The Neural Network Approach for the Shortest Path Planning Problem

Valentin Dimakov

Department of Computers and Mechanics,
Brest polytechnic institute, Moscowskaja 267,
224017 Brest, BELARUS
E-mail: VDimakov@mail.ru, cm@brpi.belpak.brest.by,
Fax: +375 162 42 21 27, Phone: +375 162 42 10 81

Abstract

This paper describes the neural network solving the shortest path planning problem. It consists of the layer set, where each layer forms a path-candidate for the fixed number of cities. During competition the layer-winner determining the sequence order of cities in the shortest path is selected. In this paper the experimental results obtained by the neural network are described in comparison with some known methods solving the same problem.

1: Introduction

The classical problem of combinatorial optimization is the problem about the shortest path [1]: let we have *n* cities, where two of them are fixed. The goal of this problem is to find the simple shortest path between two these cities. Certainly, the problem solution is the chain of cities forming the shortest path. Thus it is necessary to define their sequence, i.e. a position of each city in the path. The classical approach of the path representation is the arrangement of cities on a matrix *n´n* as it is shown on a Figure 1, where each city of *A..E* can have different positions in the shortest path, i.e. *1..4*. Thus this scheme allows uniquely defining the cities-participants and their places in the path. If two cities are fixed then they have the first and the last place in the path correspondingly. Hence we have three-dimensional nature of the problem: at first we have to define the place of each city in the shortest path, and secondly, the number of cities forming the shortest path can be in the range from *2* to *n*.

There are very interesting and efficient approaches solving this problem like famous Dijkstra's algorithm [2] and dynamic programming method [3]. But they are oriented only on solution of the problem on numerical computers. The development of neural network technique is oriented on different technological basis allowing creating efficient systems for different applications.

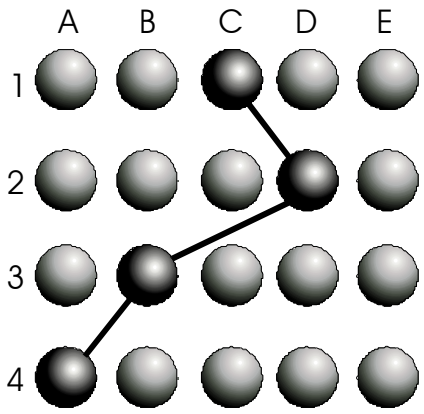

Figure 1. The possible scheme of the arrangement of cities in the path between cities "C" and "A".

There is also a possibility to solve this problem using the neural network technique like the Hopfield's neural network [4]. The solution using the neural network like that is enough difficult and it is defined by a weight matrix as follows [5]:

$$T_{AiBj} = -ad_{Aa}d_{Ba}d_{i1}d_{j1} - 2bd_{Ab}d_{Bb} + bd_{Ab}d_{ij} -$$
$$-gd_{ij}(1-d_{AB}) - hd_{AB}(1-d_{ij}) - 2qd_{ij}(1-d_{Ab})*$$
$$*(1-d_{Bb})(1-d_{AB}) - qd_{ij}d_{AB}(1-d_{Ab}) - [2(1-d_{Aa})* \quad (1)$$
$$*(1-d_{AB}) + d_{AB}]qd_{ij}(1-d_{Ba})(1-d_{i1}) - 2qd_{ji+1}*$$
$$*(1-d_{Ab})(1-d_{Ba}) - c(1-d_{AB})C_{AB}(d_{ji+1} + d_{ji-1}),$$
$$U_{Ai} = -ad_{Aa}d_{i1} - bd_{Aa}$$

where

$$d = \begin{cases} 1, if\ i = j \\ 0, otherwise \end{cases} \quad (2)$$

The difficult here is to define constant parameters a,b,g,h,q and c. In this paper the simpler neural network approach solving efficient this problem is considered. On basis of the problem condition the general architecture of the proposed neural network is defined (Figure 2).

