Journal of Engineering Science and Technology
Vol. 7, No. 3 (2012) 339 - 350

A HYBRID HOPFIELD NEURAL NETWORK AND TABU SEARCH ALGORITHM TO SOLVE ROUTING PROBLEM IN COMMUNICATION NETWORK

TAYSEER S. ATIA*, MANAR Y. KASHMOLA

Department of Computer Science, University of Technology, Baghdad, Iraq
*Corresponding Author: tayseer_salman@yahoo.com

Abstract

The development of hybrid algorithms for solving complex optimization problems focuses on enhancing the strengths and compensating for the weakness of two or more complementary approaches. The goal is to intelligently combine the key elements of these approaches to find superior solutions to solve optimization problems. Optimal routing in communication network is considering a complex optimization problem. In this paper we propose a hybrid Hopfield Neural Network (HNN) and Tabu Search (TS) algorithm, this algorithm called hybrid HNN-TS algorithm. The paradigm of this hybridization is embedded. We embed the short-term memory and tabu restriction features from TS algorithm in the HNN model. The short-term memory and tabu restriction control the neuron selection process in the HNN model in order to get around the local minima problem and find an optimal solution using the HNN model to solve complex optimization problem. The proposed algorithm is intended to find the optimal path for packet transmission in the network which is fills in the field of routing problem. The optimal path that will be selected is depending on 4-tuples (delay, cost, reliability and capacity). Test results show that the propose algorithm can find path with optimal cost and a reasonable number of iterations. It also shows that the complexity of the network model won't be a problem since the neuron selection is done heuristically.

Keywords: Hopfield neural network, Tabu search, Optimization problem.

1. Introduction

An Artificial Neural Network (ANN) is a system that consists of a number of simple processors running in parallel. The Hopfield Neural Network (HNN), a

Nomenclatures

C_{xi}	Cost of Neuron at location *(x, i)*
c(u,v)	capacity from *u* to *v*
d(u,v)	delay from *u* to *v*
E_{obj}	Energy function for objective
I_{xi}	Bias of Neuron at location *(x,i)*
i	Column number
n	Number of neuron in column or row
net_{xi}	Activity of Neuron at location *(x, i)*
r(u,v)	reliability from *u* to *v*
t(u,v)	cost from *u* to *v*
u	Source node
V_{xi}	Output of Neuron at location *(x, i)*
$V_{xi}(t)$	Output of Neuron at location *(x, i)* for current iteration
$V_{xi}(t+1)$	Output of Neuron at location *(x, i)* for next iteration
v	Destination node
$W_{xi,yj}$	Weight between Neuron *xi* and *yj*
x	Row number

Abbreviations

ATM	Asynchronous transfer mode
HNN	Hopfield neural network
SP	Shortest path
TS	Tabu search
TSP	Travel sales person
VB6	Visual Basic version six

well-known network, was proposed to solve optimization problems based on the Lyapunov energy function [1]. Many researchers have subsequently applied it to the complex optimization problem such as find the minimum cost path [2], solving shortest path problem [3], shortest path computation and routing in ATM network [4].

Hopfield Neural Network suffers from the local minima problem [2, 5], where the energy function results in a solution, but it is not the optimal solution. Many researchers have incorporate different techniques with HNN in order to get rid of the local minima problem, such as the Boltzmann machine with temperature parameter T to control the probability of acceptance of the change of the selected neuron [5].

The Chaotic Neural Network is also used to get around this problem. In this network, a random noise is added to the network [1].

Routing is the process of selecting paths in a network to help in forwarding; forwarding means place the packet in its route to its destination [6].

In this paper we propose a hybrid Hopfield Neural Network (HNN) and Tabu Search (TS) algorithm called HNN-TS to find the optimal path in the network. We employ the short-term memory and tabu restriction feature from TS in the HNN to get globally optimal or near optimum results.

