocol uses another level of hierarchy together with the autonomous system, as OSPF also uses a second level called an area. The area is a subsection of the Autonomous System (AS) which has a backbone area responsible to hold all the areas together in a network so as to enable OSPF to work efficiently in a networking environment. OSPF than uses Link State Advertisement[5] is used to advertise new nodes or links. There are five main types of link advertisements, router link, network link, summary link to network, summary link to AS and external link. Each of these link state advertisements will contribute to identifying the environment better in a network scenario. **(Forouzan, 2013)**

[5] Link State Advertisement is used to advertise new nodes or links.

1.4.2 Routing Delays

As the networking layer is not a perfect layer, delays are expected in such network environment. There are four major types of delays transmission delays, propagation delays, processing delays and queuing delays. All these delays form part of a total delay which ultimately slows the performance of that particular network.

Transmission Delays

This delay involves the transmission rate and also the packet length. By definition, transmission delay is when "the first bit of a packet is put on the line at time t1 and the last bit is put on at time t2" **(Forouzan, 2013)**, then the transmission delay is said to be (t1-t2). This makes the transmission delay longer for a longer packet, and also makes it transmit faster if vice versa.

$$\boldsymbol{Delay\ transmission} = \frac{Packet\ Length}{Transmission\ Rate}$$

Equation 1 Transmission Delay (Forouzan, 2013)

Propagation Delays

This delay is the time for a bit for a packet to travel from a point A to point B in the transmission media

$$\boldsymbol{Delay\ propagation} = \frac{distance}{propagation\ speed}$$

Equation 2 Propagation Delay (Forouzan, 2013)

Processing Delays

The processing delay is the time taken for a packet to be processed in a router. This processing might involve error-correction, removal of header, delivery of packet to output port and processing in the upper layers. The processing delay can differ for each and every packet in the network as different processing is required for each. **(Forouzan, 2013)**

Queuing Delays

As the router has an input and an output queue connected to each port, it may be the case that packets have to wait to be processed. This situation creates what is known as a queuing delay. Consequently, this delay is the waiting time in the input queue before being processed and also the waiting time in the output queue before it gets retransmitted. **(Forouzan, 2013)**

1.5 Shortest Path Algorithms

1.5.1 Dijkstra Algorithm

The Dijkstra algorithm (Dijkstra, 1959) was first compiled by Edsger Dijkstra was published in 1959. It is used to solve the shortest path problem from a source to a destination node. The node can represent various objects such as routers, places, houses and buildings amongst others. The Dijkstra Algorithm is also sometimes called the single-source shortest path problem which gives the opportunity to find the shortest paths from a given source to all points in the network, graph or tree by using such algorithm. This makes such Dijkstra Algorithm relate to a spanning tree as its outcome represents all the shortest paths from a particular node in a graph/network. **(Forouzan, 2013)**

1.5.2 Bellman–Ford Algorithm

Bellman-Ford algorithm (Bellman, 1958) solves the single-source problem. The major difference from the Dijkstra Algorithm is that this algorithm can also solve edge weights that have a negative value. Bellman-Ford algorithm consequently is capable of solving the shortest path from a single-source vertex to all the vertices in a weighted network, topology or graph.

According to International Journal of Advance Research in Science and Engineering, the Bellman–Ford algorithm is slower than the Dijkstra algorithm, but "more versatile, as it is capable of

handling graphs in which some of the edge weights are negative numbers." **(Sarika Tyagi, June, 2013)**

As the Bellman–Ford algorithm can have negative edge values it can also have negative cycles. A negative cycle is a cycle whose edges have a negative value, thus if a negative cycle occurs, the Bellman–Ford algorithm "cannot produce a correct shortest path answer if a negative cycle is reachable from the source." **(Sarika Tyagi, June, 2013)**

1.5.3 Floyd–Warshall Algorithm

Floyd–Warshall algorithm (Floyd, 1962) is also used to solve the shortest route problem. Floyd–Warshall algorithm uses all the pair's nodes to solve the shortest path problem. Floyd–Warshall algorithm is used mainly for graph analysis for finding the shortest path, just like the Bellman and Ford algorithm, it can be used to find negative edges and is also used to find the transitive closure[6] of a relation. **(Sarika Tyagi, June, 2013)** The algorithm identifies the shortest path between all pairs of vertices, though it does not return details of the paths.

[6] A transitive closure of a relation is when a relation 'R' on a 'set X' is transitive if, for all x, y ,z in X, whenever x relates to y and y relates to z then x relates to z. E.g. x => and y => z then x => z.

1.6 Neural Networking

Neural networks can be considered as having a neural node structure where high data processing is done, which can simulate the biological nervous system of the brain. Therefore a highly complex network of computers can be compared to a brain nervous system where parallel and non-linear information is processed.

This structural organization of neurons is complemented by the ability of learning which is referred to as experience. Experience is learned over time and develops the neurons more to give a better ability to the individual.

According to Simon Haykin, a development neuron is "synonymous with a plastic brain: Plasticity permits the developing nervous system to adapt to its surrounding environment." **(Haykin, 1999)** This plasticity makes neural networks a very particular area not only in biomedicine but also in the computing networking field as it offers a model of an adapting, learning and high processing structure which could be useful for many areas such as network routing, pattern recognition, monitoring, data-verification, industrial process control, and data-validation.

The two main aspects of a neuron are the ability to learn from the changing environment and the ability to interconnect neurons together assigning them synaptic weights to achieve such computing power. **(Haykin, 1999)** These synaptic weights will therefore lead to the concurrent problem solution faster and achieve better results. According to Haykin, “a neural network is a massively parallel distributed processor that has a natural propensity for storing experimental knowledge and making it available for use.” **(Haykin, 1999)** Also the neural network is similar to a "network that acquires the knowledge through a learning process and by being a network that uses synaptic weights to store the knowledge." **(Haykin, 1999)**

Therefore being similar to the brain, according to Amit Parasmal Borundiya **(Borundiya, March 2008)**, a "neural network tries to simulate some of the properties of the brain for solving the real world problems, such as speech recognition, image analysis, and some NP hard problems which are difficult to solve using the conventional computers."

Making artificial neural networks a candidate for such problem solving one must keep in mind the properties of the artificial neural networks. These properties are described by **(Haykin, 1999)** below as:

- By construction of an input and output mapping, an artificial neural network learns. Learning can be either supervised or unsupervised.
- Neurons are interlinked and each neuron in the same network of neurons affects other neurons.
- Artificial Neural Networks also are capable of presenting the information gathered in the learning phase which is also organized.
- Artificial Neural Networks neurons are capable of being fault tolerant, neurons are capable of fixing themselves if damaged and this makes them more robust.

Therefore, as a result of such properties one can conclude that neural networks are an artificial representation of the neurons found in a brain. As neurons can transmit information and data while being interconnected so does an artificial neural network. As a normal brain, a neural network also has a learning capability; which can be either supervised or unsupervised.

1.6.1 HNN – Hopfield Neural Networks

Another area of artificial neural networks is the Hopfield networks. Hopfield networks are characterized by artificial neurons. Artificial neurons in Hopfield networks have N inputs. Each of these artificial neurons with their respective inputs (i) have an associated weight (wi). These neurons also have an output. In the Hopfield network the neuron can also be updated by following certain procedures. According to the Council for Innovative Research **(Maity, 2013)**, these procedures are:

- The value of each input, Xi is determined and the weighted sum of all inputs, Σi Wi Xi, is calculated.
- The output state of the neuron is set to +1 if the weighted input sum is larger or equal to 0. It is set to -1 if the weighted sum is smaller than 0.
- A neuron retains its output state until it is updated again.

Written as a formula:

$$\boldsymbol{O = 1 : \sum i\, Wi\, Xi \geq 0}$$

$$\boldsymbol{-1 : \sum i\, Wi\, Xi < 0}$$

Equation 3 Neuron Update Formula

(Maity, 2013)

A Hopfield neural network (HNN) is also defined as a network of artificial neurons, just like artificial neural networks (ANN). According to the Council for Innovative Research, a Hopfield neural network is " a network of N such artificial neurons, which are fully connected**." (Maity, 2013)**

As already identified neurons can be updated, according to the Council for Innovative Research there are two main methods of updating a neuron "asynchronous" and "synchronous".

In the asynchronous updating method each neuron is chosen at random (asynchronous random updating) or specifically and the weighted input sum is calculated and then updating the neuron immediately. **(Maity, 2013)**

On the other hand in the synchronous updating method the weighted sum of all neurons is calculated without updating the neurons than all neurons are set to their new obtained weighted sum value. **(Maity, 2013)**

1.6.1.1 Lyapunov function - Energy Function and Algorithms

The Lyapunov function was developed by A. Lyapunov in 1899 for the study of stability of dynamical systems. Since then the Lyapunov functions have been extended to study stability of dynamical systems of all kinds, including chaotic systems, control systems, discrete systems and in this case neural network algorithm for computing algorithms to solve the shortest path problem**. (Konstantopoulos, n.d.)**

According to the Stanford University, the Lyapunov function can be considered as a "generalized energy function for a system" **(University, 2008-2009).** This is possible because the Lyapunov theory is created to make "conclusions about trajectories of a system without finding the trajectories" **(University, 2008-2009)**.

The typical Lyapunov theorem, according to Stanford University, has the form:

$$\boldsymbol{V : R^n \rightarrow R}$$

Equation 4 Lyapunov Function - Generalized Energy Function

This equation shows that if there exist a function that satisfies some conditions of V and v not, then the trajectories of system satisfy some property. Moreover, according to the Stanford University, if such a function V exists, we call it a Lyapunov function and that proves the property holds for the trajectories. **(University, 2008-2009)**

According to Dong-Chul Park et al. the HNN has been successfully used to solve various optimization problems known as NP-complete **(Yip, 1995). (Keum, 2008)** The energy function of the Hopfield model is defined as follows **(Keum, 2008)**:

$$E = \frac{1}{2}\sum_{i-1}^{P}\sum_{j-1}^{p} L_{ij}\, V_i\, V_j + \sum_{i-1}^{p} \theta_i\, V_{\ i}$$

Equation 5 Energy Function of the HNN

Where P, Vi and θi, are the number of neurons, the output and bias of the ith neuron, respectively. **(Keum, 2008)**

1.6.1.2 Ali and Kamoun

Ali and Kamoun in 1993, in order to "outperform" the limitations of the prior proposed a "novel adaptive algorithm, where the weight matrix just carries convergence information" **(Aliabadi, n.d.)**

A type of Hopfield Neural Network (HNN) was used by Ali et al (1993) is the continuous Hopfield neural network. **(Jzau-Sheng Lin *, April 21, 2000)** According to Jzau-Sheng Lin et al, this type of HNN was applied to the optimal routing problem in the packet-switched computer network to minimize "network-wide average time delays". Ali and Faouzi Kamoun et al. (1993) have argued in their publication that under appropriate assumptions the optimized routing algorithm relies heavily on shortest path computations **(Ali MM, n.d.).** Ali and Faouzi Kamoun paper thus focuses on the a " proposed model that will enable the routing algorithm to be implemented in real time and also to be adaptive to changes in link costs and network topology." **(Ali MM, n.d.)**

A modified energy function was created to enhance the Hopfield Energy Function. Ali and Kamoun proposed such Hopfield algorithm so that the optimal solution would be found. The equation is as follows:

$$E = \frac{\mu_1}{2}\sum_{i-1}^{N}\sum_{\substack{j-1 \\ j\neq i}}^{N} L_{ij}\, v_{ij} + \frac{\mu_2}{2}\sum_{i-1}^{N}\sum_{\substack{j-1 \\ j\neq i}}^{N} \gamma_{ij}\, v_{ij} + \frac{\mu_3}{2}\sum_{i-1}^{N}\left(\sum_{\substack{j-1 \\ j\neq i}}^{N} V_{ij} - \sum_{\substack{j-1 \\ j\neq i}}^{N} V_{ji}\right)^2$$

$$+ \quad \frac{\mu_4}{2}\sum_{i-1}^{N}\sum_{\substack{j-1 \\ j\neq i}}^{N} V_{ij}\,\left(1 - V_{ij}\right) + \frac{\mu_5}{2}\,\left(1 - V_{ds}\right)$$

Equation 6 Ali and Kamoun Modified Energy Function (Keum, 2008)

According to Dong-Chul Park et al., this modified energy function initializes the values μ_1 to minimize the cost of a route by taking into consideration all the existing links, μ_2 excludes the non-existent links from being included. Then the μ_3 term is given the '0' as a value if the number of incoming direction links equal the number of outgoing links. The μ_4 term is there to keep the modified equation to converge to a valid route while the μ_5 term includes the source and destination node such that a valid, converged path is formed. **(Keum, 2008)**

1.6.1.3 Park and Choi

Consequently, following Ali and Kamoun who have proposed an adaptive algorithm that used information regarding link cost, according to the given topology which could result in slow convergence **(C.W. Ahn, 2001)**, Park and Choi et al have proposed a modified algorithm. (C.W. Ahn, 2001) This modified algorithm enhanced the perform speed and convergence however according to the Electronics Letters this algorithm is still dependent on the network topology. **(C.W. Ahn, 2001)**

Park and Choi et al concluded in their paper that a three process approached is used, this approach is taken from the conclusion of their paper" first the recursive Hopfield neural network is used to determine the optimal routing order among source and multiple destinations, next the screening procedure to get a simplified network for routing with multiple destinations of the network is applied, and then the improved routing algorithm based on Hopfield neural network is simultaneously applied to the simplified network to find the overall path." **(Choi, n.d.)**

As suggested by Park and Choi in their conclusion, "A Neural Network Based Multi-Destination Routing Algorithm for Communication Network" **(Choi, n.d.)**, such algorithm can be compared to Dijkstra algorithm, Kruskal algorithm and Ali and Kamoun and shows "promising results" especially when used on large scale problems and large number of nodes. **(Choi, n.d.)**

Park and Choi proposed another Hopfield neural network energy function which does not include loops or partitions and is practical for large networks making such HNN an improvement on the previous algorithm. The equation used is as follows:

$$Epc = \frac{\mu_1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} L_{ij}\, v_{ij} + \frac{\mu_2}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} \gamma_{ij}\, v_{ij} + \frac{\mu_3}{2}\sum_{i=1}^{N}\left(\sum_{\substack{j=1 \\ j\neq i}}^{N} V_{ij} - \sum_{\substack{j=1 \\ j\neq i}}^{N} V_{ji} - \emptyset_i\right)^2$$
$$+ \frac{\mu_4}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} V_{ij}\,(1 - V_{ij}) + \frac{\mu_5}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N} v_{ij}\, v_{ji}$$

Equation 7 Park and Choi Modified Energy Function (Keum, 2008) (Choi, n.d.)

In this equation $\emptyset$ represents 1 if " i = s" and -1, If "i = dn" or 0 otherwise. Where "s" is the source node and "d" is the destination node.

1.6.1.4 C.W. Ahn and R.S. Ramakrishna

C.W. Ahn and R.S. Ramakrishna proposed another Hopfield neural network (HNN). According to the shortest path routing algorithm using Hopfield neural network paper (C.W. Ahn, 2001), this "new approach was proposed to speed up convergence while improving route optimality and which is relatively insensitive to variations in the network topology". Also, this approach avoided loops as it added two terms into the previous Park-Choi energy function in the algorithm. **(C.W. Ahn, 2001)** Also the algorithm provided by Ahn et al. was an improvement over the other two approaches yet it was difficult to implement as "the energy function has seven terms and it is hard to tune the associated seven parameters." **(C.W. Ahn, 2001)**

In their paper (C.W. Ahn, 2001), the C.W. Ahn et al. proposed algorithm uses all the information acquired from the neurons to be able to achieve "speedy and optimal" convergence. Moreover, this convergence is able to happen as the proposed algorithm "draws on the highly correlated knowledge at the local neuron" which means that it uses all the required information obtained from the neuron which "counters a tendency to converge to one of the suboptimal paths reachable". **(C.W. Ahn, 2001)**

As to improve the speed of convergence and also the quality of the energy function Ahn and Ramakrishna have created a new modified energy function as an improvement on the previous modified functions. The equation used is as follows:

$$Ear = Epc + \frac{\mu_6}{2}\left(\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N}\left(\sum_{\substack{k=1 \\ k\neq i,j}}^{N} V_{ik} - 1\right) v^2{}_{ij}\right) + \frac{\mu_7}{2}\left(\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N}\left(\sum_{\substack{k=1 \\ k\neq i,j}}^{N} V_{kj} - 1\right) v^2{}_{ij}\right)$$

Equation 8 Ahn and Ramakrishna Hopfield Energy Function (Keum, 2008) (C.W. Ahn, 2001)

One can notice that in such modified energy function the terms μ_6 and μ_7 were added to further eliminate loops and partitions while having a more speedy and accurate convergence.