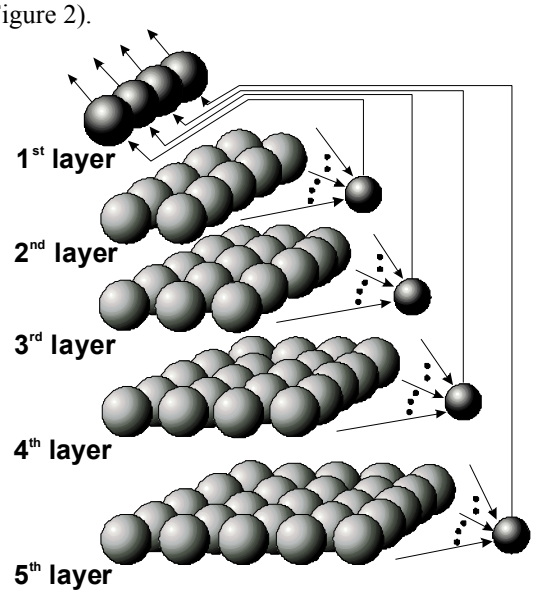

Figure 2. The architecture of the neural network for the problem solution of 5 cities.

2: The neural network architecture

Generally the neural network consists of n layers, where n is a total number of cities. The first layer selects the best variant of paths-candidates, which are offered by the remaining layers. The second layer forms the path-candidate consisting of two cities; the third one forms the path-candidate consisting of three cities etc.; and, at last, the n^{th} layer forms the path-candidate consisting of n cities, where two of them are fixed. At least one path-candidate of *n-1* offered must exist for successful solution.

The structure of the first layer is shown on a Figure 3.

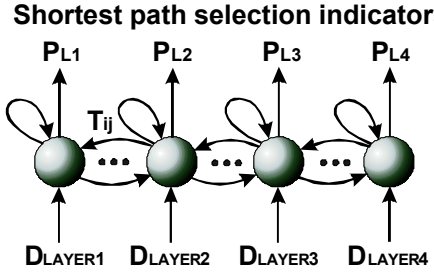

Figure 3. The structure of the first layer.

It is the single-layer relaxation competing neural network determining the neuron-winner having the output value equal to "1", and the all-remaining neurons have output value equal to "0". As input value for each neuron the value describing length of each offered path is used. The braking and self-exciting connections T_{ij} interconnect the neurons among themselves. Their weight values are follows:

$$T_{ij} = \begin{cases} 1 & , if\ \ i = j \\ -1/(m+1), & otherwise \end{cases} \quad (3)$$

where m is a network dimension. The iterative process of such neural network is described as:

$$Pl_i(0) = D_{LAYER\,i} \quad (4)$$

$$Pl_i(t+1) = fn\left(\sum_{j=1}^{m} T_{ij} * Pl_j(t)\right), \quad (5)$$

where

$$fn(x) = \begin{cases} 0\ , if\ x \le 0 \\ 1\ , if\ \ x \ge 1 \\ x\ , otherwise \end{cases} \quad (6)$$

Here D_{LAYERi} is a parameter describing a length of path formed by other layer.

During an iterative process one of neurons becomes the winner determining the best path.

The important components of the neural network are the layers forming the paths-candidates consisting of different number of cities, i.e. *2,3,..,n* (Figures 4 and 7). Hence the neural network has a pyramidal form.

The schema of such layer is similar to the scheme of the city arrangement in the path (Figure 1). Here each neuron has a set of braking T_{ij} and exiting W_{ij} connections. The braking connections determine state

when only one neuron can remain active in the each column and in the each row. Their weight factors are determined as:

$$T_{1ik} = \begin{cases} 0 & ,if \;\; i = k \\ -2/(n+1), & otherwise \end{cases} \quad (7)$$

$$T_{2ij} = \begin{cases} 0 & ,if \;\; i = j \\ -2/(p+1), & otherwise \end{cases} \quad (8)$$

where n is a number of neurons in the row, and p is a number of neurons in the column.