2. Hopfield Neuron Network for Optimal Path Problem

Two different methods can be used to represent the HNN model; the vertex path representation and the edge path representation. The edge path representation method is used to represent the HNN, It uses an $n \times n$ binary matrix to represent the edges. Each neuron in the array is identified by double indices (x, i), where x and i indicate the row and column number, respectively. The neuron at location (x, i) shows the link from node x to i in the network. In order to characterize the neuron activities at location (x, i), the neuron state V_{xi} is defined as:

$$V_{xi} = \begin{cases} 1 & \text{if there is an edge from node } x \text{ to node } i \\ 0 & \text{otherwise} \end{cases}$$

Corresponding to each edge (x, i), there is a non-negative weight C_{xi} representing the cost from node x to node i.

2.1. Definition of the objective function

To compute the best path using the HNN model, the objective function is defined as:

$$E_{obj} = \frac{1}{2} \sum_{x=1}^{n} \sum_{i=1}^{n} C_{xi} V_{xi} \tag{1}$$

Since we select the neuron to be activated or deactivated heuristically, there is no need to add the term that checks the availability of only one active neuron in each row and column. Also, the term that checks the generated path contains the start and destination nodes.

$$V_{xi}(t+1) = \begin{cases} 1 & net_{xi} > 0 \\ V_{xi}(t) & net_{xi} = 0 \\ 0 & net_{xi} < 0 \end{cases} \tag{2}$$

and

$$net_{xi} = \sum_{j=1}^{n} W_{xi,yj} V_{yj} + I_{xi} \tag{3}$$

2.2. Interconnection weights and input bias

Interconnection weights and input bias are derived on the basis of the cost of the link. There are two types of connection weights:

- Connection weight between neurons in the same row or column. This weight is assigned to a negative value to discourage activating the neuron in the same row or column, and it is of equal cost.

- Connection weight between neurons in a different row and column. This weight is equal to the cost value between the two neurons if there is a connection between them in the cost matrix, otherwise it is 0.

The bias is a positive number assigned to each neuron to describe the preference of the neuron for the active state. The bias is derived from the cost of the link that the neuron represents. The smaller the cost, the great the preference for the neuron to be in the active state.

$$bias = \frac{1}{link \cos t} \tag{4}$$

2.3. Path factors and their formulation

This paper addresses the problem of optimum path selection between two nodes, the source and destination. This path is called the best path. The most important factors selected to form the criteria for an optimum path are reliability, delay, capacity and cost. The link selection for the data path should be in such away that the path has optimum values of reliability, delay, capacity and cost. We propose the following mathematical expression to quantify these factors:

$$MinimizeF = \sum \frac{1}{r(u,v)} * \frac{1}{c(u,v)} + d(u,v) + t(u,v) \tag{5}$$

The proposed formula ensures that the combination of these tuples (r, d, c and t) is minimized where r and c are taken in the inverse form in order to minimize their value if they have a large value. Then, they are multiplied to absorb their effect. Finally the result is added to d and t. So, if d and t are large values, then the overall path combination will result in maximum value. Since the path is a sequence of communication links from s to d, each one with 4-tuple, then the minimization function will sum all these values.

3. Hybrid HNN and TS Based on Short-Term Memory

In TS short term memory, also called Recency-based memory is used to keep track of solution attributes that have changed during the recent past. This memory is exploited by labeling the selected attributes that occur in solutions recently visited "tabu-active".

Consequently, solutions that contain tabu–active elements become tabu. This will prevent certain solutions in the recent past from being revisited.

In researches that use HNN to solve complex optimization problems like SP (Shortest Path), TSP (Travel Sales Person) focuses on the random selection of a neuron to be activated. Each time a random neuron is selected, if its activation level is greater than the threshold, then the neuron state will change to 1.

The resulting solution could be a feasible or an infeasible solution, so the HNN takes a long time to learn and the resulting final solution may not be optimal.

Another problem facing the researcher when solving optimization problems using HNN is the local minima problem. In this case, the network falls in a region and cannot cross it. This results in a sequence of solutions with the same energy function.

The proposed hybrid algorithm incorporates the short-term memory feature in the neuron selection operation of the HNN algorithm. The idea behind the proposed algorithm is that the HNN starts with an initial solution generated in a heuristic procedure in such away that it is a feasible solution. Then the neuron selection procedure is controlled by a short-term memory strategy. The neuron selection procedure takes the initial generated solution as an input parameter. Each time one node is dropped from the solution. It represents a neuron in the HNN. The new solution generated by HNN is stored in the short-term memory. The dropped neuron is also labeled tabu for a number of iterations (determined randomly).