1.6.1.5 Dong-Chul Park and Kyo-Reen Keum

This algorithm according to Dong-Chul Park et al. is also another shortest path routing algorithm using the HNN with a modified Lyapunov function (energy function). **(Keum, 2008)** This modified function allows the routing order from a source to multiple destinations to be determined. **(Keum, 2008)**The proposed energy function mainly prevents the solution path from having also other loops and partitions, according to the Dong-Chul Park et al. **(Keum, 2008)**

Taking into account the other modified functions Park and Keum have also mathematically computed another energy function which optimized and obtained a better quality route when compared to the other modified functions. The equation used by Park and Keum is as follows:

$$E = \frac{\mu_1}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N}\left(\sum_{\substack{k=1 \\ k\neq i}}^{N}(1 - C_{ik})C_{ij}\right) + \frac{\mu_2}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N}\gamma_{ij}\, v_{ij} + \frac{\mu_3}{2}\sum_{i=1}^{N}\left(\sum_{\substack{j=1 \\ j\neq i}}^{N}V_{ij} - \sum_{\substack{j=1 \\ j\neq i}}^{N}V_{ji} - \emptyset_i\right)^2$$

$$+ \frac{\mu_4}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N}V_{ij}\,(1 - V_{ij}) + \frac{\mu_5}{2}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq i}}^{N}\left(v_{ij}\sum_{\substack{k=1 \\ k\neq i}}^{N}v_{ki}\right)$$

Equation 9 Park and Keum Hopfield Energy Function (Keum, 2008)

One can notice that in such modified energy function the terms μ_6 and μ_7 were omitted as the function does not require fast convergence but was designed to be more stable which is done by the term μ_5.

2.1 Introduction

The purpose of this study is to determine whether neural networks can be applied to find the shortest routing path in a router application within a Local Area Network environment. Simulations were used to identify the shortest path from a given node to any other node on pre-defined topologies which were built to simulate a Local Area Network with different nodes size, for both Hopfield Neural Networks (HNN) and Non-Neural algorithms simultaneously.

As non-neural algorithms are presently used in networking protocols such as OSPF and RIP, Hopfield Neural Networks were simulated and compared to such present day network protocols. By simulating different scenarios the algorithms were compared on convergence rate, optimal path and best metric usage to demonstrate the use and application of Hopfield Neural Networks within protocols and networking.

The comparison of the Hopfield Neural Networks (HNN) and non-neural shortest path algorithms were computed using a Single-Destination and Multi-Destination approach. Both of these approaches would give different paths and different metrics which allow such shortest path algorithms to be compared.

A quantitative approach was used since various measurements, statistical data and tabulations were required to compare the Hopfield Neural Networks with the non-neural network algorithms.

2.2Application

2.2.1 Hopfield Neural Network Simulation Application

The neural network simulation was coded by professor Dong-Chul Park (Associate Editor for the IEEE Transactions on Neural Networks, IEEE Senior Member). This software was created to simulate different types of Hopfield Neural Networks, its source code was made publicly available on the university's website (http://icrl.mju.ac.kr/HNN_2.html) (Research, 2014) and was coded using C++. This code was converted to Java for this dissertation since it had missing graphical components. A graphical user interface was added to offer more interaction and a better visual representation of the work.

Further features were added to the simulation application; these features enabled the simulation of network topologies (LANs) having a source and destination node and also two-way undirected traffic. The added features also enabled the capability of having different values in both directions. This change involved creating new matrix and arrays to support the links needed for the topologies created.

2.2.2 Non-Neural Network Simulation Application

The non-neural simulation application was coded in Microsoft Visual Studio 2010, C# and .NET 4.0 making use of the Component Factory Krypton toolkit (Ltd, 2013). This program was created in Ventspils University (Research, 2014) and it is used to apply the non-neural network algorithms Dijkstra, Bellman -Ford and Floyd-Warshall on the topologies constructed which then can be compared to the non-neural solution.

The modifications to the program were done so that the workspace area could be enlarged to enable larger topologies to fit.

Another modification was made which allowed random weights between the values of 0 to 1000 to be applied to t links so that these could have a wider range of random costs These weights represent the metric between each node making such modification a very important as it effects the results and the topologies within the dissertation.

2.3 Topologies

2.3.1 Undirected Graph Topologies

The topologies were mathematically computed by using an undirected graph which was necessary to stimulate a bi-directional traffic in LAN simulation. The undirected graph was used as the relationship between pairs of nodes is symmetric meaning that each edge has no particular direction. The mathematical aspect of undirected graphs were taken as G = (V, E), where V is the set of vertices (nodes), and E is the set of its edges (links). Also the cost between two nodes were mathematically computed by using a matrix W = [Wij], where Wij is the cost from node “i” to node “j”.

Taking into account each link [i, j] a non-negative number was assigned to Wij which is the metric cost from any node "i' to node "j". Another important mathematical function of having undirected graph is that if no link is present between two given nodes the Wij is set to a large number so that it is not a candidate to form the optimal routing path. This mathematical process was repeated for both link [i, j] and link [j, i] which was the other link created between each node to simulate different two-way metrics between two nodes, as node "i" to node "j" has a different metric from that of node "j" to node "i".

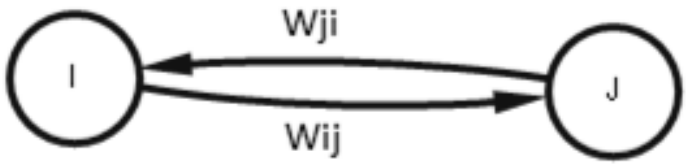


Figure 1 Undirected Graph Representation

2.3.2 Creation of Topologies and Metrics

The proposed solution of applying neural networks to find the shortest routing path to achieve optimal routing commences with the definition of an undirected graph. In this case the undirected graph represents various underlying topologies which were also created to simulate local area networks (LAN) scenarios of different proportions. The topologies included a 20 node, 40 node and 180 node scenarios. Every node represents a router with the capability of packet switching and forwarding which is connected in a LAN environment.

These scenarios were limited by the size of such topologies so the results could be more accurate in terms of convergence, comparison and dissimilarity to other pre-applied algorithms such as the Dijkstra, Bellman-Ford and Floyd-Warshall algorithms. The nodal size of these topologies demonstrate and simulate different application of algorithms which expressed the exponential growth in terms of algorithm convergence of both the applied neural networks and the non-neural algorithms.

The topologies were first created by assigning each node a link and also a random weight which represented the metric. In this experiment, metrics can include costs such as the number of hops, the speed of the medium, and packet loss on the link, latency, path reliability, path bandwidth, throughput and load. These factors will determine the values of the path and will be used by the algorithms, both neural and otherwise, to compute the shortest path first. These type of metrics are called additive metrics as they are the sum of the costs of individual links along that path.

Non-Neural Graphs were created using the C# application. Such topologies were created to represent two-way traffic as explained in section 2.4. The topologies were created to stimulate different types of Local Area Network structures to give a better realistic and authentic simulation.

Following the non-neural graphs, Hopfield Neural Network topologies were constructed. Every node represented a router with the capability of packet switching and forwarding which is connected in a LAN environment. These topologies were set up by creating an array that assigned the cost for each node and each node was created to withstand two values representing two different metric values as each link constituted of different metric cost. With such topology setup in java using arrays, which is one of the modification in the program mentioned in section 2.2, the shortest path could be found in multiple directions with the random metrics assigned in the produced topologies which were constructed. The topologies were first graphically built for the Non-Neural Algorithms than replicated in numeric form to be used in the java solution for the Hopfield Neural Networks.

Exponential growth is characterized by having increasing nodes in the different scenarios which symbolizes more routers and more links in the topologies thus representing a more practical and realistic simulation. Exponential growth on topologies is a necessity as it shows a better view in applying Hopfield Neural Networks in LAN environments.

The links in each scenario are created to simulate two way traffic and each link is given a different cost metric for each direction. This two way traffic simulation was necessary so that the application of neural networks will be more realistic in terms of infrastructure and set-up and consequently yield more accurate results. Consequently the shortest path could be found in multiple directions with the random metrics assigned in the produced topologies which were constructed.

2.4 Simulation

2.4.1 Single and Multi Destination Set-Up

2.4.1.2 Hopfield Neural Network and Non-Neural Traffic Single-Destination

There are two types of experiments to be done over topologies, a single source to a single destination and an experiment for a single source to a multi destination experiment. Each experiment provides a different data set as the cost of the optimal path is different. The optimal routing order from a single source node (router) to another single destination was identified by

$$S\ (source) \leftrightarrow D\ (Destination)$$

This also represents a single source and a single destination which have a bi-directional connection with each other. The topologies for the non-neural networks were also constructed using bi-directional connections.

Each node was given a letter to identify it and each link was given a cost. This information was listed in a legend in which the node letter and array position were clearly shown. This was required in order to construct both the neural and non-neural topologies in the same way.

Each non-neural algorithm was applied to all the topologies. Each topology produced three results these being the Dijkstra, Bellman-Ford and Floyd Warshall Algorithms.

Moreover, this was also used on Hopfield Neural Networks. The values of both source and destination then could be changed within the Hopfield Neural Network Program to obtain the result of the path from the given source node to the given destination node. When computing such bi-directional connection intermediate nodes will also be present and therefore the possibility of

calculating the optimal path is achieved. Each topology produced the necessary results for the Hopfield Neural Network Algorithms respectively.

Paths were adapted to the topology infrastructure respectively in the 20 node, 40 node and 180 node scenarios. Each path created, for Hopfield Neural Networks and Non-Neural Networks was identified uniquely and each path was chosen to satisfy different path directions. Each path was also unique in terms of link values so that the convergence cost could be attained. Paths were also constructed to give a different outcome and produce a different situation every time. This was done purposely so that a more realistic simulation would be given.

2.4.1.3 Non-Neural Traffic Multi-Destination

The algorithms were performed on each path uniquely and a path was divided in many sub-paths unlike the Hopfield neural network approach. This was done so that the total cost for multiple destinations could match with the total cost of the Hopfield neural networks. The optimal routing order from a single source node (router) to a multi-destination node was identified differently by

$$S\ (source)\ \leftrightarrow\ D1\ \leftrightarrow D2 \leftrightarrow D3 \ldots \leftrightarrow\ DN(Destination\ to\ nth)$$

$$S\ (source)\ \leftrightarrow\ D1$$

$$D1\ \leftrightarrow D2$$

$$D2\ \leftrightarrow D3$$

$$\ldots$$

$$D3 \leftrightarrow\ DN(Destination\ to\ nth)$$

Where ‘D’ is the destination and ‘S’ is the source. The above shows that each destination was computed as a source to another destination so that the optimal path could be achieved and compared with the pre-defined algorithm path given in the Hopfield Neural Network. This enabled to have a total cost for multiple destinations for the non-neural algorithms which could be compared with the Hopfield neural network total cost for multiple destinations. The topologies were also constructed using the C++ modified program and matched the Neural topologies in terms of cost and nodes and nodes were also linked together the same way.

2.5.1.4 Hopfield Neural Network - Traffic Multi-Destination Problem

The optimal routing order from a single source to a multi-destination was found by using the following equation:

$$S\ (source)\ \leftrightarrow\ D1\ \leftrightarrow D2 \leftrightarrow D3 \ldots \leftrightarrow\ DN(Destination)$$

Where "N" is the number of destinations, "S" is the source and "D" is the destinations computed in the progress which are predefined in the Hopfield Neural Network Program and can be modified to adjust the wanted path by changing the values for the destinations and have the wanted path in the scenario.

This optimal routing order allowed the creation of multiple paths with different nodes. The paths were created randomly but also specific for each topology infrastructure respectively in the 20 node, 40 node and 180 node scenarios. The paths were uniquely different and presented multiple routing orders. The paths were distributed along the topology and also starting from different nodes as a starting point. This was very important in the implementation as the network could be more realistic and also because with such implementation an experimentation of different routing scenarios could be extracted.

2.4.2 Non-Neural Algorithms

2.4.2.1 Applying Non-Neural Algorithms

The non-neural algorithms were deployed using the C# solution mentioned in section 2.3.2. The algorithms were performed on different paths which were created randomly with the intention of creating different instances of traffic patterns and also creating a more realistic environment.

Non-neural algorithms which included the Dijkstra, Bellman-Ford and Floyd-Warshall Algorithm were applied to the topologies respectively to create different paths including both single-destination and multi-destination approaches as explained in the above section. Dijkstra and Bellman-Ford produced the result graphically which could be interpreted. This interpretation included a graphical interface with the given topology and path. The result then included the shortest route path highlighted from the given source node. The shortest route was also presented with the total metric cost being represented with the Õ symbol and the path taken was represented with the Π symbol which enabled the shortest path trace from the source to the destination node, as explained in the figure below, where node K uses 'Π = J' and node J uses 'Π = A' which means that the shortest path from A to K is from 'Node A to Node J to Node K' respectively.

Figure 2 Total Metric Cost Non-Neural

The Floyd-Warshall Algorithm produced a different type of result as it produced a matrix form result. The matrix was also outputted graphically. The matrix results produced the optimal route with the needed total metric cost. The matrix result was produced on an excel workbook after the algorithm was applied to the relevant path in each of the respective topologies.

2.4.3 Hopfield Neural Networks (HNN)

2.4.3.1 Hopfield Neural Networks - Parameter Setting

Hopfield Neural Networks rely mostly on the parameter values for the equations used in networks to compile the most optimal path. Hopfield Neural Networks are sensitive to such parameter values, in this case the parameter values chosen were used as specified in the relevant papers of each of the neural algorithms. By setting and tuning such parameter values for the equations used by the neural network program the possibility of having an energy function that falls into a local minimum state is reduced and this means that the algorithm would converge better. In the following table are all the parameter values as described in each of the relevant papers:

Table 1 Parametric Values

Parametric Values	Symbol	Ali & Kamoun (Ali MM, n.d.)	Park & Choi (Choi, n.d.)	Ahn & Ramakrishna (C.W. Ahn, 2001)	Park & Keum (Keum, 2008)
Time Constant	τ	1	1	1	1
Slope of Logistic Function	λ	1	1	1	1
Incremental Time Step	δ	0.0001	0.0001	0.0001	0.0001
Weight Coefficients	μ_1	550	550	550	950
	μ_2	2550	2550	2550	2550
	μ_3	1950	1950	1950	1900
	μ_4	250	250	475	100
	μ_5	1350	1350	500	500
	μ_6	–	–	400	–
	μ_7	–	–	400	–

Applying such parameters will make the HNN converge better and give better results. The parameters include the time constant "τ" which is the time of each neuron present in the Lyapunov function used by the HNN algorithms. Another parameter includes an Incremental Time step "δ" which defines the incrimination (change) of time in the energy function, meaning the change in time for each neuron present.

The parameter Slope of Logistic Function is used by the HNN energy function as a special case of the Sigmund function, it is mainly used as an activation to the neuron within the neural network and serves to maintain the energy function within certain ranges. This helps the HNN Algorithms remain convergent most of the time as such parameter function would prevent a Local Minimum State of the energy function, where it would be incapable to converge otherwise.

The last parameters are the weight coefficients which were retrieved from the relative papers, the uses of such weights play an important role within the HNNs as they also allow for the energy function used by HNNs to provide convergences to a valid solution and also a better optimal path.

2.4.3.2 Applying Hopfield Neural Network Algorithms

The Hopfield Neural Algorithms were deployed using the Java solution, using algorithms from Ali and Kamoun, Park and Choi, C.W. Ahn and R.S. Ramakrishna. Each algorithm was applied to different topologies.