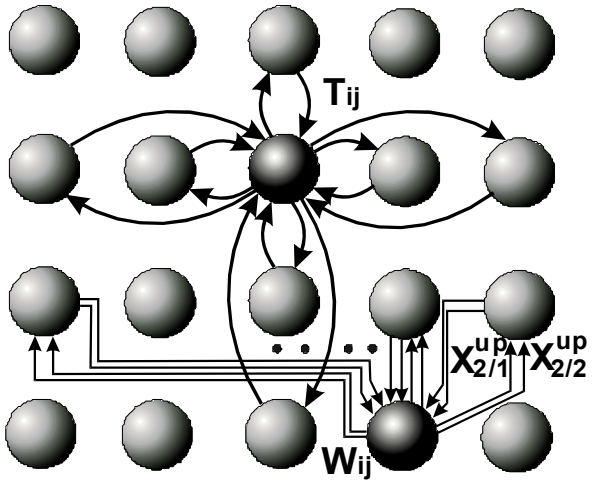

Figure 4. The structure of the layer forming the path-candidate: a connecting schema of neurons by the braking and by the exiting connections.

The formation of the shortest path in each layer is carried out according to the weight factors of exiting connections W_{ij} defined a priori:

$$W_{1ij/1} = W_{2ij/1} = \begin{cases} 0 & ,if \;\; i = j \\ 1/d_{ij} & ,otherwise \end{cases}, \quad (9)$$

$$W_{1ij/2} = W_{2ij/2} = 1 \quad (10)$$

where d_{ij} is a distance between i^{th} and j^{th} cities, W_{1ij} is a connection weight factor between a neuron of k^{th} row and a neuron of $(k\text{-}1)^{th}$ row, W_{2ij} is a connection weight factor between a neuron of k^{th} row and a neuron of $(k\text{+}1)^{th}$ row correspondingly. The weight factors W_{1ij} and W_{2ij} are the same for all rows of the layer. Thus the first and the last row of the layer does not have the exiting connections between themselves.

Each layer has two sorts of neurons: intermediate neurons are placed on intermediate rows of the layer (Figure 5) and final neurons are placed on the first and on the last row of the layer (Figure 6). Both sorts of neurons have a complex structure allowing forming the path-candidate in appropriate conditions for each layer.

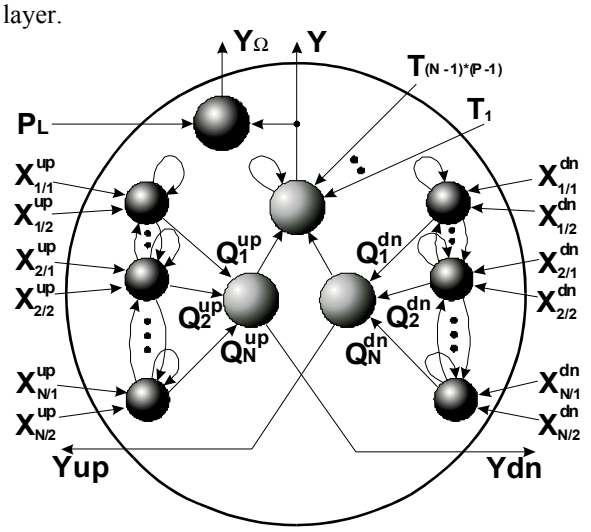

Figure 5. The scheme of the intermediate neural element.

The intermediate neuron (Figure 5) has two low-connected subsystems. Each of them obtains the information through exiting connections from neurons of the previous row and propagates it on the next row. Thus, the information propagates through each neuron both "top-down" and "down-top". The intermediate neuron contains two competing neural networks working similar to the neural network of the first layer (Figure 3). Their input values X_i are formed by product of output values of neurons of the previous row $Y_{up(dn)}$ on appropriate weight factor $W_{1(2)ij}$.