The generated new solution is now used by the neuron selection procedure and the same operation is repeated after a determined number of iterations. The dropped neurons are allowed to participate again in solution generation. In this case, the combination of added and dropped neurons generates a number of feasible solutions. This process allows only feasible neurons to participate in solution generation. It also searches for a specific region heuristically rather than searching the whole search space randomly without heuristics' , and this results in a long iteration and local minima problem. The proposed hybrid algorithm to solve the best path problem in the network is described as follows:

Algorithm (1) General HNN-TS Algorithm

Step1: *get the cost and connection matrix of the network*
Step2: *set the initial input for all neurons*
Step3: *calculate the bias term*
Step4: *calculate the connection weight of each pair of neurons*
Step5: *generate an initial solution using a heuristic procedure*
Step6: *repeat*
Step7: *select a neuron to be* ***on/off*** *using the neuron selection procedure*
Step8: *until there is no available neuron to be selected*

The neuron selection procedure represents the key element to embed the short term memory operation in the HNN model.

Algorithm (2) The Neuron Selection Algorithm

Step1: *set* $i = 0$

Step2: *set* $j = i+2$, x = *initial solution.*

Step3: *if there is a link between neuron position* i *and position* j *in* x *then*

label all neurons between i *and* j *as tabu for* n *iteration.*

Step4: *for each tabu- active neuron*

1: Set neuron threshold = (C_{KL} * *No. of tabu- active neuron)*

2: Compute neuron activation level as net_{kl}

3: If net_{kl} < *neuron threshold then set neuron state to* ***OFF.***

Step5: *select new neuron at position* i,j *to be activated.*

Step6: *compute neuron activation level as* net_{ij}

Step7: *if* net_{ij} > *neuron threshold then*

1: set neuron state to ***ON***

2: update neuron weight and its neighborhood as

new weight = old weight + cost

3: compute e *for new solution.*

Step8: *set* x=*new solution after dropping tabu- active neuron.*

Step9: $j = j+1$, *If* j < *new solution length, then go to step3.*

Step10: *if* $n = 0$ *for any tabu-active element then move them to* ***addlist***

Step11: *for each element in* ***addlist***

Step12: *set* y = *current element from add list*

Step13: z = *last feasible solution in HNN model*

Step14: *check if there is a link between* y *and any two neurons at position* i, $i+1$ *in* z

Step15: *select new neurons at position* i,y *and* $i+1,y$ *to be activated one neuron at a time*

Step16: *compute neuron activation level as* ***net***

Step17: *if* ***net*** > *neuron threshold then*

1: set neuron state to ON.

2: update neuron weight and its neighborhood as

new weight = old weight + cost

3: compute e *for new solution.*

4. Simulation Model for Hybrid HNN-TS

The test problem used to examine the effectiveness of the proposed algorithm is a network with 50 nodes (routers) and 98 links. Each link is characterized by 4-tuples (delay, capacity, reliability, and cost). Figure 1 shows the network structure. Table 1 summarizes the network factors. The HNN model used to represent this network is a fully connected matrix with a size of 50*50 neurons. Figure 2 shows 5×5 simple HNN model for explanation.

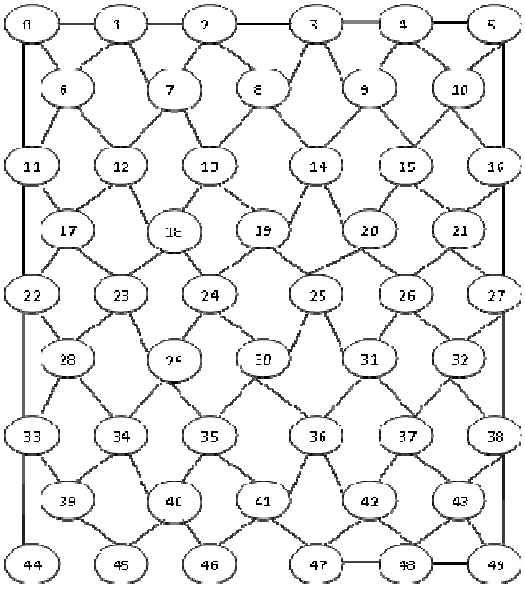

Fig. 1. Network Structure.