Paths for both single destination and multi-destination were created with random nodes from the respective topologies but also with the intention to create different patterns. These different paths are important as they simulate different types of traffic within a "Local Area Environment" and thus making a more realistic and practical simulation.

Each Hopfield Algorithm was applied by running it in the java solution and a result was outputted for the most optimal path which represented the most optimal path that traffic would take if such Hopfield neural network algorithms were used.

The Hopfield Neural Network algorithms were applied numerous times on each path on both the single-destination and multi-destination paths on all the topologies so that a good perspective and result could be conducted.

A number of results were conducted, each algorithm for Hopfield Neural Networks algorithms were run 10 consecutive times for each scenario respectively. Each time a different route was chosen and the path along with the respective cost were analyzed and noted. This meant that these results included 30 tests with different paths and different outcomes. Therefore each Hopfield Neural network algorithm was tested 30 times and compared to the non-neural algorithms that were also calculated.

Chapter 3 - Experiments and Results

3.1 Overview

All the results were computed using the two modified solutions and each result was uniquely taken on the predefined topologies which represent Local Area Network (LAN) scenarios. A total of 60 results were obtained by running simulations on the 20, 40 and 180 nodal topologies. The obtained results were all unique and different in terms of comparison, different path lengths, different costs and also different topologies. The main results show the convergence rate of each Hopfield neural network algorithm and the best optimal path cost of both neural and non-neural algorithms.

3.2 Single-Destination Results

3.2.1 Non-Neural Algorithms - Single-Destination Results

For the non-neural algorithm the same approach was adapted so that the optimal path can be found by computing the non-neural algorithms Dijkstra, Bellman-Ford and Floyd Warshall respectively over the topologies as necessary. This gave the required results that were needed to obtain the optimal path percentage and also the convergence rate percentage. The following result is a sample that shows the optimal path with the metric cost for each non-neural network algorithm in a single-source to a single-destination scenario. The following are a sample of the collected data and results of the single-destination for the Hopfield Neural Networks in a 40 Node Scenario.

Table 2 Sample Single Source to Single Destination - Dijkstra Algorithm

Dijkstra Path Traversed		
Source/Destination	**Path Taken**	**Cost(σ)**
Node 1 to 3	1-> 7-> 3->	676
Node 1 to 5	1-> 5->	607
Node 1 to 9	1->7->10->8->9->	875
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 7-> 10-> 17-> 18-> 22-> 20	1420
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	1-> 7-> 10-> 17-> 25-> 15-> 26-> 27-> 30->	1983
Node 1 to 34	1-> 7-> 10-> 17-> 25-> 15-> 26-> 27-> 34->	2311

Table 3 Sample Single Source to Single Destination - Bellman-Ford Algorithm

Bellman Ford Path Traversed		
Source/Destination	**Path Taken**	**Cost(ω)**
Node 1 to 3	1-> 7-> 3->	699
Node 1 to 5	1-> 5->	607
Node 1 to 9	1->7->10->12->9->	1264
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	1->7->10->17->25->15->26->24->30->	2603
Node 1 to 34	1->7->10->17->18->29->32->31->34->	3404

Table 4 Sample Single Source to Single Destination - Floyd Warshall

Floyd Warshall Path Traversed	
Source/Destination	**Cost(τ)**
Node 1 to 3	1457
Node 1 to 5	690
Node 1 to 9	366
Node 1 to 15	837
Node 1 to 20	1421
Node 1 to 25	870
Node 1 to 30	1595
Node 1 to 34	1910

The metric cost of these algorithms can then be compared to view the best optimal path and the best algorithm from both neural network and non-neural algorithms. This comparative result can be shown and explained better in the table 5 as shown below.

Table 5 Total Results for 40 Node Network - Single Source to Single Destination

	Hopfield Neural Networks Algorithms				Non-Neural Algorithms		
Source/Destination	Cost(α)	Cost(β)	Cost(ρ)	Cost(φ)	Cost(σ)	Cost(ω)	Cost(τ)
Node 1 to 3	843	843	843	843	676	699	1457
Node 1 to 5	607	607	607	607	607	607	690
Node 1 to 9	1749	1749	1749	875	875	1264	366
Node 1 to 15	-	839	839	839	839	839	837
Node 1 to 20	2026	2026	2026	2026	1420	2026	1421
Node 1 to 25	-	633	633	633	633	633	870
Node 1 to 30	-	-	-	1983	1983	2603	1595
Node 1 to 34	-	-	-	2311	2311	3404	1910

3.2.2 Hopfield Neural Networks Convergence - Single-Destination Result

A total number of 120 simulations were carried out, meaning that each algorithm was tested 30 consecutive times with different paths and yielded different costs in the process. The following is a sample result of the 40 nodal topology structure when using a Hopfield Neural Network algorithmic approach to obtain the single-destination result which was selected to show the performance of the algorithms in such instance.

A convergence chart is produced from multiple generated results obtained by using the Hopfield Neural Network Algorithm application. A percentage of convergence success rate for the single destination is also extracted. Table 6 shows the optimal path with the metric cost for each neural network algorithm, this was done to all the other produced results in order to calculate both the convergence rate and the most optimal path. The following are a sample of the collected data and results of the single-destination for the Hopfield Neural Networks in a 40 Node Scenario.

Table 6 Sample Single Source to Single Destination - Ali and Kamoun Algorithm

	Ali & Kamoun Algorithm Path Traversed	
Source/Destination	**Path Taken**	**Cost(α)**
Node 1 to 3	1-> 2-> 3->	843
Node 1 to 5	1-> 5->	607
Node 1 to 9	1-> 7-> 10-> 12-> 8-> 9->	1749
Node 1 to 15	No Convergence	No Convergence
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	No Convergence	No Convergence
Node 1 to 30	No Convergence	No Convergence
Node 1 to 34	No Convergence	No Convergence

Table 7 Example Single Source to Single Destination - Park and Choi Algorithm

Park & Choi Algorithm Path Traversed		
Source/Destination	**Path Taken**	**Cost(β)**
Node 1 to 3	1-> 2-> 3->	843
Node 1 to 5	1-> 5->	607
Node 1 to 9	1-> 2-> 3-> 8-> 9->	1749
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	No Convergence	No Convergence
Node 1 to 34	No Convergence	No Convergence

Table 8 Example Single Source to Single Destination - Ahn and Ramakrishna

Ahn & Ramakrishna Algorithm Path Traversed		
Source/Destination	**Path Taken**	**Cost(ρ)**
Node 1 to 3	1-> 2-> 3->	843
Node 1 to 5	1-> 5->	607
Node 1 to 9	1-> 2-> 3-> 8-> 9->	1749
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	No Convergence	No Convergence
Node 1 to 34	No Convergence	No Convergence

Table 9 Example Single Source to Single Destination - Park and Keum Algorithm

Park & Keum Algorithm Path Traversed		
Source/Destination	**Path Taken**	**Cost(φ)**
Node 1 to 3	1-> 2-> 3->	843
Node 1 to 5	1-> 5->	607
Node 1 to 9	1-> 7-> 10-> 12-> 8-> 9->	875
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	1-> 7-> 10-> 17-> 25-> 15-> 26-> 27-> 30->	1983
Node 1 to 34	1-> 7-> 10-> 17-> 25-> 15-> 26-> 27-> 34->	2311

3.2.3 Hopfield Neural Network Algorithm Convergence Rate – Single Destination Result

The following is a convergence rate result of the single destination. This result was produced from the application of Hopfield Neural Networks on the 20, 40 and 180 node topologies. These results were obtained from the Hopfield Neural Network application and were converted into percentage. Each of the result had different structure and path length with different link cost. The algorithm was said to be convergent if there existed a link between the given source and destination node, otherwise the algorithm was said to be non-convergent. The following series of tables will illustrate the different convergence rates in each topologies and also a total convergence rate percentage for every neural network.

Table 10 Convergence Rate - 20 Node Topology Single Destination

Convergence %	Convergent	Non Convergent
Ali & Kamoun	88%	12%
Park & Keum	99.8%	0.2%
Park & Choi	99.8%	0.2%
Ahn & Ramakrishna	99.8%	0.2%

Table 11 Convergence Rate - 40 Node Topology Single Destination

Convergence %	Convergent	Non Convergent
Ali & Kamoun	50%	50%
Park & Keum	99.8%	0.2%
Park & Choi	75%	25%
Ahn & Ramakrishna	75%	25%

Table 12 Convergence Rate - 180 Node Topology Single Destination

Convergence %	Convergent	Non Convergent
Ali & Kamoun	30%	70%
Park & Keum	99.8%	0.2%
Park & Choi	35%	65%
Ahn & Ramakrishna	35%	65%

The 180 node scenario for the single destination problem gives the same convergence rates as the 40 node topologies however it decreases drastically in convergence if longer paths are taken. This makes such 180 topology an impractical scenario to implement such single destination routing.

3.2.4 Hopfield Neural Network Path Optimality – Single-Destination Result

The total path optimality percentage is found as a result of comparison between each Hopfield Neural Network in the respective topologies of 20, 40 nodes and 180 Nodes. The best algorithm metric cost was listed for each result in each topology then the total optimal path was also found for the single destination. This result will be used in the observation and findings in the next section to show neural network scalability and performance for the single destination. The following table and chart represents the optimal path percentage.

Table 13 Optimal Path Percentage

Optimality Path %	20 Node Topology	40 Node Topology	180 Node Topology
Ali & Kamoun	88%	38%	28%
Park & Keum	75%	99.8%	99.8%
Park & Choi	88%	63%	54%
Ahn & Ramakrishna	75%	63%	48%

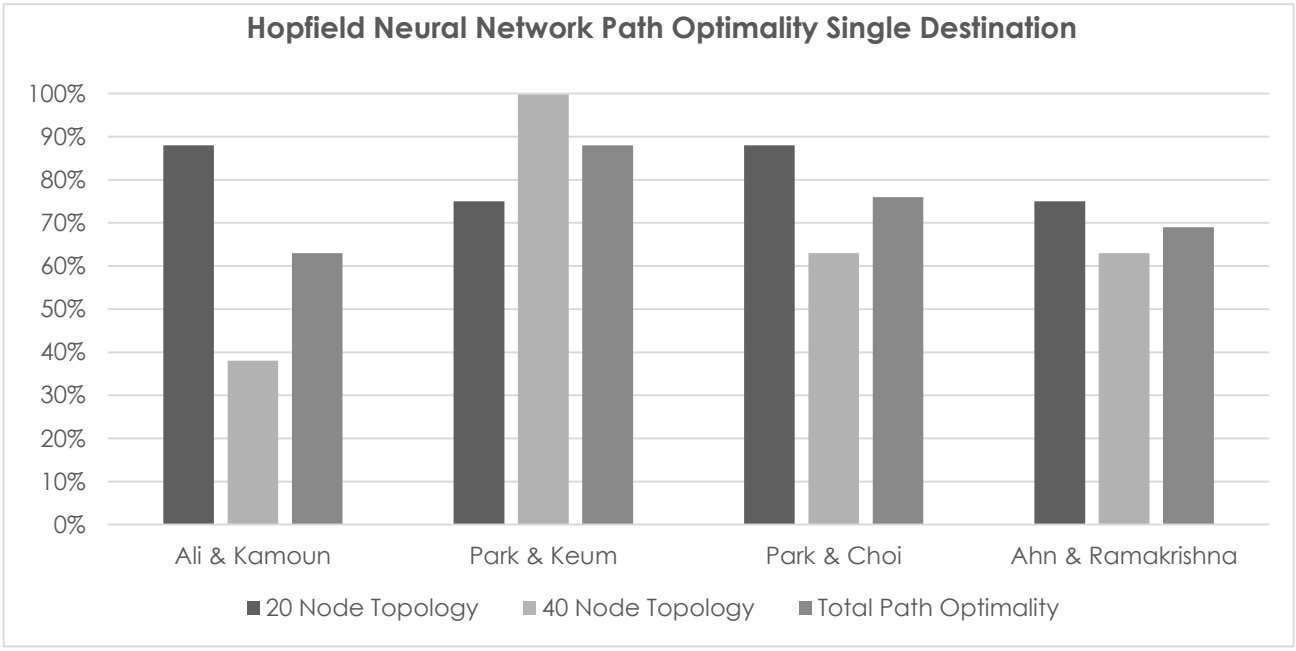


Figure 3 Hopfield Neural Network Path Optimality Single Destination

3.3 Multi-Destination Results

3.3.1 Non-Neural Convergent Algorithms - Multi-Destination

Using the same methodology the optimal path can be found as the shortest path is computed using the non-neural algorithms Dijkstra, Bellman-Ford and Floyd Warshall respectively, over the topologies, this was further done to each of the 120 generated results and thus this could lead in obtaining important results such as the optimal path percentage and also the convergence rate percentage. The following result is a sample that shows the optimal path with the metric cost for each non-neural network algorithm.

Table Error! Bookmark not defined. **Sample for a 40 Node Network - Single Source to Multiple Destinations Non-Neural**

Source to Multiple Destinations	**Dijkstra Algorithm**	**Bellman Ford Algorithm**	**Floyd Warshall Algorithm**
1 -> 23	1178	1178	1178
23 -> 38	1646	1746	1646
38 -> 11	1271	4083	1271
11 -> 9	1481	1641	1481
Total Cost For Multiple Destination	5576	8648	5576

The metric cost of these algorithms can then be compared to view the best optimal path and the best algorithm from both neural network and non-neural algorithms. This sample is shown in table 12 below.

Table 14 Sample for a 40 Node Network - Total Metric for Hopfield Neural Networks and Non-Neural Algorithms

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	5576	8648	5576	**4581**	**9345**	**8797**	–

3.3.2 Hopfield Neural Networks Convergence - Multi-Destination

The following is the sample result of a 40 nodal Topology using a Hopfield Neural Network approach which was taken from 120 tested results as a sample. A convergence chart is produced from a 10 random generated results obtained by using the Hopfield neural network approach as a percentage of convergence successes. The below result also show the optimal path with the metric cost for each neural network algorithm, this was done to all the other produced results in order to calculate both the convergence and optimal path.

Table 15 Sample for a 40 Node Network - Single Source to Multiple Destinations

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost	Metric
Park & Keum	**1**-> 7-> 10-> 17-> 25-> **23**-> 27 -> 30 -> **38** -> 33 -> 24 -> 15 -> **11** -> **9**	**4581**	
Park & Choi	**1**-> 6-> 13-> 18-> **23**-> 27 -> 28 -> 29 -> 32 -> 35 -> 36 -> 37 -> **38** -> 40 -> 33 -> 24 -> 15 -> **11** -> **9**	**9345**	
Ahn & Ramakrishna	**1**-> 6-> 13-> 18-> **23**-> 27 -> 28 -> 29 -> 32 -> 35 -> 36 -> 37 -> **38** -> 33 -> 24 -> 15 -> **11 -> 9**	**8797**	
Ali & Kamoun	No Convergence	–	

In this case the results show the path and metric cost of the Hopfield Neural Network Algorithms, each result is taken individually and can also be compared with each other, this is done further in the following section observations and findings.

3.3.3 Hopfield Neural Network Algorithm Convergence Rate – Multi-Destination

The convergence rate for all the topologies was calculated using 10 results for each and a total convergence rate was calculated using the total 30 generated results. Each of the result had different configurations with same link cost. The algorithm was said to be convergent if there existed a link between the given source and destination node. Otherwise the algorithm was said to be non-convergent. The following series of tables will illustrate the different convergence rates in each topologies and also a total convergence rate percentage for every neural network.

Table 16 20 Node Topology Convergence %

Convergence %	Convergent	Non Convergent
Ali & Kamoun	99.8%	0.2%
Park & Keum	90.0%	10.0%
Park & Choi	90.0%	10.0%
Ahn & Ramakrishna	90.0%	10.0%

Table 17 40 Node Topology Convergence %

Convergence %	Convergent	Non Convergent
Ali & Kamoun	20.0%	80.0%
Park & Keum	99.8%	0.2%
Park & Choi	90.0%	10.0%
Ahn & Ramakrishna	90.0%	10.0%

Table 18 180 Node Topology Convergence %

Convergence %	Convergent	Non Convergent
Ali & Kamoun	0.2%	99.8%
Park & Keum	99.8%	0.2%
Park & Choi	14.0%	86.0%
Ahn & Ramakrishna	41.0%	59.0%

These result show the main convergence rate of all the neural network algorithms within the topologies respectively, further to this the following results will show the total convergence performance for all the neural network algorithms when applied on the topologies as an overall result.