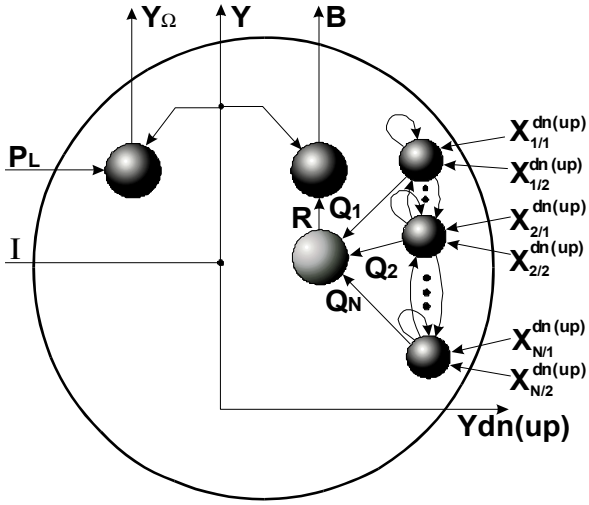

Figure 6. The scheme of the final neural element.

Thus, the dynamical process of the work of the competing neural network of a left part of the neuron (Figure 5) can be described as follows:

$$Y_i(0) = \frac{1}{X^{up}_{i/2} \Big/ X^{up}_{i/1} + 1 \Big/ X^{up}_{i/2}} \tag{11}$$

$$Y_i(t+1) = fn\left(\sum_j T_{ij} * Y_j(t)\right) \tag{12}$$

It is continued till the stable state when only one neuron remains active. After that we can obtain weight of path consisting of neurons of the previous rows of the layer:

$$Q^{up}_i = \begin{cases} \dfrac{1}{X^{up}_{i/2} \Big/ X^{up}_{i/1} + 1 \Big/ X^{up}_{i/2}}, if\ Y_i = 1 \\ 0, otherwise \end{cases} \tag{13}$$

The dynamical process for the right part of the neuron is the same.

The information for the subsequent rows of the layer is formed as follows:

$$Y_{up} = \sum_i^n Q^{dn}_i \tag{14}$$

$$Y_{dn} = \sum_i^n Q^{up}_i \tag{15}$$

After definition of activity of each neuron in the layer it is necessary to define in fact the path-candidate for the layer. For this purpose the rules are defined when only one neuron remains active in each row and in each column. Such state is defined by the relaxation neural network higher level. Equations (7) and (8) describe the weight factors of braking connections. The dynamical process of work is described for the neuron of i^{th} row and j^{th} column as follows:

$$Y_{ij}(0) = fn\left(\frac{1}{1 \Big/ Y_{up} + 1 \Big/ Y_{dn}}\right) \tag{16}$$

$$Y_{ij}(t+1) = fn\left(Y_{ij}(t) + fn2\left(\left(\sum_k^n T_{1ik} * Y_{kj}(t)\right) + \left(\sum_s^p T_{2sj} * Y_{is}(t)\right), Y_{ij}(t)\right)\right) \tag{17}$$

where

$$fn2(x,y) = \begin{cases} x, if\ y \geq 0.5 \\ 0, otherwise \end{cases} \tag{18}$$

The output value Y_W of each neuron is a part of the final solution of the neural network. It specifies activity of a neuron of the selected layer and it is determined as follows:

$$Y_\Omega = \begin{cases} Y, if\ P_L = 1 \\ 0, otherwise \end{cases} \tag{19}$$

where P_L is signal acting from the first layer. It determines *the layer-winner.*