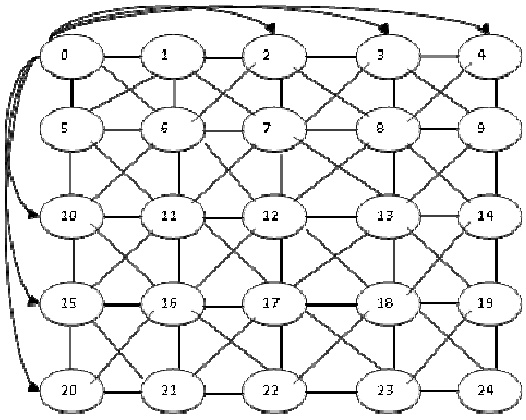

Fig. 2. HNN Model for 5×5 Node.

Figure 2 does not show all possible links, but it does at least show some of them. Every node (neuron) has a connection with all other nodes in the model except itself. The output of the HNN is a binary matrix of the size 50×50, where the optimal path is represented as a sequence of 1's. An active neuron at position (i, j) is represented by 1 in the output matrix, and it means there is a link between node i and node j in the network.

Table 1. Network Factors.

S to D	Delay/ Second	Capacity/ MB	Reliability	Cost	S to D	Delay/ Second	Capacity/ MB	Reliability	Cost
0 to 1	0.1	5	0.7	3	22 to 28	0.8	7	0.7	6
0 to 6	0.3	6	0.8	4	22 to 33	0.7	9	0.8	7
0 to 11	0.5	6	0.5	2	23 to 28	0.5	6	0.7	7
1 to 2	0.5	5	0.7	2	23 to 29	0.4	7	0.8	7
1 to 6	0.4	8	0.9	2	24 to 29	0.2	7	0.6	2
1 to 7	0.3	7	0.8	1	24 to 30	0.4	6	0.5	4
2 to 3	0.4	4	0.6	3	25 to 30	0.9	8	0.6	5
2 to 7	0.2	5	0.8	2	25 to 31	0.5	7	0.4	3
2 to 8	0.1	7	0.9	4	26 to 31	0.7	7	0.7	4
3 to 4	0.7	9	0.7	5	26 to 32	0.8	8	0.8	5
3 to 8	0.2	6	0.9	5	27 to 32	0.5	5	0.5	2
3 to 9	0.3	7	0.5	4	27 to 38	0.6	7	0.7	6
4 to 5	0.2	6	0.5	3	28 to 33	0.6	4	0.3	1
4 to 9	0.1	8	0.4	1	28 to 34	0.7	8	0.5	5
4 to 10	0.3	6	0.5	5	29 to 34	0.5	8	0.6	2
5 to 10	0.5	7	0.6	8	29 to 35	0.7	9	0.5	4
5 to 16	0.6	9	0.9	6	30 to 35	0.1	7	0.4	3
6 to 11	0.7	7	0.6	4	30 to 36	0.6	6	0.6	1
6 to 12	0.5	6	0.8	5	31 to 36	0.4	6	0.8	6
7 to 12	0.4	8	0.6	3	31 to 37	0.3	8	0.7	5
7 to 13	0.9	9	0.5	6	32 to 37	0.6	7	0.6	4
8 to 13	0.7	5	0.5	2	32 to 38	0.4	9	0.4	4
8 to 14	0.6	4	0.7	6	33 to 39	0.5	5	0.5	2
9 to 14	0.3	3	0.8	7	33 to 44	0.8	5	0.9	6
9 to 15	0.2	1	0.9	1	34 to 39	0.2	7	0.4	6
10 to 15	0.1	2	0.5	1	34 to 40	0.4	6	0.4	7
10 to 16	0.8	8	0.9	7	35 to 40	0.3	8	0.3	2
11 to 17	0.4	5	0.3	1	35 to 41	0.4	6	0.8	5
11 to 22	0.6	8	0.4	3	36 to 41	0.5	8	0.6	6
12 to 17	0.3	5	0.7	6	36 to 42	0.9	9	0.9	9
12 to 18	0.6	6	0.5	4	37 to 42	0.7	6	0.5	3
13 to 18	0.8	7	0.5	4	37 to 43	0.8	5	0.4	1
13 to 19	0.6	8	0.4	5	38 to 43	0.5	5	0.2	1
14 to 19	0.4	5	0.7	4	38 to 49	0.8	6	0.1	2
14 to 20	0.5	6	0.6	4	39 to 44	0.3	8	0.4	9
15 to 20	0.1	3	0.8	2	39 to 45	0.4	6	0.3	2
15 to 21	0.3	4	0.7	2	40 to 45	0.5	7	0.5	4
16 to 21	0.5	6	0.6	6	41 to 46	0.6	7	0.7	4
16 to 27	0.4	7	0.6	4	41 to 47	0.7	5	0.5	2
17 to 22	0.5	6	0.6	5	42 to 47	0.8	8	0.7	8
17 to 23	0.6	7	0.8	6	42 to 48	0.5	7	0.6	6
18 to 23	0.7	8	0.6	5	43 to 48	0.3	4	0.5	2
18 to 24	0.9	5	0.7	3	43 to 49	0.4	6	0.3	5
19 to 24	0.1	6	0.8	6	44 to 45	0.4	7	0.8	6
19 to 25	0.3	7	0.5	6	45 to 46	0.3	6	0.7	4
20 to 25	0.2	4	0.5	4	46 to 47	0.1	8	0.9	6
20 to 26	0.4	7	0.5	3	47 to 48	0.6	7	0.8	5
21 to 26	0.6	5	0.8	4	48 to 49	0.5	5	0.5	5
21 to 27	0.7	7	0.5	9					