3.3.4 Hopfield Neural Network Path Optimality – Multi-Destination

The total path optimality percentage is found as a result of comparison between each Hopfield neural network in the respective topologies. The best algorithm metric cost was listed for each result in each topology then the total optimal path was also found by viewing all the topologies as a whole. This result will be used in the observation and findings in the next section to show neural network scalability and performance. The following table and chart represents the optimal path of neural networks in relation to the topology size and also shows the total optimal path which identifies the scalability of the Hopfield neural network algorithms.

Table 19 Hopfield Neural Network Path Optimality Multi Destination

Optimality Path %	20 Node Topology	40 Node Topology	180 Node Topology
Ali & Kamoun	80.0%	10.0%	14.0%
Park & Keum	60.0%	99.8%	85.0%
Park & Choi	60.0%	60.0%	0.2%
Ahn & Ramakrishna	20.0%	50.0%	14.0%

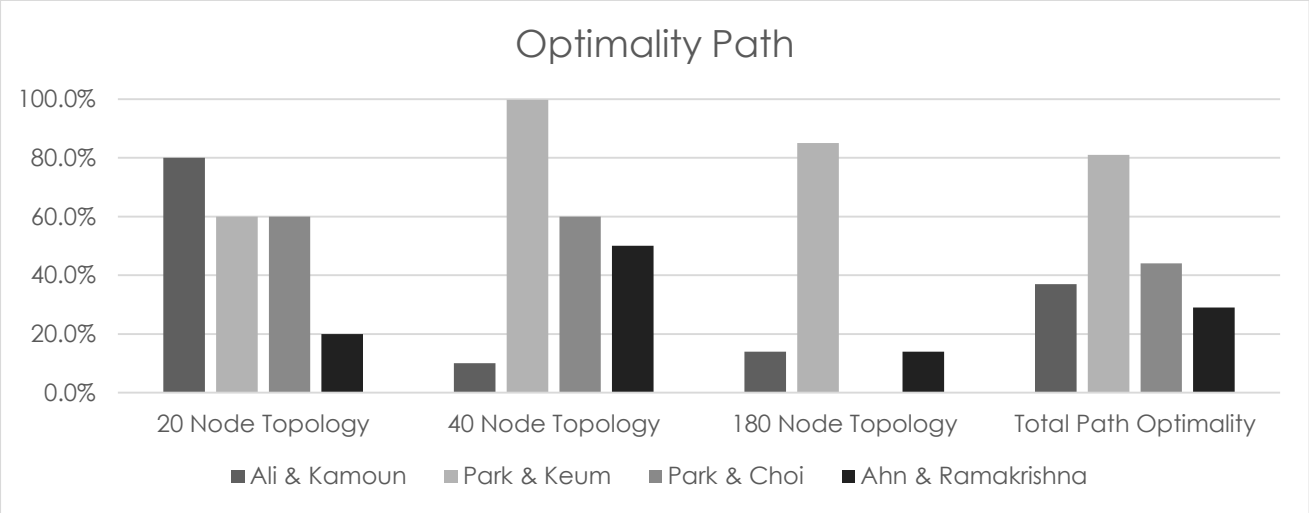


Figure 5 Optimality Path

Chapter 4 - Observations and Findings

4.1 Overview

This section includes the observations made on the findings presented in the result section. These observations give an insight on the convergence rate and also the comparison of results featuring the best non-neural and Hopfield neural network algorithm. Also this section presents observation and findings that identify that not only Hopfield Neural Networks can be applied, but that they can also perform well in the environments presented when compared to the present established non-neural algorithms in the Routing Information Protocol (RIP) and Open Shortest Path First (OSPF) protocols.

4.2 Non-Neural Algorithms Observations and Performance

Non-neural algorithms consisted of the Dijkstra, Bellman-Ford and Floyd-Warshall Algorithms. The algorithms were analyzed and compared on multiple terms, mostly on the best shortest path found with the least possible metric. The most reliable algorithms, Dijkstra and the Floyd-Warshall Algorithm proved to find the optimal path with the least possible cost metric in the conducted results.

This made both Dijkstra and Floyd-Warshall Algorithms to be better and more efficient than the Bellman-Ford Algorithm which is used by the Routing Information Protocol (RIP) in today's packet switched networks. This result was obtained by conducting a number of simulations using both single-destination paths and also multi-destination paths. Furthermore different topologies were used so that the result could be more realistic and in each conducted experiment both Dijkstra and Floyd-Warshall Algorithms proved to be better in finding the shortest path and the most optimal path with the least cost.

Using different topologies, an analysis was done on which non-neural algorithm would be best applied on small, medium and large topologies. The result was that in small topologies having not more than 20 nodes all the non-neural algorithms could be applied although applying Bellman-Ford would be a better candidate as the Routing Information Protocol (RIP) uses hop count which may resolve faster in smaller network topologies and such analysis shows that unlike other non-neural networks it does not scale well in large sized networks. Another observation is that as network topologies scale in size, having over 40 nodes(routers) within the topology environment both Dijkstra and Floyd-Warshall would be better candidates over the bellman-ford as they scale better, thus making the Open Shortest Path First (OSPF) protocol more suitable for larger networks.

This analysis shows that the Open Shortest Path First (OSPF) protocol works more efficiently and is more scalable when compared to the Routing Information Protocol (RIP) protocol as the underlying algorithm used in OSPF is more reliable. Also such analysis on the given result quantifies the metric costs used which enable better comparison of the non-neural algorithms.

This analysis of the non-neural algorithms shows that the non-neural algorithms Dijkstra and Floyd-Warshall are better candidates in large size networks although they can be applied on smaller sized size networks. On the other hand analysis further shows also that with the present result the Bellman-Ford is the least scalable and is to be used in smaller network sizes. Furthermore the analysis opened other comparisons of algorithmic nature with the Hopfield Neural Networks these will be shown in the next sections where non-neural algorithms are compared to neural algorithms to show application and optimality.

4.3 Hopfield Neural Network Algorithms Observations and Analysis

Hopfield Neural Network Algorithms consisted of Ali & Kamoun, Park & Keum, Park & Choi and Ahn & Ramakrishna. Each of the algorithms was tested on different topologies. Also the Hopfield Neural Network Algorithms were tested by having both a single-destination and multi-destination approach. The results enabled the comparison of the neural networks and their performance in various topologies as they were applied. Results were taken from multiple tests conducted specifically with different paths, random link costs and different topologies to eliminate bias and have a more realistic result in the simulation.

4.3.1 Convergence Issue and Local Minimum State of the Lyapunov Function

In the "single source to a single destination Results" the Hopfield Neural Networks Algorithms had a convergence issue as they won't converge if a considerable amount of nodes are added between the source and destination, this problem is not present in the multi-destination problem. This was shown in the various results conducted and such constituted of this convergence issue. This convergence issue was present in the Hopfield Neural Networks algorithms as they use the Lyapunov Function which with the increase of parameters or size of distance between a source and destination node can have the possibility of falling in what is known as the local minimum state which destabilize the equation making it vulnerable and prone to non-convergence. This was seen and analyzed in the results where the convergence rates of Hopfield neural network deteriorated into non-convergence states having minimal convergence a clear example of this is the Ali & Kamoun Algorithm that deteriorated from 88% to 50% as the topology changed for the single destination from a 20 node to a 40 node. This change occurred due to the stability of the parameters

which were set to hold a stable optimal path but could not be applied to larger scale networks as the convergence issue won't allow such algorithm to solve the single destination problem. Park & Choi and Ahn & Ramakrishna Algorithms also were slightly affected by the change in topology and effected in terms of convergence rates and optimal path discovery. The least affected algorithm was Park & Keum which did not suffer from instability with the given parameters and had very minimal convergence rates when applied on the single-destination problem.

In the multi-destination results, the non-convergence rate was very minimal as the path was distributed between multiple nodes (destinations) and so the problem of the actual routing was divided among the path itself. This enabled the algorithms to stabilize and not fall in the local minimum of the Lyapunov Function thus achieving more convergence. In the multi-destination routing problem it was discovered that Ali & Kamoun, Park & Choi and Ahn & Ramakrishna algorithms did not scale well to the largest topology that of 180 nodes.

The last observation was carried out on the 180 Node Scenario which was the largest simulate Local Area Network (LAN) environment. The observation for the multi-destination problem proved that the least convergent Hopfield neural network algorithms were Ali & Kamoun and Park & Choi this was expected as these were the least stable algorithms with the least convergence rates. The observation also suggested that the most reliable algorithms were Park & Keum and Ahn & Ramakrishna as these were more scalable.

This shows that these three Hopfield Neural Networks are best applied on small and medium networks ranging from 20 to 80 nodes after which fast degradation and non-convergence issues are potential. The Park & Keum Hopfield neural network algorithm on the other hand was successfully applied on different networks and had a 99.8 % success rate of convergence for the multi-destination routing problem.

4.3.2 Hopfield Neural Networks Algorithms Performance

This results analysis show that such algorithms can be applied on different topologies and can also be applied as required as Ali & Kamoun, Park & Choi and Ahn & Ramakrishna are more stable in smaller and medium topological structures of Local Area Networks while Park & Keum is better implemented over medium and larger topologies as it is more stable in that environment. The best algorithm to prove the best optimal paths overall was Park & Keum Algorithm which proved to have an 81% path optimality over all topologies and a 96% convergence rate including the larger topology.

The next best algorithm was Park & Choi with a total of 70% total convergence and a 44% path optimality. Park & Choi had a low path optimality as the 180 node topology was accounted for, and as the algorithm is not stable for such large topologies the degradation resulted into less path optimality, when considering this path optimality issue the algorithm performance can be increased to 60% if the topology structure is of not more than 40 nodes.

The other two Hopfield neural networks Ali & Kamoun and Ahn & Ramakrishna had 52% and 78% as an average rate of convergence respectively while having a 37% and 29% optimal path discovery rates respectively. The analysis of such results shows that both of these algorithms do not scale well as they will not converge and are not stable enough to be applied on larger networks but work well in smaller controlled environments suggested to have between 20 and 40 nodes.

Such Hopfield Neural Network observations and analysis show not only that these algorithms can be applied but also of yielding potential advantages over the non-neural algorithms in some cases. Such advantages include that such Hopfield Neural Network algorithms can be tuned to find the same or better optimal paths as the present non-neural algorithm, and under appropriate

assumptions such as the introduction of adaptive routing and learning these Hopfield neural networks might have an edge over the present algorithms, yielding a potential advantage.

Also such observation show that although such Hopfield neural network algorithms have convergence issues, if correctly applied, the convergence issues are minimized and with the correct configuration could be applied in network topologies.

4.4 Non-Neural and Hopfield Neural Network Algorithm Performance Analysis

These results analysis are important as they show the comparison of both the Hopfield neural networks and the non-neural algorithms and this gives an insight of the performance of both as applied in different scenarios.

The analysis shows that form the conducted experiments and simulation results in the 20 node Topology in the multi-destination problem, all the algorithms had good convergence and practically they virtually all converged to give an optimal path in the same range. However in many cases Ali & Kamoun, Park & Keum, Park & Choi and Ahn & Ramakrishna proved to perform better than the Bellman-Ford algorithm used by the Routing Information Protocol (RIP) protocol and also sometimes in a moderate way surpassed the optimal routing of the Dijkstra and Floyd-Warshall algorithms which also shows that the present used Open Shortest Path First OSPF protocol could be matched in terms of optimality and shortest path discovery by the Hopfield neural networks.

The next analysis was carried on the single-destination problem, environment of the 20 node scenario. The analysis shows that both Hopfield Neural Networks and Non-Neural algorithms were

virtually the same in terms of optimal metric cost and path discovery. The analysis conducted resulted in showing that in the 20 node infrastructure the algorithms performed the same.

Further analysis was carried out on another local area environment, this time the 40 node scenario results were analyzed. From such analysis it was deducted that in the single-destination problem, also both Hopfield neural networks and non-neural networks converged to give an optimal path similarly with nearly same metric cost in all conducted experiments. These results are summarized in table 9 in section 3.4.2.

Moreover, another analysis was conducted on the multi-destination problem, in the 40 node scenario. This result analysis shows a balance between Hopfield neural networks and non-neural network algorithms. Although the results show that Dijkstra and Floyd-Warshall Algorithms converged better in many cases these were matched by Park & Keum Hopfield neural algorithm in terms of optimality and convergence. Again Ali & Kamoun, Park & Keum, Park & Choi and Ahn & Ramakrishna Hopfield neural network algorithms also matched and exceeded the non-neural algorithm Bellman-Ford in many instances during the results. The least convergent algorithm was Ali & Kamoun in this scenario as it is the least stable from the algorithms with a 37% path optimality average and Park & Keum with an 81% convergence and a more success rate when implemented in such topology.

The last observation and finding was done for the 180 node scenario, as the Hopfield Neural Networks suffered most of the degradation in these results it was evident that only the Park & Keum algorithm was able to be applied and fully function in such scenario, this was also expected as this Hopfield neural network was more able to scale than the rest. Consequently when comparing such Hopfield neural network to the non-neural algorithms one could identify that the optimal path rate is also very similar with minor differences, this proves that neural networks not

only could be applied to multiple type scenarios but also that some of them, particularly Park & Keum could even scale to the point of reaching the required amount of nodes (routers) needed in larger scenarios.

The observations shows that in some cases Hopfield neural network exceed expectation and can be applied on various scenarios and environments. Moreover, these observations show that the Hopfield Neural Networks can be applied in topologies and exceed optimality in terms of shortest path findings.

While existing non-neural algorithms are good and offer scalability the benefit of using neural networks would increase with the assumption of having adaptive routing and learning can improve the present system.

This means that Hopfield Neural Networks can be implemented in nowadays network environments and offer the same mechanism of packet switching in terms of optimality while also exceeding them with the specific adaptation and introduction of learning and adaptive routing in such Local Area Network (LAN) environments.

4.5 Learning

An important observation that was part of this research was that under appropriate assumptions future implementation of learning to the Hopfield Neural Networks can be done. Assuming that the performance of the Hopfield neural networks would increase, which can be based and assumed by the present studies, these become perfect candidates to be applied over in a Local Area Environment within specific made protocols. This assumption would also lead improvements that can be done to such algorithms such as using reinforcement learning, supervised learning or unsupervised learning approaches, used already by vehicle routing, speech recognition and sequential decision making games, the Hopfield neural networks could be adapted to work more efficiently in terms of long-term cost and adapt their parameters to be more stable within the environments. This will produce better results as it will deviate from conventional networks that build rules around packets on various attributes and will instead make use of Hopfield Neural Networks which are capable to learn which makes them better than non-neural present algorithms as they would converge better with the ability to learn and adapt in the concurrent environments and achieving faster communication in computer networks. However this is only an assumption which is based on the present neural network learning research and also on the convergences, scalability and adaptability taken from this research as observation.

Hopfield Neural Networks were applied successfully within different scenarios, under different metrics, and algorithms such as Park and Keum also showed promising results which exceed expectation. These were Hopfield Neural networks were tested packet-switched communication networks that were created and under the assumption of having a Local Area Network (LAN) environments.

However, Hopfield neural networks solving single-destination routing problem have a convergence issue. The convergence issues occur when Hopfield Neural Networks did not converge and solve the shortest path problem when given a long path as the Energy function fell into a local minimum state preventing the shortest path to be computed by the Hopfield Neural Network algorithms.

Moreover, experiments show that Hopfield Neural Networks performed the same optimality, or even exceed, non-neural algorithms present in today's networks, including the Routing Information Protocol (RIP) and the Open Shortest Path First Protocol (OSPF) in various topologies were the Hopfield Algorithms converged.

Hopfield Neural Networks were identified to be an adequate solution to the existing shortest route problem as they are able to compute the shortest path and converge to the wanted path optimality in most of the topologies.