In comparison with the intermediate neuron the finite neuron (Figure 6) contains only one relaxation network and an element forming a value B describing the length of the generated path. The work of the finite neural element can be described as:

$$R = \sum_i^n Q_i \tag{20}$$

$$Y = Y_{up(dn)} = I \tag{21}$$

$$B = \begin{cases} R, if\ Y = 1 \\ 0, othewise \end{cases} \tag{22}$$

Here I is an initialization value defined as:

$$I = \begin{cases} 1, if\ a\ city\ is\ fixed\ as\ active \\ \quad in\ the\ first\ and\ in\ the\ last \\ \quad row\ of\ the\ layer \\ 0, otherwise \end{cases} \tag{23}$$

At the end of the process forming the path in each layer the special elements form the value D_{LAYER} characterizing the length of the generated path (Figure 7).

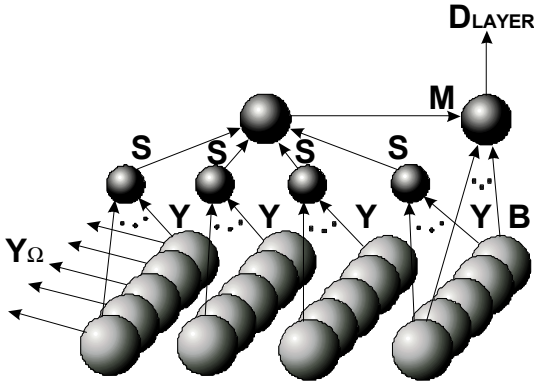

Figure 7. The schema of formation of the *LAYER* value.

It is determined as follows:

$$S_k = \sum_i Y_{ki} \tag{24}$$

$$M = \prod_k S_k \tag{25}$$

$$D_{LAYER} = \begin{cases} \sum_i^n B_i, if\ M = 1 \\ 0 \quad , otherwise \end{cases} \tag{26}$$

where k is a row index in the layer and M indicates the successful formation of the path-candidate in the layer.

The total number of neural elements of the neural network is:

$$T = \left(\sum_1^n i * n \right) - 1 = \frac{n^3}{2} + \frac{n^2}{2} - 1 \tag{27}$$

However the n layers is too much for the shortest path problem solution because ratio between the number of cities forming the shortest path and the total number of cities is quite little. Therefore we can reduce a number of layers by the following way:

$$E = \min\left\{ \left\lfloor 2 \times \sqrt{n} \right\rfloor + 1, n \right\} \tag{28}$$

3: Simulation and results

It was carried out the simulation of this neural network solving the shortest path problem on the sequential personal computer with processor Intel Pentium II-350 under control of the Windows NT v4.0 operating system. The modeling algorithm has been optimized for sequential computer to reduce the computation time. The model of the neural network was tested on solution of 4 sorts of tasks: 5, 15, 50 and 100 cities. The solution average time presented in seconds is shown in the Table 1 in comparison with other two famous methods for each sort of tasks.

Table 1.

Simulation method	***Size of the problem (a number of cities)***			
	5	**15**	**50**	**100**
Dynamic programming method	0	0	0	0
Dijkstra's algorithm	0	0	0	0
The neural network	0	0,02	1,58	22,03

As obvious from the Table 1 the model of the neural network works slower than pure numerical algorithms because it simulates analogue processes. Nevertheless it has formed good solutions for all 4 sorts of tasks (for example, see results in the Table 2 for the map presented on a Figure 8).

4: Conclusion

The neural network solving the shortest path problem is presented. It is used in the system forming the global route map during the motion in the unknown environment. The neural system has difficult pyramidal structure allowing forming simple shortest path to control efficiently by a mobile robot motion. This paper describes the architecture of such neural network. The experimental results have shown effectiveness of the proposed model.