5. Computer Program

The proposed hybrid algorithm does not require a programming language with a special capability. So, it can be implemented using C,C++,VB6,VB.Net and any other languages. The network model is implemented using VB6 as programming

language to simulate the network model. This language facilitates the implementation of the user interface. An optimal path can be computed between any two nodes by just feeding the network model with source and destination nodes, then the network program compute a feasible solution by use a heuristic procedure. This procedure constructs the initial path by finding the shortest path between two nodes; add the path to the feasible solution and so on, till reach the destination. After that, the network is trained to activate the feasible solution. This requires a single iteration to activate a neuron corresponding to each node in the path, for example if the path length is 10 nodes, then 10 iterations required to activate this feasible solution. The network simulation is started to generate different solutions. At this stage the neuron selection procedure is activated. This procedure employs the process of ADD and DELETE node, and implements the short-term memory operation to control the added and deleted neurons. In this memory the neuron is added to the network or deleted from the network for two iterations (controlled by tabu tenure).

For every generated solution a move value is computed either after add new node or after delete an old node. Move value represents the value added or removed from the cost. For example, if new neuron is activated in the network, then move value represents the new cost of this neuron. If an old neuron is deactivated in the network, the removed cost that represents this neuron will be the move value. The energy value is computed for each path, the path with minimum energy will be the optimal path. Finally, we explain the output matrix for optimal path, if the network consists of 5 nodes. The required path from 0 to 4,The optimal path is 0 2 3 4. The output matrix looks like this:

$$V_{xi} = \begin{cases} \mathbf{4} & & \mathbf{0} & \mathbf{1} & \mathbf{2} & \mathbf{3} \\ \mathbf{0} & 0 & 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{2} & 0 & 0 & 0 & 1 & 0 \\ \mathbf{3} & 0 & 0 & 0 & 0 & 1 \\ \mathbf{4} & & 0 & 0 & 0 & 0 \end{cases}$$

6. Simulation Example and Discussion

We select the path from 0 to 27 to simulate the model of the proposed algorithm. First we use the heuristic procedure to generate a feasible initial solution. Then the neuron selection procedure is used to select a neuron for activation or deactivation.