In the process of identifying that such neural networks can be applied to environments and solving the shortest route problem, non-neural algorithms Dijkstra, Bellman-Ford and Floyd-Warshall were also analyzed. This lead to the consequent comparison between the non-neural and neural

network algorithms, in a both single-destination and multi-destination environment using asymmetrical traffic, the results proved that neural networks can be applied within the scenarios.

As the network simulation scaled in terms of nodes a degradation in Hopfield Neural Networks was identified due to a convergence problem which occurs due to parametric values and function stability, nevertheless specific algorithms such as Park and Keum could scale to perform on large scale networks with a convergence rate of 99.8%, this was mostly dependent on the scenario simulated the 20 node, 40 node and 180 node respectively.

Another important assumption that could be observed from this research is that neural networks are capable to perform learning, which in other situations improves their overall results, this happens in sequential decision making and pattern recognition mostly. When applied to the Hopfield Neural Network, under the above assumptions, this would allow more benefits such as innovation and optimization. This is because it would improve neural networks to converge better and increase the optimality as required. This will deviate from conventional networks that build rules around packets on various attributes and will instead make use of Hopfield Neural Networks which are capable to learn and build rules as required on packets and not use the standard rules making Hopfield Neural Networks more beneficial to computer networks and achieving faster communications. This is an important assumption as it could shape the outcome of applying the Hopfield Neural Networks as with such learning such algorithms might be given a better edge and perform better.

The results obtained from the various simulations neural networks proved to be algorithms that can be used and adapted to networks and while applying them a new solution to solving the shortest path problem is achieved which drifts away from the traditional model of networking. This drift from the traditional model of networking has its advantages as it can, as seen in the obtained results,

compute the shortest path on specific metrics such as different traffic and result in faster networks and more flexibility as traffic conditions may rapidly vary in time.

However, the Hopfield Neural Networks also suffer from convergence issues as the energy function used by the algorithms can fall into a local minimum state where convergence is reduced and the shortest path could not be computed. This would create a problem of degradation in larger topologies. Such convergence and degradation issues will thus balance out the advantages and disadvantages of applying such Hopfield Neural Network Algorithms.

The research enables a means of further analysis and investigation on neural networks, as such research gives a strong ground to the idea of implementing neural networks within the present computer networks and also gives the opportunity of opening new fields in computer science, such as that of neural networking in computer networks and the application of neural networks to solve problems such as the shortest route problem.

Ali MM, K. F. e. a., n.d. *Neural networks for shortest path computation and routing in computer networks.*. s.l.:IEEE Trans Neural Netw. 1993;4(6):941-54..

Aliabadi, D. E., n.d. *Modeling Shortest Path Routing Problem withHopfield Neural Network.* s.l.: Sabanci University, Faculty of Engineering and Natural Science, Istanbul, Turkey.

Bellman, R., 1958. *On a Routing Problem.* s.l.:Quarterly of Applied Mathematics 16: 87–90..

Borundiya, A. P., March 2008. *Implementation of Hopfield Neural Network Using Double Gate MOSFET*. Ohio: Russ College of Engineering and Technology of Ohio University - Master of Science.

Brande, J. K., November 7, 1997. *Computer Network Routing with a Fuzzy Neural Network.* s.l.:Virginia Polytechnic Institute and State University.

C.W. Ahn, R. R. C. K., 2001. *IEE 2001 Electronics Letters 13th September2001 Vol. 37 No. 19.* s.l.:Department of Computer Engineering,Chalmers University of Technology, SE-412 96 Goteborg, Sweden.

Chang Wook Ahn, S. M. I. a. R. S. R. S. M. I., 1, JANUARY 2004. *QoS Provisioning Dynamic Connection-Admission Control for Multimedia Wireless Networks Using a Hopfield Neural Network.* s.l.:IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 53, NO. 1.

Choi, D.-C. P. a. S.-E., n.d. *A Neural Network Based Multi-Destination Routing Algorithm for Communication Network.* s.l.:School of Electrical and Electronic Engineering MyongJi University.

Dijkstra, E. W., 1959. *A Note on Two Problems in Connexion with Graphs.* s.l.:Numerische Mathematlk l, 269 - 27 I (l 959).

Fadil, Y. A., 21-6-2010. *ROUTING USING GENETIC ALGORITHM FOR LARGE NETWORKS.* s.l.:Engineering College , Diyala University..

Floyd, R. W., 1962. *Algorithm 97: Shortest path.* s.l.:Communications of the ACM Volume 5 Issue 6, June 1962.

Forouzan, B. A., 2013. *Data Communications and Networking.* ISBN 0-07-296775-7 - 5th Edition ed. s.l.:McGraw-HilI Forouzan networking series - Alan R. Apt.

Hart, P. N. N. R. B., n.d. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths.* s.l.:Artificial Intelligence Group of the Applied Physics Laboratory, Stanford Research Institute Menlo Park, Calif..

Haykin, S., 1999. *NEURAL NETWORKS - A Comprehensive Guide.* Second Edition ed. Hamilton, Ortario, Canada : McMaster University, Prentice Hall International, Inc..

Hedrick, C., June 1988. *Request for Comments: 1058 : Routing Information Protocol RFC.* s.l.: Rutgers University.

Jzau-Sheng Lin *, M. L. A. N.-F. H., April 21, 2000. *The Shortest-Path Computation in MOSPF Protocol through an Annealed Chaotic Neural Network.* s.l.:Department of Electronic Engineering National Chinyi Institute of Technology.

Keum, D.-C. P. a. K.-R., 2008. *A shortest path routing algorithm using Hopfield neural network with an improved energy function.* s.l.:Department of Information Engineering, Myongji University, Yongin, South Korea; bDepartment of Marine Diesel Engine Shop Test, STX Engine Co., Seoul, South Korea.

Konstantopoulos, S. F. a. T., n.d. *Lyapunov function methods.* s.l.:LMS/EPSRC Short Course Stability, Coupling Methods and Rare Events Heriot-Watt University, Edinburgh, 4-9 September 2006.

Korst, E. H. A. a. r. t. s. a. J. a. n. H., n.d. *Boltzmann Machines and Their Applications.* s.l.:Philips Research Laboratories JA Eindhoven, the Netherlands.

lehr, B. W. M. A., n.d. *Backpropagation and its Applications.* s.l.:Standford University, Department of Electrical Engineering.

MacKay, D. J., 2003. *Information Theory, Inference, and Learning Algorithms.* Cambridge: Cambridge University Press 2003..

Maity, P. S. R. P. S. S. S. S. c. S., 2013. *A COMPARATIVE STUDY OF VARIOUS METHODS OF ANN FOR SOLVING TSP PROBLEM.* Kolkata, West Bengal, India: Council for Innovative Research - International Journal of Computers & Technology.

Neapolitan, R. E., n.d. *Learning Bayesian Networks.* s.l.:Northeastern Illinois University.

Nenad Koji´c, I. R. a. B. R., 2006. *Neural Network for Optimization of Routing in Communication Networks.* s.l.:FACTA UNIVERSITATIS (NI˙S).

Newton, W., May 5, 2002. *A Neural Network Algorithm for Internetwork.* s.l.:Bachelor of Engineering with Honours in Software Engineering.

Pallapa Venkataram*, S. G. B. V. K., 17 May 2002. *Neural network based optimal routing algorithm.* s.l.:Protocol Engineering and Technology (PET) unit, Department of Electrical Communication Engineering, Indian Institute of Science,.

Prince+, N. N., 2011. *Real – World Applications of Neural Network.* s.l.:Department of Computer Engineering, Federal Polytechnic, Oko, Anambra State, Nigeria. (2011) IACSIT Press, Singapore.

Ravindra K. Ahuja*, T. L. M. a. J. B. O., n.d. *NETWORK FLOWS.* Cambridge, MA. 02139: Massachusetts Institute of Technology and Sloan School of Management.

Rojas, R., 1996. *Neural Networks A Systematic Introduction.* R. Rojas: Neural Networks, Springer-Verlag, Berlin, 1996 ed. Springer-Verlag, Berlin, 1996: s.n.

Ruohonen, K., 2013. *GRAPH THEORY.* s.l.:Keijo Ruohonen.

Sarika Tyagi, S. N. S., June, 2013. *SHORTEST PATH USING NEURAL NETWORK.* Meerut, (India): 1CS, RKGITW, Ghaziabad, (India), 2CSE, SITE, Meerut, (India).

University, S., 2008-2009. *EE363 - Basic Lyapunov theory (Lecture 12).* s.l.:Stanford University .

Vella, A. & Vella, C., 2009. *Neural networks.* Malta: Goldsmith, University of London.

Wilamowski, B. M., 2003. *Neural Network Architectures and Learning.* Maribor Slovebnia: Fellow Member IEE and Auburn University USA.

Yip, P. a. P. Y.-H., 1995. *Combinatorial optimization with use of guided evolutionarysimulated annealing. IEEE transactions on neural networks, 6 (2), 290–295.*. s.l.:IEEE transactions on neural networks, 6 (2), 290–295..

Appendix 1 – Results Generated

The following are the 120 generated result to achieve a better and more non-bias result. These were the results that were referred to in Chapter 3 in the Experimentation and Results Section.

Data Analysis - Sample Data Results for a **20 Node Network - Single Source to Single Destination**

Hopfield Neural Networks Convergent Algorithms

Ali & Kamoun Algorithm Path Traversed

Source/Destination	Path Taken	Cost(α)
Node 1 to 3	1-> 2-> 3->	61
Node 1 to 5	1-> 2-> 3-> 4-> 5->	85
Node 1 to 6	1-> 2-> 8-> 7-> 6->	82
Node 1 to 9	1-> 10-> 9->	23
Node 1 to 12	1-> 10-> 11-> 12->	65
Node 1 to 15	No Convergence	-
Node 1 to 18	1-> 10-> 11-> 12-> 13-> 18->	136
Node 1 to 20	1-> 10-> 11-> 20->	57

Park & Choi Algorithm Path Traversed

Source/Destination	Path Taken	Cost(β)
Node 1 to 3	1-> 2-> 3->	61
Node 1 to 5	1-> 2-> 3-> 4-> 5->	85
Node 1 to 6	1-> 2-> 3-> 4-> 5-> 6->	116
Node 1 to 9	1-> 10-> 9->	23
Node 1 to 12	1-> 10-> 11-> 12->	65
Node 1 to 15	1-> 10-> 11-> 12-> 13-> 14-> 15->	145
Node 1 to 18	1-> 10-> 11-> 12-> 13-> 18->	136
Node 1 to 20	1-> 10-> 11-> 20->	57

Ahn & Ramakrishna Algorithm Path Traversed

Source/Destination	Path Taken	Cost(ρ)
Node 1 to 3	1-> 2-> 3->	61
Node 1 to 5	1-> 10-> 11-> 12-> 13-> 14-> 15-> 6-> 5->	227
Node 1 to 6	1-> 10-> 11-> 12-> 13-> 14-> 15-> 6->	187
Node 1 to 9	1-> 10-> 9->	23
Node 1 to 12	1-> 10-> 11-> 12->	65
Node 1 to 15	1-> 10-> 11-> 12-> 13-> 14-> 15->	145
Node 1 to 18	1-> 10-> 11-> 12-> 13-> 18->	136
Node 1 to 20	1-> 10-> 11-> 20->	57

Park & Keum Algorithm Path Traversed		
Source/Destination	**Path Taken**	**Cost(ϕ)**
Node 1 to 3	1-> 2-> 3->	61
Node 1 to 5	1-> 10-> 11-> 12-> 13-> 14-> 15-> 6-> 5->	227
Node 1 to 6	1-> 10-> 11-> 12-> 13-> 14-> 15-> 6->	187
Node 1 to 9	1-> 10-> 9->	23
Node 1 to 12	1-> 10-> 11-> 12->	65
Node 1 to 15	1-> 10-> 11-> 12-> 13-> 14-> 15->	145
Node 1 to 18	1-> 10-> 11-> 12-> 13-> 18->	136
Node 1 to 20	1-> 10-> 11-> 20->	57

Non-Neural Convergent Algorithms

Dijkstra Path Traversed		
Source/Destination	**Path Taken**	**Cost(σ)**
Node 1 to 3	1-> 2-> 3->	61
Node 1 to 5	1-> 2-> 8-> 7-> 4-> 5->	85
Node 1 to 6	1-> 2-> 8-> 7-> 6->	82
Node 1 to 9	1-> 10-> 9->	23
Node 1 to 12	1-> 10-> 11-> 12->	65
Node 1 to 15	1-> 2-> 8-> 7-> 14-> 15->	103
Node 1 to 18	1-> 10-> 11-> 12-> 19-> 18->	111
Node 1 to 20	1-> 10-> 11-> 20->	57

Bellman Ford Path Traversed		
Source/Destination	**Path Taken**	**Cost(ω)**
Node 1 to 3		61
Node 1 to 5		85
Node 1 to 6		187
Node 1 to 9		23
Node 1 to 12		96
Node 1 to 15		151
Node 1 to 18		136
Node 1 to 20		64

Floyd Warshall Path Traversed	
Source/Destination	**Cost(τ)**
Node 1 to 3	61
Node 1 to 5	85
Node 1 to 6	82
Node 1 to 9	23
Node 1 to 12	65
Node 1 to 15	103
Node 1 to 18	111
Node 1 to 20	57

Total Results for 40 Node Network - Single Source to Single Destination

Source/Destination	Cost(α)	Cost(β)	Cost(ρ)	Cost(ϕ)	Cost(σ)	Cost(ω)	Cost(τ)
Node 1 to 3	61	61	61	61	61	61	61
Node 1 to 5	85	85	227	227	85	85	85
Node 1 to 6	82	116	187	187	82	187	82
Node 1 to 9	23	23	23	23	23	23	23
Node 1 to 12	65	65	65	65	65	96	65
Node 1 to 15	-	145	145	145	103	151	103
Node 1 to 18	136	136	136	136	111	136	111
Node 1 to 20	57	57	57	57	57	64	57

Data Analysis - Sample Data Results for a **20 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	1-> 11-> 12-> 8-> 4-> 7 -> 14 -> 15 -> 17 -> 18 -> 19 -> 12 -> 8 ->	**351**
Park & Choi	1-> 11-> 12-> 8-> 4-> 7 -> 14 -> 15 -> 17 -> 18 -> 19 -> 12 -> 8 ->	**351**
Ahn & Ramakrishna	1-> 2-> 3-> 4-> 14-> 15-> 17-> 18-> 19-> 20-> 5 -> 6 -> 15 -> 17 -> 18 -> 19 -> 20 -> 11 -> 12 -> 13 -> 14 -> 7 -> 8 ->	**-**
Ali & Kamoun	1-> 2-> 3-> 4-> 7 -> 14 -> 15 -> 17 -> 18 -> 19 -> 12 -> 8 ->	**301**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1-> 4	64	64	64
4-> 15	60	162	60
15-> 19	96	165	96
19-> 8	81	81	81
Total Cost For Multiple Destination	**301**	**472**	**301**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	301	472	301	**351**	**301**	**351**	**-**

Data Analysis - Sample Data Results for a **20 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	1-> 11-> 12-> 8-> 7 -> 14 -> 15 -> 16 -> 15 -> 17 -> 18 -> 19 -> 20	**291**
Park & Choi	1-> 11-> 12-> 8-> 12 -> 13 -> 14 -> 15 -> 16 -> 15 -> 17 -> 18 -> 19 -> 20 ->	**364**
Ahn & Ramakrishna	No Convergence	**–**
Ali & Kamoun	1-> 2-> 8-> 12 -> 13 -> 14 -> 15 -> 16 -> 15 -> 17 -> 18 -> 19 -> 20 ->	**334**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1-> 8	47	47	47
8-> 6	35	35	35
6-> 17	86	86	86
17-> 20	90	193	90
Total Cost For Multiple Destination	**233**	**361**	**233**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	233	361	233	**291**	**334**	**364**	**–**

Data Analysis - Sample Data Results for a **20 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	1-> 11-> 12 -> 13 -> 18 -> 17 -> 15 -> 6 -> 5	**219**
Park & Choi	1-> 11-> 12 -> 13 -> 18 -> 17 -> 15 -> 6 -> 5	**219**
Ahn & Ramakrishna	1-> 11-> 20 -> 19 -> 18 -> 19 -> 20 -> 11 -> 12 -> 13 -> 14 -> 15 -> 6 -> 5	**361**
Ali & Kamoun	1-> 11-> 12 -> 13 -> 18 -> 13 -> 14 -> 15 -> 6 -> 5 -	**264**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1-> 11	41	41	41
11 -> 18	70	70	70
18 -> 15	40	40	40
15 -> 5	82	82	82
Total Cost For Multiple Destination	**233**	**233**	**233**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	233	233	233	**219**	**264**	**219**	**361**