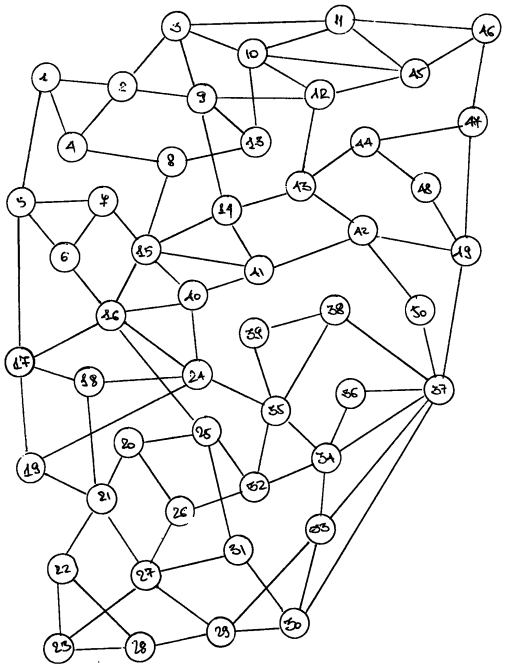

Figure 8. An example of the map of 50 cities.

Table 2.

Task num.	***Dynamic programming method***		***Dijkstra's algorithm***		***Neural network***	
	Solution	*Path length*	*Solution*	*Path length*	*Solution*	*Path length*
1	23-28-29-33-37-49-47-46	213	**23-27-26-32-34-37-49-47-46**	**209**	**23-27-26-32-34-37-49-47-46**	**209**
2	30-31-25-16-15-8-4	158	30-31-25-16-15-8-4	158	30-31-25-16-15-8-4	158
3	35-24-40-41-14-9-3	125	35-24-40-41-14-9-3	125	35-24-40-41-14-9-3	125
4	23-28-29-33-37-49-48-44	195	**23-22-21-19-17-16-15-14-43-44**	**190**	**23-22-21-19-17-16-15-14-43-44**	**190**

5	45-12-43-42-50-37-30	179	45-12-43-42-50-37-30	179	45-12-43-42-50-37-30	179
6	17-16-40-41-42-49	119	17-16-40-41-42-49	119	17-16-40-41-42-49	119
7	2-9-14-43-42-50-37	132	2-9-14-43-42-50-37	132	2-9-14-43-42-50-37	132
8	11-10-13-8-15-16-25-31	177	11-10-13-8-15-16-25-31	177	11-10-13-8-15-16-25-31	177
9	28-29-33-37	102	28-29-33-37	102	28-29-33-37	102

5: Acknowledgements

The given work is carried out within the framework of the project INTAS-BELARUS-97-2098 "Intelligent neural system for autonomous control of a mobile robot". The author thanks European Union for the informational and financial support.

References

1. Michael R. Garey and David S. Johnson. Computers and intractability// San Francisco, Bell Telephone Laboratories, Inc., 1979.
2. Dijkstra E.W. "A note on two problems in connexion woth graphs", *Numerische Mathemati*k, 1:269-271, 1959.
3. Bellman R. "On a routing problem", *Quarterly of Ap-plied Mathematic*s, 16(1):87-90, 1958.
4. Lestor R., Ford, Jr., and D.R. Fulkerson. *Flows in Networks,* Princeton University Press, 1962.
5. Ahuja R.K., Mehlhorn K., Orlin J.B., and R.E. Tarjan. "Faster algorithms for the shortest path problem". Technical report 193, MIT Operations Research Cen-ter, 1988.
6. Gabow H.N., and R.E. Tarjan. "Faster scaling algo-rithms for network problems", *SIAM Journal on Com-puting*, 18(5): 1013-1036, 1989.
7. Zhan F.B. "Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Proce-dures", *Journal of Geographic Information and Deci-sion Analysi*s, 1(1), 69-82, 1997.
8. J.J. Hopfield and D.W. Tank. Collective computation with continuos variable// Disordered systems and Biological organization, Springer-Verlag, 1986, pp. 155-170.
9. I.I. Melamed. Neural network and combinatorial optimization// In Proceedings of the 16th IFIP Conf. Syst. Model Optim., Compieqne, France, 1993, pp. 537-542.