Initial solution = 0 1 2 3 4 5 10 15 9 14 8 13 7 12 6 11 17 22 33 28 23 18 24 19 25 20 26 21 16 27 , initial cost = 145

We need 30 iterations to activate the feasible solution using the HNN model, one neuron at a time. Table 2 summarizes the short term memory operation in the select neuron procedure, where iterations column represents the number of iterations in the simulation process. The add and drop column represents the new added and/or deleted neuron. For the first entry in Table 2, number of iterations required to generate a feasible solution with cost = 82 and energy = 42.070 are 14 iterations. The dropped neurons are (*1 2 3 4 5 10 15 9 14*). In the simulation process, if we want to drop one node from the path , it requires to remove every

link for this node, for example if want to drop node 1 from the path, we must remove links between nodes (0, 1) and nodes (1, 2). The complete elimination process for nodes (*1 2 3 4 5 10 15 9 14*) is explained in the first entry of the second column. Third column explain how node 0 and node 6 is connected together in the path after remove nodes *(1 2 3 4 5 10 15 9 14*) to generate feasible solution:

0 6 11 17 22 33 28 23 18 24 19 25 20 26 21 16 27, this path has move value = 63, it represents the eliminated cost for nodes (*1 2 3 4 5 10 15 9 14*) after subtract the new cost for link between 0 and 6. Last entry in the Table 2 adds node 16 to the path, we can note at third column how the link between node 16, node 21 and link between node 16, node 27 have been activated to generate the feasible path. We can note how the energy oscillates between high and low values in this table. Table 3 summarizes the new generated paths and the added /dropped nods. After 154 iterations, we find that the most optimal solution = 30 for the path 0 1 2 3 4 5 16 27. Table 4 shows some selected paths with the best cost found, the number of iterations and time to convergence.

Table 2. Short-Term Memory Operation In Neuron Selection Procedure.

Iterations	Tabu-active Net Tenure		Move value	Cost	Energy
	Drop Link	Add Link			
30 – 44	(0,1) , (1,2) , (2,3),(3,4),(4,5), (5,10), (10,15), (15,9),(9,14), (14,8), (8,13), (13,7),(7,12)	(0,6)	63	82	42.070
44 - 47	(11,17), (17,22)	(11,22)	2.328	79.672	41.037
47 - 50	(22,33), (33,28)	(22,28)	0.664	79.008	37.116
50 - 53	(21,16) , (16,27)	(21,27)	6.613	72.395	33.728
53 -55	----	(1,0), (1,6)	-0.654	73.059	36.691
55-57	----	(16,21),(16,27)	-3.268	76.327	40.079

Table 3. Summery of Generated Paths.

Drop Node	Add Node	Generated Path
1,2,3,4,5,10,15,9,14,8,13,7,12	---	0 6 11 17 22 33 28 23 18 24 19 25 20 26 21 16 27
17	---	0 6 11 22 33 28 23 18 24 19 25 20 26 21 16 27
33	---	0 6 11 22 28 23 18 24 19 25 20 26 21 16 27
16	---	0 6 11 22 28 23 18 24 19 25 20 26 21 27
----	1	0 1 6 11 22 28 23 18 24 19 25 20 26 21 27
----	16	0 1 6 11 22 28 23 18 24 19 25 20 26 21 16 27

Table 4. Different Selected Paths and Their Cost.

Path	Best Cost	Iterations	Time/s
3-40	105	281	3
49-1	38	1	1
15-30	58	98	2
5-19	34	60	1
0- 45	35	9	1

Finally, the cost and energy distribution for 3 selected paths are shown in Figs. 3 to 5. Figure 3 shows the cost and energy distribution for path (0-27) for 80 iterations, and Figs. 4 and 5 show the cost and energy distribution for paths (3-40) and (15-30) for 87 and 73 iterations, respectively. As we can see, the energy function for these paths suffers from the local minima problem, but we can get around this problem by using the neuron selection procedure. In this procedure, the neuron is heuristically selected to be activated or deactivated according to the short term memory operation. The neuron selection procedure controls the process of finding solutions on the network model. The termination condition is decided by the neuron selection procedure rather than by the energy function.