Data Analysis - Sample Data Result for a **20 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	16-> 15-> 6-> 5-> 11 -> 20 -> 19 -> 18 -> 17	**214**
Park & Choi	16-> 15-> 6-> 5-> 11 -> 20 -> 19 -> 18 -> 17	**214**
Ahn & Ramakrishna	16-> 15-> 6-> 5-> 11 -> 20 -> 19 -> 18 -> 17	**214**
Ali & Kamoun	16-> 15-> 6-> 5-> 11 -> 20 -> 19 -> 18 -> 17	**214**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
16 -> 5	110	110	110
5 -> 1	78	181	78
1 -> 20	57	64	57
20 -> 17	86	127	86
Total Cost For Multiple Destination	**331**	**482**	**331**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	**331**	482	**331**	**214**	**214**	**214**	**214**

Data Analysis - Sample Data Result for a **20 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	17-> 18-> 19-> 18 -> 19 -> 20 -> 11 -> 10 -> 11 -> 1 -> 2 -> 3 ->	**348**
Park & Choi	17-> 18-> 19-> 18 -> 19 -> 20 -> 11 -> 10 -> 11 -> 1 -> 2 -> 3 ->	**348**
Ahn & Ramakrishna	17-> 18-> 19-> 18 -> 19 -> 20 -> 11 -> 10 -> 11 -> 1 -> 2 -> 3 ->	**348**
Ali & Kamoun	17-> 18-> 19-> 18 -> 19 -> 20 -> 11 -> 10 -> 11 -> 1 -> 2 -> 3 ->	**348**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
17 -> 19	57	179	57
19-> 8	81	135	81
8-> 10	45	88	45
10-> 3	93	107	93
Total Cost For Multiple Destination	**276**	**509**	**276**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	**276**	**509**	**276**	**348**	**348**	**348**	**348**

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	8-> 9-> 10-> 11 -> 20 -> 19 -> 18 -> 13 -> 14	**207**
Park & Choi	8-> 9-> 10-> 11 -> 20 -> 19 -> 18 -> 13 -> 14	**207**
Ahn & Ramakrishna	8-> 9-> 10-> 11 -> 20 -> 19 -> 18 -> 19 -> 20 -> 11 -> 12 -> 8 -> 7 -> 14	**349**
Ali & Kamoun	8-> 9-> 10-> 11 -> 20 -> 19 -> 18 -> 13 -> 14	**207**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
8-> 10	45	88	45
10-> 20	53	60	53
20-> 18	60	101	60
18 -> 14	46	46	46
Total Cost For Multiple Destination	**204**	**295**	**204**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	**204**	**295**	**204**	**207**	**207**	**207**	**349**

Data Analysis - Sample Data Result for a **20 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	4-> 8-> 12-> 19-> 18 -> 17 -> 18 -> 13 -> 14 -> 7 -> 6 -> 7 -> 8 -> 9	**376**
Park & Choi	4-> 8-> 12-> 19-> 18 -> 17 -> 15 -> 6 -> 7 -> 8 -> 9	**305**
Ahn & Ramakrishna	4-> 8-> 12-> 19-> 18 -> 17 -> 18 -> 19 -> 20 -> 11 -> 12 -> 8 -> 4 -> 5 -> 6 -> 7 -> 8 -> 9	**536**
Ali & Kamoun	4-> 8-> 12-> 19-> 18 -> 17 -> 15 -> 6 -> 7 -> 8 -> 9	**305**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
4-> 19	101	101	101
19-> 17	53	53	53
17-> 6	56	56	56
6-> 9	103	103	103
Total Cost For Multiple Destination	**313**	**313**	**313**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	**313**	**313**	**313**	**376**	**305**	**305**	**536**

Data Analysis - Sample Data for a 2**0 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	1-> 11-> 12-> 13-> 14-> 7-> 14 -> 15 -> 17 -> 18 -> 19 -> 20 -> 11 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 6	**500**
Park & Choi	1-> 11-> 12-> 13-> 14-> 7-> 14 -> 15 -> 17 -> 18 -> 19 -> 20 -> 11 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 6	**500**
Ahn & Ramakrishna	1-> 11-> 12-> 13-> 14-> 7-> 14 -> 15 -> 17 -> 18 -> 19 -> 20 -> 11 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 6	**500**
Ali & Kamoun	1-> 2-> 8-> 7-> 14 -> 15 -> 17 -> 18 -> 19 -> 20 -> 11 -> 10 -> 11 -> 12 -> 13 -> 14 -> 15 -> 6	**466**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1-> 7	51	51	51
7-> 18	104	166	104
18-> 10	80	120	80
10 -> 6	82	183	82
Total Cost For Multiple Destination	**318**	**520**	**318**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	**318**	**520**	**318**	**500**	**466**	**500**	**500**

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	5-> 6-> 7-> 14-> 13-> 11 -> 20 -> 19 -> 18 -> 17	**228**
Park & Choi	5-> 6-> 7-> 14-> 13-> 11 -> 20 -> 19 -> 18 -> 17	**228**
Ahn & Ramakrishna	5-> 6-> 15-> 17-> 18-> 13-> 11 -> 20 -> 19 -> 18 -> 17	**262**
Ali & Kamoun	5-> 6-> 7-> 14-> 13-> 11 -> 20 -> 19 -> 18 -> 17	**228**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
5 -> 13	80	121	80
13-> 1	96	204	96
1-> 20	57	64	57
20 -> 17	86	127	86
Total Cost For Multiple Destination	**319**	**516**	**319**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	**319**	**516**	**319**	**228**	**228**	**228**	**262**

Data Analysis - Sample Data Result for a **20 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	No Convergence	**–**
Park & Choi	No Convergence	**–**
Ahn & Ramakrishna	16-> 15-> 6-> 5-> 6 -> 15 -> 17 -> 18 -> 13 -> 18 -> 19 -> 20 -> 11 -> 1 -> 2 -> 8 -> 9 -> 10 -> 11 -> 20	**602**
Ali & Kamoun	16-> 15-> 6-> 5-> 6 -> 7 -> 14 -> 13 -> 12 -> 8 -> 2 -> 8 -> 12 -> 19 -> 20	**452**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
16 -> 5	110	110	110
5-> 13	80	121	80
13-> 2	107	109	107
2-> 20	59	66	59
Total Cost For Multiple Destination	**356**	**406**	**356**

Total Results for 20 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Ali & Kamoun**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	**356**	**406**	**356**	**–**	**452**	**–**	**602**

The results continue with the 40 Node Scenario Generated Results:

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Single Destination**

Hopfield Neural Networks Convergent Algorithms

Ali & Kamoun Algorithm Path Traversed

Source/Destination	Path Taken	Cost(α)
Node 1 to 3	1-> 2-> 3->	843
Node 1 to 5	1-> 5->	607
Node 1 to 9	1-> 7-> 10-> 12-> 8-> 9->	1749
Node 1 to 15	No Convergence	No Convergence
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	No Convergence	No Convergence
Node 1 to 30	No Convergence	No Convergence
Node 1 to 34	No Convergence	No Convergence

Park & Choi Algorithm Path Traversed

Source/Destination	Path Taken	Cost(β)
Node 1 to 3	1-> 2-> 3->	843
Node 1 to 5	1-> 5->	607
Node 1 to 9	1-> 2-> 3-> 8-> 9->	1749
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	No Convergence	No Convergence
Node 1 to 34	No Convergence	No Convergence

Ahn & Ramakrishna Algorithm Path Traversed

Source/Destination	Path Taken	Cost(ρ)
Node 1 to 3	1-> 2-> 3->	843
Node 1 to 5	1-> 5->	607
Node 1 to 9	1-> 2-> 3-> 8-> 9->	1749
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	No Convergence	No Convergence
Node 1 to 34	No Convergence	No Convergence

Park & Keum Algorithm Path Traversed		
Source/Destination	**Path Taken**	**Cost(ϕ)**
Node 1 to 3	1-> 2-> 3->	843
Node 1 to 5	1-> 5->	607
Node 1 to 9	1-> 7-> 10-> 12-> 8-> 9->	875
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	1-> 7-> 10-> 17-> 25-> 15-> 26-> 27-> 30->	1983
Node 1 to 34	1-> 7-> 10-> 17-> 25-> 15-> 26-> 27-> 34->	2311

Non-Neural Convergent Algorithms

Dijkstra Path Traversed		
Source/Destination	**Path Taken**	**Cost(σ)**
Node 1 to 3	1-> 7-> 3->	676
Node 1 to 5	1-> 5->	607
Node 1 to 9	1->7->10->8->9->	875
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 7-> 10-> 17-> 18-> 22-> 20	1420
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	1-> 7-> 10-> 17-> 25-> 15-> 26-> 27-> 30->	1983
Node 1 to 34	1-> 7-> 10-> 17-> 25-> 15-> 26-> 27-> 34->	2311

Bellman Ford Path Traversed		
Source/Destination	**Path Taken**	**Cost(ω)**
Node 1 to 3	1-> 7-> 3->	699
Node 1 to 5	1-> 5->	607
Node 1 to 9	1->7->10->12->9->	1264
Node 1 to 15	1-> 7-> 10-> 17-> 25-> 15->	839
Node 1 to 20	1-> 5-> 14-> 20->	2026
Node 1 to 25	1-> 7-> 10-> 17-> 25->	633
Node 1 to 30	1->7->10->17->25->15->26->24->30->	2603
Node 1 to 34	1->7->10->17->18->29->32->31->34->	3404

Floyd Warshall Path Traversed	
Source/Destination	**Cost(τ)**
Node 1 to 3	1457
Node 1 to 5	690
Node 1 to 9	366
Node 1 to 15	837
Node 1 to 20	1421
Node 1 to 25	870
Node 1 to 30	1595
Node 1 to 34	1910

Total Results for 40 Node Network - Single Source to Single Destination

Source/Destination	Cost(α)	Cost(β)	Cost(ρ)	Cost(ϕ)	Cost(σ)	Cost(ω)	Cost(τ)
Node 1 to 3	843	843	843	843	676	699	1457
Node 1 to 5	607	607	607	607	607	607	690
Node 1 to 9	1749	1749	1749	875	875	1264	366
Node 1 to 15	-	839	839	839	839	839	837
Node 1 to 20	2026	2026	2026	2026	1420	2026	1421
Node 1 to 25	-	633	633	633	633	633	870
Node 1 to 30	-	-	-	1983	1983	2603	1595
Node 1 to 34	-	-	-	2311	2311	3404	1910

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Cost
Park & Keum	**1**-> 7-> 10-> 12-> **8**-> 12 -> **16** -> 12 -> 17 -> 18 -> 29 -> **32** -> 35 -> 36 -> 37 -> 38 -> **40** ->	**6006**
Park & Choi	**1**-> 2-> 3-> **8**-> 12 -> **16** -> 15 -> 26 -> 27 -> 31 -> **32** -> 35 -> 36 -> 37 -> 38 -> **40** ->	**7917**
Ahn & Ramakrishna	**1**-> 2-> 3-> **8**-> 12 -> **16** -> 15 -> 26 -> 27 -> 28 -> 29 -> **32** -> 35 -> 36 -> 37 -> 39 -> **40** ->	**8242**
Ali & Kamoun	Ali & Kamoun Algorithm does not converge in this condition	**––**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1-> 8	693	693	693
8-> 16	850	850	850
16-> 32	1530	4717	1530
32-> 40	351	2315	351
Total Cost For Multiple Destination	**3424**	**8575**	**3424**

Total Results for 40 Node Network - Single Source to Single Destination

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**
Total Cost For Multiple Destination	3424	8575	3424	**6006**	**7917**	**8242**

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Metric Cost
Park & Keum	**1**-> 7-> **10**-> 12 -> **16** -> 15 -> **24** -> 26 -> **27**	**4191**
Park & Choi	**1**-> 7-> **10**-> 12 -> **16** -> 15 -> **24** -> 26 -> **27**	**4191**
Ahn & Ramakrishna	**1**-> 7-> **10**-> 12 -> **16** -> 15 -> **24** -> 26 -> **27**	**4191**
Ali & Kamoun	**1**-> 7-> **10**-> 12 -> **16** -> 15 -> **24** -> 26 -> **27**	**4191**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1 -> 10	366	366	366
10 -> 16	471	471	471
16 -> 24	1639	1639	1639
24 -> 27	1002	1002	1002
Total Cost For Multiple Destination	**3478**	**3478**	**3478**

Total Results for 40 Node Network - Single Source to Single Destination Metric Cost

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	**3478**	**3478**	**3478**	**4191**	**4191**	**4191**	**4191**

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Metric Cost
Park & Keum	**1**-> 2-> **3**-> 8 -> **12** -> 8 -> **9** -> 12 -> **17**	**2748**
Park & Choi	**1**-> 2-> **3**-> 8 -> **12** -> **9** -> 12 -> **17**	**3137**
Ahn & Ramakrishna	**1**-> 2-> **3**-> 8 -> **12** -> **9** -> 12 -> **17**	**3137**
Ali & Kamoun	**1**-> 2-> **3**-> 8 -> **12** -> **9** -> 12 -> **17**	**3137**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1 -> 3	676	843	676
3 -> 12	721	721	721
12 -> 9	366	1061	366
9 ->1	373	373	373
Total Cost For Multiple Destination	**2136**	**2998**	**2136**

Total Results for 40 Node Network - Single Source to Single Destination Metric Cost

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	**2136**	**2998**	**2136**	**2748**	**3137**	**3137**	**3137**

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Metric Cost
Park & Keum	**1**-> 5-> **14**-> 20 -> 21 -> **35** -> 36 -> **37 -> 39**	**3828**
Park & Choi	**1**-> 5-> **14**-> 20 -> 21 -> **35** -> 36 -> **37 -> 39**	**3828**
Ahn & Ramakrishna	**1**-> 5-> **14**-> 20 -> 21 -> **35** -> 36 -> **37 -> 39**	**3828**
Ali & Kamoun	No Convergence	**–**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1 -> 14	1338	1338	1338
14 -> 35	1294	1294	1294
35 -> 37	528	1191	528
37 ->39	6	6	6
Total Cost For Multiple Destination	**3166**	**3829**	**3166**

Total Results for 40 Node Network - Single Source to Single Destination Metric Cost

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	**3166**	**3829**	**3166**	**3828**	**3828**	**3828**	–

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Metric Cost
Park & Keum	**1**-> 7-> **10**-> 6 -> **14** -> 19 -> 18 -> **23 -> 27**	**4870**
Park & Choi	**1**-> 7-> **10**-> 6 -> **14** -> 19 -> 18 **-> 23 -> 27**	**4870**
Ahn & Ramakrishna	**1**-> 7-> **10**-> 6 -> **14** -> 19 -> 18 -> **23 -> 27**	**4870**
Ali & Kamoun	No Convergence	**–**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1 -> 10	366	366	366
10 -> 14	1145	1466	1145
14 -> 23	1962	2156	1962
23 ->27	782	882	782
Total Cost For Multiple Destination	**4255**	**4870**	**4255**

Total Results for 40 Node Network - Single Source to Single Destination Metric Cost

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	**4255**	**4870**	**4255**	**4870**	**4870**	**–**	**4870**

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Metric Cost
Park & Keum	**1**-> **7**-> 10 -> 17 -> 25 -> **15** -> 24 -> **33**	**2333**
Park & Choi	**1**-> **7**-> 10 -> 17 -> 25 -> **15** -> 24 -> **33**	**2333**
Ahn & Ramakrishna	**1**-> **7**-> 10 -> 17 -> 25 -> **15** -> 24 -> **33**	**2333**
Ali & Kamoun	No Convergence	**–**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1 -> 10	67	67	67
10 -> 14	383	383	383
14 -> 23	389	389	389
23 ->27	1494	1494	1494
Total Cost For Multiple Destination	**2333**	**2333**	**2333**

Total Results for 40 Node Network - Single Source to Single Destination Metric Cost

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	**2333**	**2333**	**2333**	**2333**	**2333**	**2333**	–

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Metric Cost
Park & Keum	**1**-> 6-> **13**-> 18 -> **29** -> 32 -> **35** -> 36 -> **37**	**4221**
Park & Choi	**1**-> 6-> **13**-> 18 -> **29** -> 32 -> **35** -> 36 -> **37**	**4221**
Ahn & Ramakrishna	**1**-> 6-> **13**-> 18 -> **29** -> 32 -> **35** -> 36 -> **37**	**4221**
Ali & Kamoun	**1**-> 6-> **1**3-> 18 -> 23 -> 27 -> 28 -> **29** -> 32 -> **35** -> 39 -> 40 -> **37**	**7341**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1 -> 13	1237	1237	1237
13 -> 29	599	599	599
29 -> 35	827	827	827
35 ->37	528	1191	528
Total Cost For Multiple Destination	**3191**	**3854**	**3191**

Total Results for 40 Node Network - Single Source to Single Destination Metric Cost

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	**3191**	**3854**	**3191**	**4221**	**4221**	**4221**	**7341**

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Metric Cost
Park & Keum	1-> 6-> 13-> 18 -> 22 -> 21 -> 35 -> 36 -> 37 -> 39	**3973**
Park & Choi	1-> 6-> 13-> 18 -> 22 -> 21 -> 35 -> 36 -> 37 -> 39	**3973**
Ahn & Ramakrishna	1-> 6-> 13-> 18 -> 22 -> 21 -> 32 -> 35 -> 36 -> 37 -> 39	**4386**
Ali & Kamoun	No Convergence	**–**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1 -> 13	1237	1237	1237
13 -> 22	287	287	287
22 -> 36	1106	1587	1106
36 ->39	471	471	471
Total Cost For Multiple Destination	**3101**	**3582**	**3101**

Total Results for 40 Node Network - Single Source to Single Destination Metric Cost

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	**3101**	**3582**	**3101**	**3973**	**3973**	**4386**	**–**

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Metric Cost
Park & Keum	**1**-> 7-> 10-> 17-> 25-> **23**-> 27 -> 30 -> **38** -> 33 -> 24 -> 15 -> **11** -> **9**	**4581**
Park & Choi	**1**-> 6-> 13-> 18-> **23**-> 27 -> 28 -> 29 -> 32 -> 35 -> 36 -> 37 -> **38** -> 40 -> 33 -> 24 -> 15 -> **11** -> **9**	**9345**
Ahn & Ramakrishna	**1**-> 6-> 13-> 18-> **23**-> 27 -> 28 -> 29 -> 32 -> 35 -> 36 -> 37 -> **38** -> 33 -> 24 -> 15 -> **11 -> 9**	**8797**
Ali & Kamoun	No Convergence	**–**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1 -> 23	1178	1178	1178
23 -> 38	1646	1746	1646
38 -> 11	1271	4083	1271
11 -> 9	1481	1641	1481
Total Cost For Multiple Destination	**5576**	**8648**	**5576**

Total Results for 40 Node Network - Single Source to Single Destination Metric Cost

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	**5576**	**8648**	**5576**	**4581**	**9345**	**8797**	**–**

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Hopfield Neural Networks	Source to Multiple Destinations	Total Metric Cost
Park & Keum	**1**-> 5-> 14-> 20-> 21-> **35**-> 32 -> 31 -> **34** -> 27 -> 23 -> 25 -> 16 -> 12 -> **8** -> 12 -> 17 -> **18**	**6833**
Park & Choi	No Convergence	**–**
Ahn & Ramakrishna	No Convergence	**–**
Ali & Kamoun	No Convergence	**–**

Non-Neural Convergent Algorithms

Source to Multiple Destinations	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
1 -> 35	1910	2632	1910
35 -> 34	1363	2182	1363
34 -> 8	1604	4251	1604
8 -> 18	1065	1065	1065
Total Cost For Multiple Destination	**5942**	**10130**	**5942**

Total Results for 40 Node Network - Single Source to Single Destination Metric Cost

	Dijkstra	Bellman Ford	Floyd Warshall	**Park & Keum**	**Park & Choi**	**Ahn & Ramakrishna**	**Ali & Kamoun**
Total Metric Cost For Multiple Destination	**5942**	**10130**	**5942**	**6833**	**–**	**–**	**–**

Now the next Results presented will be the 180 Scenario Generated Results:

Data Analysis - Sample Data Result for a **180 Node Network - Single Source to Multiple Destinations**

Hopfield Neural Networks Convergent Algorithms

Result 1

Path : 1 -> 4 -> 8 -> 16 -> 32 -> 38 -> 43 -> 48 -> 52	**Cost**
Ali & Kamoun	No Convergence
Park & Keum	15469
Park & Choi	55608
Ahn & Ramakrishna	16467

Result 2

Path : 1 -> 6 -> 9 -> 15 -> 18 -> 20 -> 35	**Cost**
Ali & Kamoun	No Convergence
Park & Keum	8251
Park & Choi	No Convergence
Ahn & Ramakrishna	No Convergence

Result 3

Path : 1 -> 10 -> 20 -> 35 -> 40 -> 116	**Cost**
Ali & Kamoun	No Convergence
Park & Keum	8677
Park & Choi	No Convergence
Ahn & Ramakrishna	No Convergence

Result 4

Path : 31 -> 28 -> 36 -> 39 -> 110	**Cost**
Ali & Kamoun	No Convergence
Park & Keum	5008
Park & Choi	No Convergence
Ahn & Ramakrishna	5008

Result 5

Path : 1 -> 10 -> 20 -> 35 -> 40 -> 116	**Cost**
Ali & Kamoun	No Convergence
Park & Keum	8677
Park & Choi	No Convergence
Ahn & Ramakrishna	15554

Result 6

Path : 1 -> 10 -> 20 -> 35 -> 40 -> 116	**Cost**
Ali & Kamoun	No Convergence
Park & Keum	4821
Park & Choi	No Convergence
Ahn & Ramakrishna	No Convergence

Result 7

Path : 1 -> 10 -> 15 -> 25 -> 17	**Cost**
Ali & Kamoun	No Convergence
Park & Keum	5130
Park & Choi	No Convergence
Ahn & Ramakrishna	No Convergence

Convergence %	Convergent	Non Convergent
Ali & Kamoun	0.2%	99.8%
Park & Keum	99.8%	0.2%
Park & Choi	14.0%	86.0%
Ahn & Ramakrishna	41.0%	59.0%

Data Analysis - Sample Data Result for a **40 Node Network - Single Source to Single Destination**

Hopfield Neural Networks Convergent Algorithms

Ali & Kamoun Algorithm Path Traversed

Source/Destination	Path Taken	Cost(α)
Node 1 to 3	1-> 3	122
Node 1 to 6	No Convergence	–
Node 1 to 9	1-> 9	688
Node 1 to 12	1-> 11-> 12->	954
Node 1 to 15	1-> 3-> 25-> 14-> 15->	1369
Node 1 to 18	1-> 9-> 20-> 19-> 18->	1156
Node 1 to 20	1-> 9-> 20->	739
Node 1 to 23	1-> 10-> 23->	1300
Node 1 to 26	No Convergence	–

Park & Choi Algorithm Path Traversed

Source/Destination	Path Taken	Cost(β)
Node 1 to 3	1-> 3	122
Node 1 to 6	1-> 9-> 8-> 7-> 6->	2558
Node 1 to 9	1-> 9	688
Node 1 to 12	1-> 2-> 12	553
Node 1 to 15	1-> 3-> 25-> 14-> 15->	1369
Node 1 to 18	1-> 9-> 20-> 19-> 18->	1156
Node 1 to 20	1-> 9-> 20->	739
Node 1 to 23	1-> 10-> 23->	1300
Node 1 to 26	1-> 3-> 25-> 14-> 15-> 26->	2118

Ahn & Ramakrishna Algorithm Path Traversed

Source/Destination	Path Taken	Cost(ρ)
Node 1 to 3	1-> 3	122
Node 1 to 6	1-> 3-> 25-> 14-> 4-> 5-> 6->	1994
Node 1 to 9	1-> 9	688
Node 1 to 12	1-> 11-> 12	954
Node 1 to 15	1-> 3-> 25-> 14-> 15->	1369
Node 1 to 18	1-> 3-> 25-> 14-> 15-> 17-> 18->	1686
Node 1 to 20	1-> 9-> 20->	739
Node 1 to 23	1-> 10-> 23->	1300
Node 1 to 26	1-> 3-> 25-> 14-> 15-> 26->	2118

Park & Keum Algorithm Path Traversed

Source/Destination	Path Taken	Cost(ϕ)
Node 1 to 3	1-> 3	122
Node 1 to 6	1-> 3-> 25-> 14-> 4-> 5-> 6->	1994
Node 1 to 9	1-> 9	688
Node 1 to 12	1-> 2-> 12	553
Node 1 to 15	1-> 3-> 25-> 14-> 15->	1369
Node 1 to 18	1-> 9-> 20-> 19-> 18->	1156
Node 1 to 20	1-> 9-> 20->	739
Node 1 to 23	1-> 10-> 23->	1300
Node 1 to 26	1-> 3-> 25-> 14-> 15-> 26->	2118

Non-Neural Convergent Algorithms

Dijkstra Path Traversed

Source/Destination	Path Taken	Cost(σ)
Node 1 to 3	1-> 3	122
Node 1 to 6	1-> 3-> 25-> 14-> 4-> 5-> 6->	1994
Node 1 to 9	1-> 9	688
Node 1 to 12	1-> 2->12	553
Node 1 to 15	1-> 3-> 25-> 14-> 15->	1369
Node 1 to 18	1-> 9-> 20-> 19-> 18->	1156
Node 1 to 20	1-> 9-> 20->	739
Node 1 to 23	1-> 2-> 12->23->	1275
Node 1 to 26	1-> 3-> 25-> 14-> 15-> 26->	2118

Bellman Ford Path Traversed		
Source/Destination	**Path Taken**	**Cost(ω)**
Node 1 to 3	1-> 3	122
Node 1 to 6	1-> 9-> 20-> 19-> 18-> 17-> 6->	2233
Node 1 to 9	1-> 9	688
Node 1 to 12	1-> 2->12	553
Node 1 to 15	1-> 9-> 20-> 19-> 18-> 17-> 15->	2059
Node 1 to 18	1-> 9-> 20-> 19-> 18->	1156
Node 1 to 20	1-> 9-> 20->	739
Node 1 to 23	1-> 10-> 23->	1300
Node 1 to 26	1-> 9-> 20-> 19-> 18-> 17-> 16-> 26->	2465

Floyd Warshall Path Traversed	
Source/Destination	**Cost(τ)**
Node 1 to 3	122
Node 1 to 6	1995
Node 1 to 9	688
Node 1 to 12	553
Node 1 to 15	1369
Node 1 to 18	1156
Node 1 to 20	739
Node 1 to 23	1275
Node 1 to 26	2118

Total Results for 40 Node Network - Single Source to Single Destination

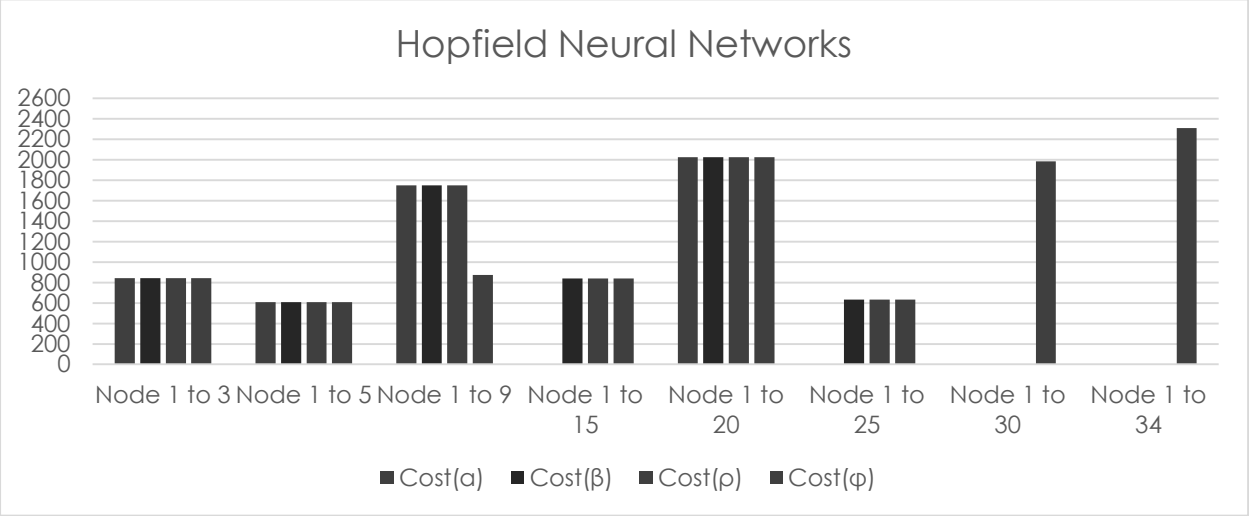
Source/Destination	Cost(α)	Cost(β)	Cost(ρ)	Cost(ϕ)	Cost(σ)	Cost(ω)	Cost(τ)
Node 1 to 3	122	122	122	122	122	122	122
Node 1 to 6	–	2558	1994	1994	1994	2233	1995
Node 1 to 9	688	688	688	688	688	688	688
Node 1 to 12	954	553	954	553	553	553	553
Node 1 to 15	1369	1369	1369	1369	1369	2059	1369
Node 1 to 18	1156	1156	1686	1156	1156	1156	1156
Node 1 to 20	739	739	739	739	739	739	739
Node 1 to 23	1300	1300	1300	1300	1275	1300	1275
Node 1 to 26	–	2118	2118	2118	2118	2465	2118

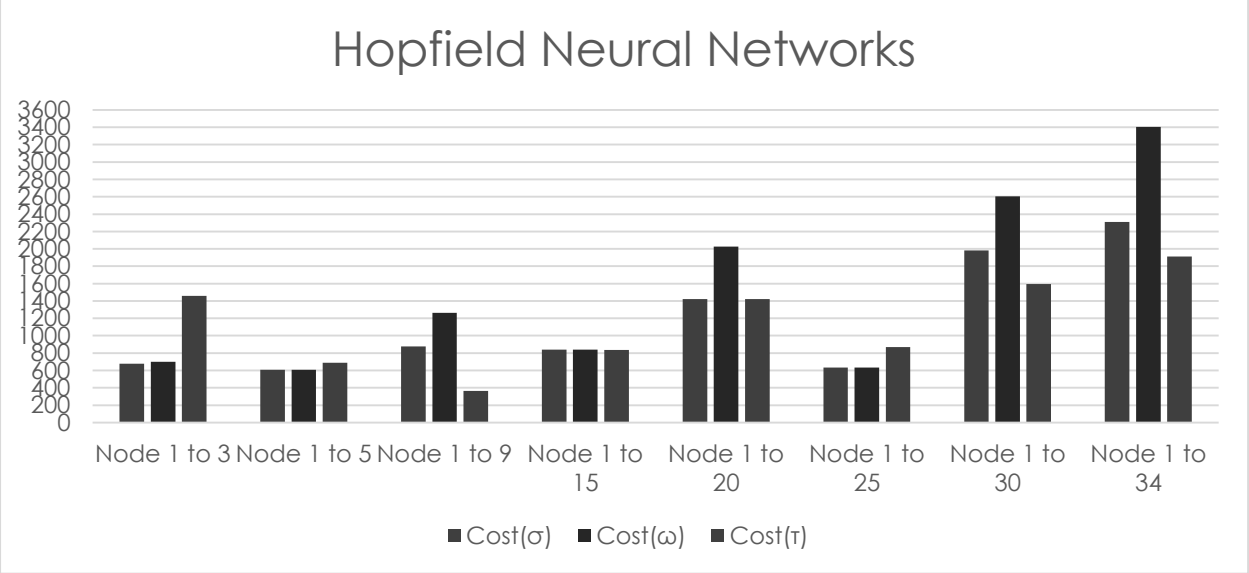
*Floyd Warshall Results will be uploaded in this link (as they are too large to fit) – the link provided is :-