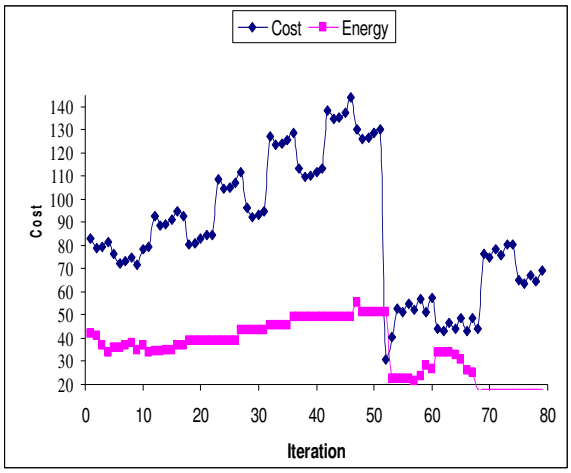

Fig. 4. Cost and Energy Distribution for Path (0-27).

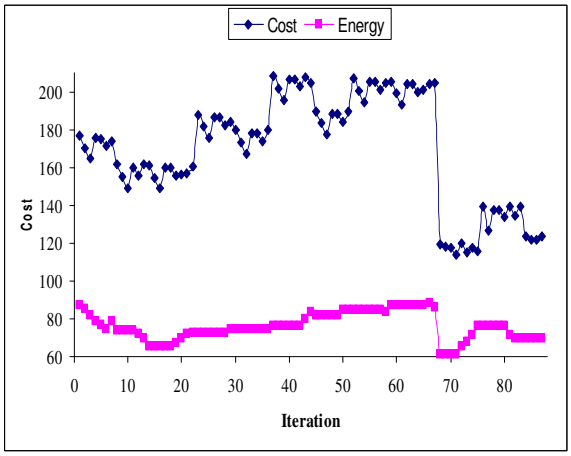

Fig. 5. Cost and Energy Distribution for Path (3-40).

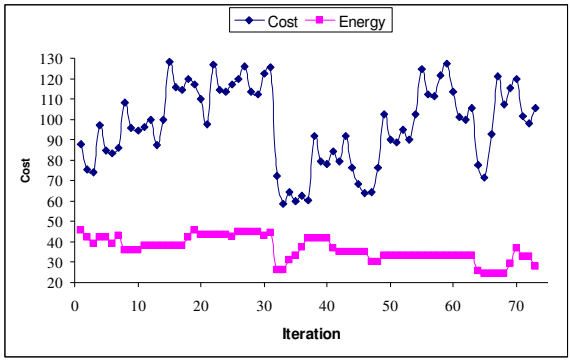

Fig. 6. Cost and Energy Distribution for Path (15-30).

7. Conclusions

Some concluding observations from the investigation are given below.

- The proposed Hybrid HNN-TS algorithm satisfies the goal of the paper.
- The proposed algorithm gets around the local minima problem by using the neuron selection procedure which controls solution generation and termination criteria.
- The proposed algorithm can process any network size with no fear of increasing complexity since the neuron selection is done heuristically.

References

1. Lin, J.-S.; LIU, M; and Huang, N.-F. (2000). The shortest path computation in MOSPF protocol through an annealed chaotic neural network. *Proceedings of the National Science Council, Republic of China. Part A,* 24(6), 463-471.
2. Hong, S.-G.; Kim, S.-W.; and Lee J.-J. (1995). The minimum cost path finding algorithm using a Hopfield type neural network. *Korea Advanced Institute of Science and Technology*, 4, 1719-1726.
3. Wang, J. (1996). A recurrent neural network for solving the shortest path problem. *IEEE Transaction on Circuits and System: Fundamental Theory and Applications*, 43(6), 482-486.
4. Selvanathan, N.; and Lee C.W. (2003). Hopfield model for shortest path computation and routing in ATM network. *Information Technology Communication*, 1, 57-62.
5. Raul, R. (1996). *Neural networks*. Springer-Verlag, Berlin.
6. Behrouz, A.F. (2006). *TCP/IP protocol suite* (3rd Ed.). McGraw-Hill International Edition.