Other Results

Other Results Generated

Source/Destination	Cost(α)	Cost(β)	Cost(ρ)	Cost(ϕ)	Cost(σ)	Cost(ω)	Cost(τ)
Node 1 to 3	843	843	843	843	676	699	1457
Node 1 to 5	607	607	607	607	607	607	690
Node 1 to 9	1749	1749	1749	875	875	1264	366
Node 1 to 15	-	839	839	839	839	839	837
Node 1 to 20	2026	2026	2026	2026	1420	2026	1421
Node 1 to 25	-	633	633	633	633	633	870
Node 1 to 30	-	-	-	1983	1983	2603	1595
Node 1 to 34	-	-	-	2311	2311	3404	1910

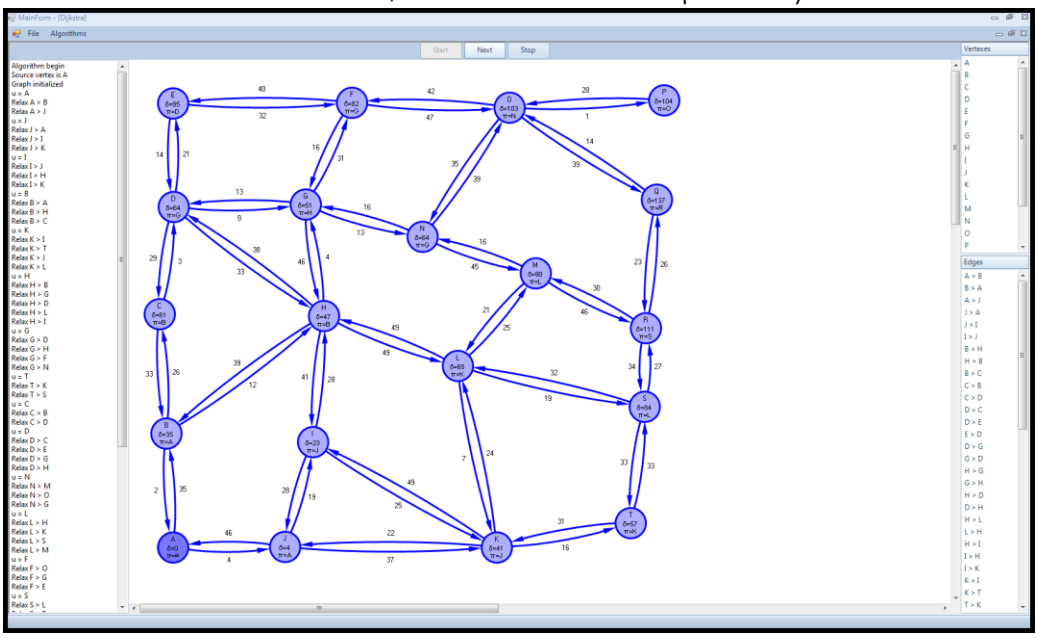


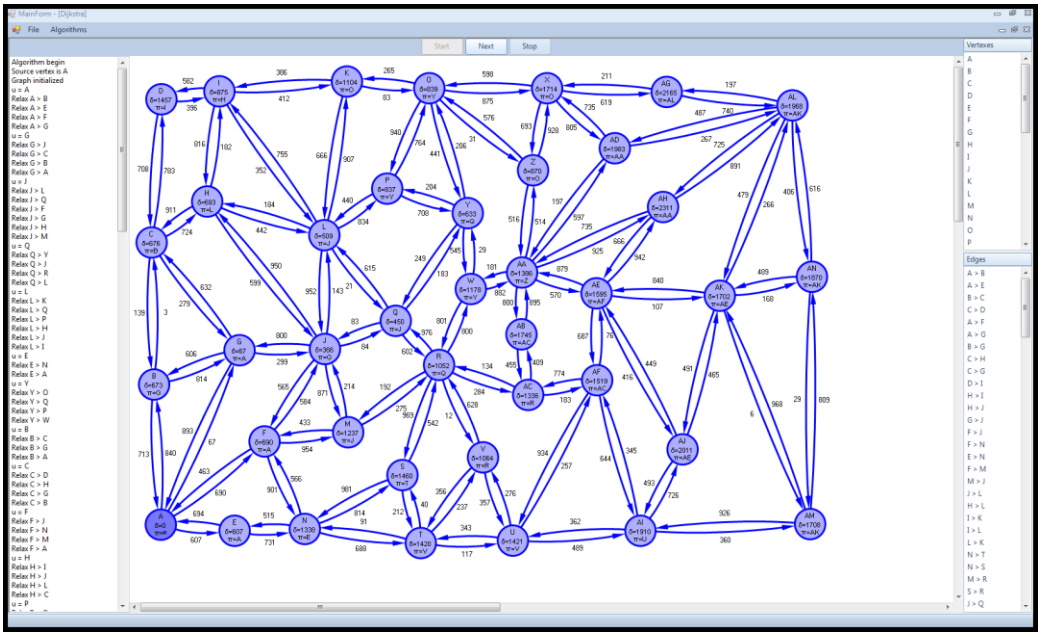


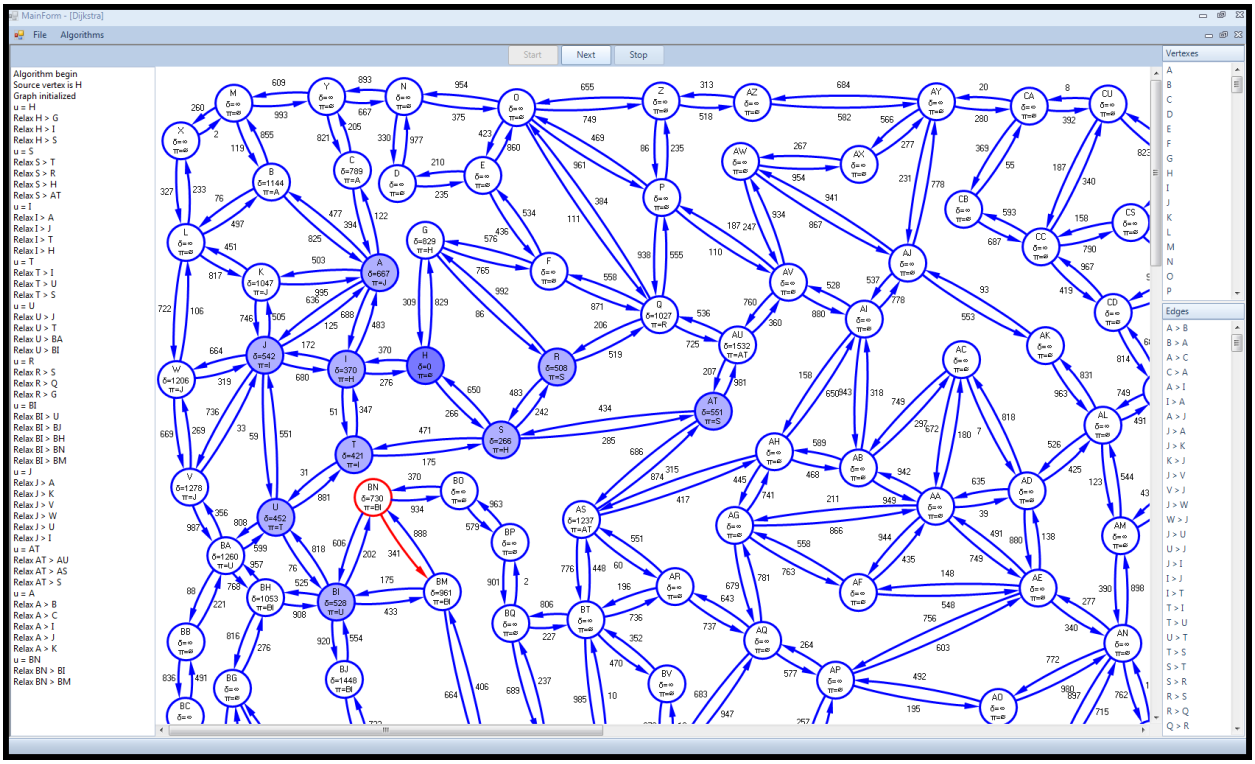
Appendix 2 – Images, Screenshots and Application Images

The following are relevant images of the application created for users to test. Also additional screenshots of scenarios and images that were used are attached.

A screenshot of the 20 Node, 40 Node and 180 respectively scenarios that were created:

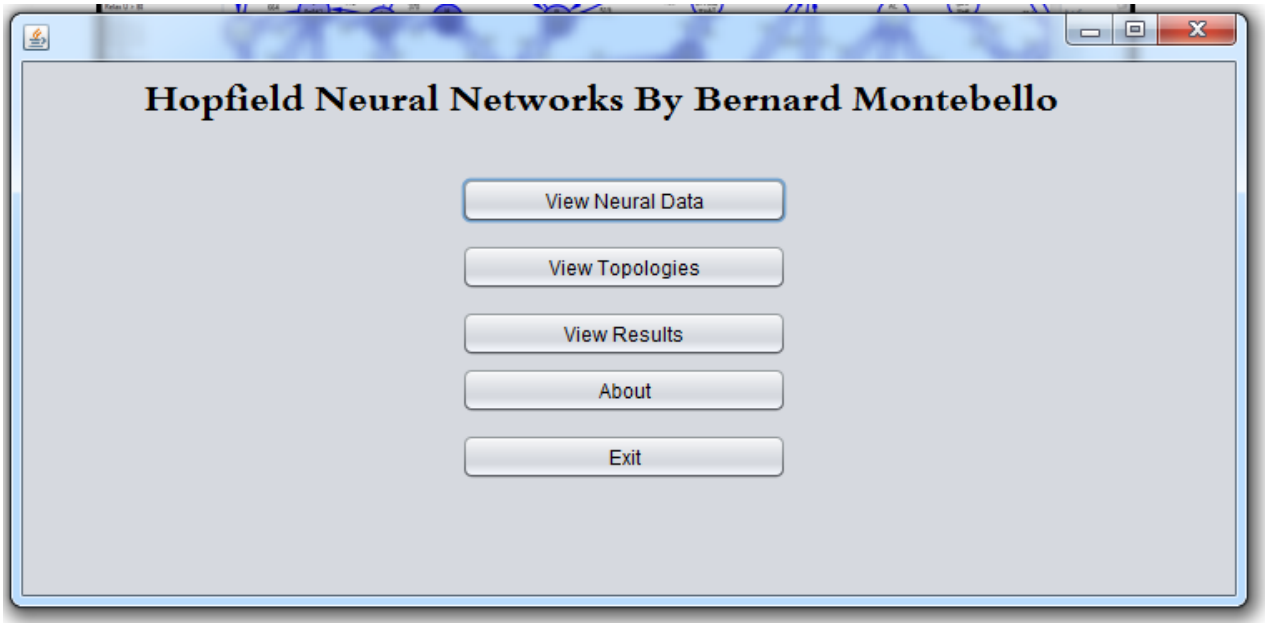




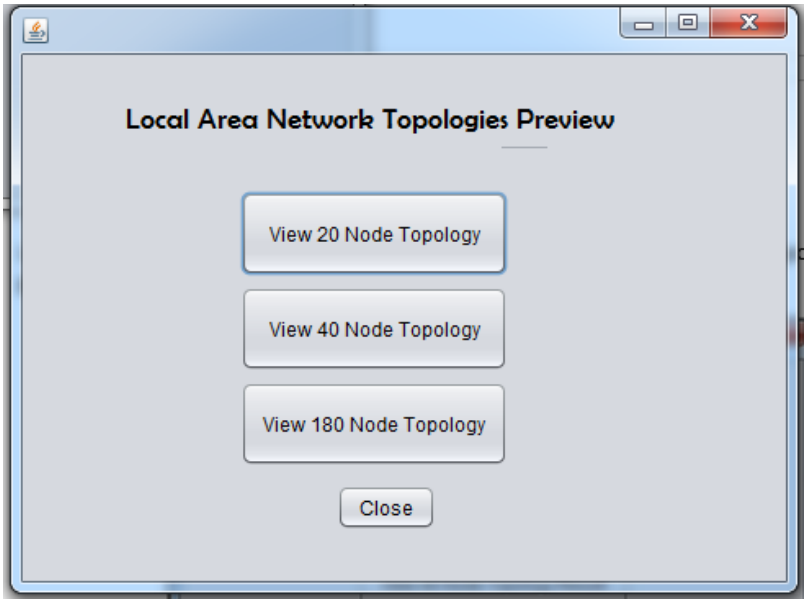


The following screenshots show the modified JAVA application with GUI:

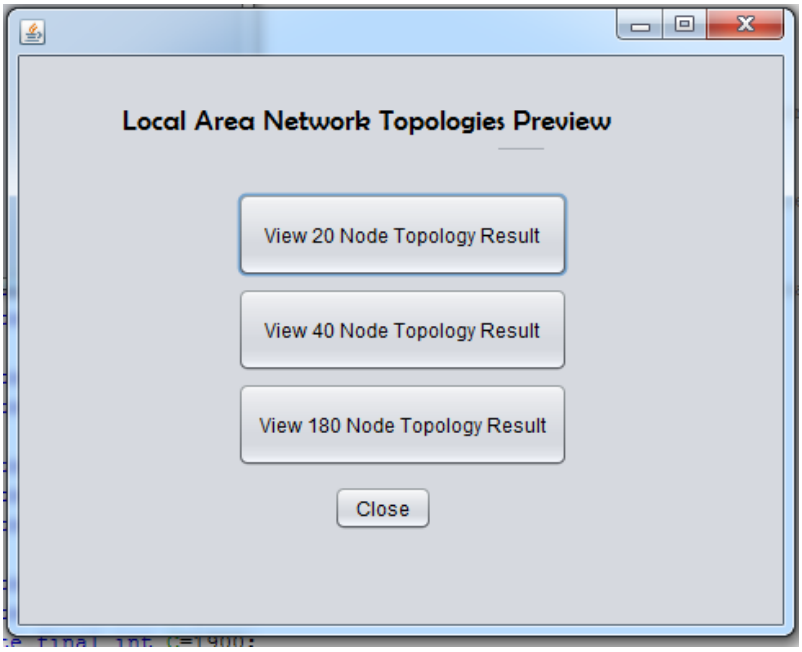
The welcome screen of the application:



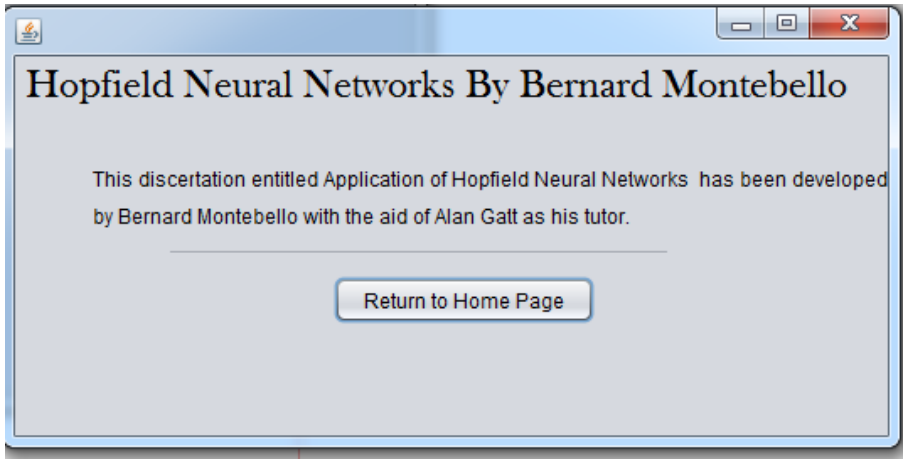
Local Area Network Preview Tap within the application: used to preview the topologies created and given weights.



This screenshot is taken from the result tab. This enable the user to preview already generated result in different tabs according to the scenarios.



The about tab, is just information of the program.



Appendix 3 – Relevant Information

This section will lead you than an online repository of all the necessary simulation applications, results, papers researched, code for simulations, videos and other material that was used in this dissertation. Relevant Papers are suggested to the reader to broaden knowledge about Hopfield Neural Networks, networking and other research areas.