A Neural Network for Combinatorial Problems of Optimization

Vladimir Golovko, S. Derechennik
Department of Computers and Mechanics,
Brest polytechnic institute, Moscowskaja 267,
224017 Brest, Republic of Belarus
cm@brpi.belpak.brest.by

Key Words: neural networks, optimization, NP-complete problems.

Abstract

A neural network architecture for the optimization problems is discussed. It is a feedforward neural network with fixed weights. Such approach consists in definition the weights, using a priori knowledge. This paper describes the neural network for solving optimization tasks.

1. Introduction

A lot of combinatorial tasks belong to the class of NP-complete problems. In this case optimum solutions can not be guaranteed to be found in polynomial time. Therefore a number of heuristics have been proposed to find approximate solutions [1,2]. Neural approaches based on different kinds of collective computation have gained interest [3-6]. This paper describes the neural network for combinatorial tasks. The architecture is inherently parallel and many units can carry out their computations at the same time. The principle of architecture and functionality of such network is shown on the example of solving the "shortest way" problem and knapsack problem.

2. The general architecture

The general architecture of the neural network is presented on Figure 1. It consists of 3 layers.

The input units receive data from outside the neural network and distribute these data to hidden units. The units in the hidden layer determine various variants of the decision of a task. A unit in the output layer is meant for definition of the optimum decision of a task:

$$Z_k = \text{opt}(y_j), \qquad (1)$$

where k – the number of the hidden unit, which identifies the optimum solution, Z_k – the optimum solution of a combinatorial task. Let's examine the decision of some optimization tasks.

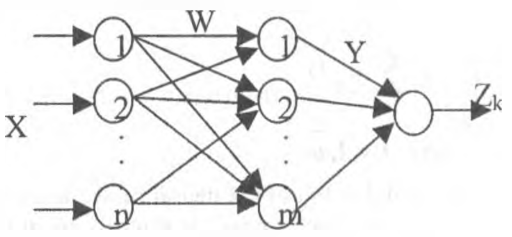

Figure 1. The general architecture of the neural network

3. The "shortest way" task

The "shortest way" problem consists in the following. Given n cities, initial and final points of route. The distances between various cities are known. What is the shortest way between the initial and final points of the route? Let's determine the structure of the neural network. The quantity of input units is given by the simple formula

$$p=(n-1)(n-2)+1, \qquad (2)$$

where n is number of cities.

The distance D_{ij} between the points of route i and j enters input units. It should be marked, that

$$i=\overline{1,n-1};\ j=\overline{1,n};\ i \neq j;\ j>i$$

for i=1.

So, if n=4, then p=7 and input vector of distances is given by

$$D=(D_{12},D_{13},D_{14},D_{23},D_{24},D_{32},D_{34}) \qquad (3)$$

Note that 1 characterizes initial and 4 – final points of route. The neurons in the hidden layer forms possible ways between initial and final points of the route. The quantity of the hidden units is as follows:

$$m=\sum_{i=1}^{n-1}\binom{n-2}{n-i-1}(n-i-1)!=(n-2)!\sum_{i=1}^{n-1}\frac{1}{(i-1)!} \quad (4)$$

where 0!=1.

It should be marked, that if n is very big, then we can obtain

$$\sum_{i=1}^{n-1}\frac{1}{(i-1)!}=e$$

In this case

$$m \approx e(n-2)!$$

Later on we shall represent the components of an input vector in linear system of coordinates

$$D=(D_1,D_2,\dots,D_p) \quad (5)$$

Then the outputs of the hidden units is computed as

$$y_j=\sum_{i=1}^{p} w_{ij}D_i$$

where $j=\overline{1,m}$

The weights W_{ij} of the neural network are formed so as to receive a set of possible routes. The dimension of weight matrix is equaled pxm. So, if n=4, then m=5 and the weight matrix is as follows:

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

It is binary, if any ways between initial and final points of route are possible. In case, when there is no way between some points of route it is necessary instead of the 1 to put infinity (∞). For example, if there is no way between the first and second points of the route, i.e. $D_{12}=\varnothing$, then $W_{11}=\infty$. Each column of the matrix W characterizes weight vector for the appropriate neuron. A unit in the output layer defines the shortest way

$$Z_k=\min_j\{y_j\}$$

where Z_k – the length of an optimum route; k – the number of an optimum route. If we know the number k of the winner, then the shortest way is possible to determine

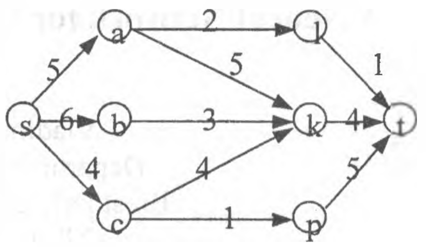

Figure 2. Source graph of ways

$$D_o=D^TW_k^T$$

where W_k – the weight vector of neuron k.

Example 1

Let a matrix distances between 4 cities be given. It is necessary to define the shortest way between 4 cities.

Tab. 8.1

	1	2	3	4
1	0	25	100	75
2	25	0	55	35
3	100	55	0	15
4	75	35	15	0

The input vector of distances is given by

D=(25, 100, 75, 55, 35, 55, 15).

Each neuron of the hidden layer defines one of the possible routes. Then

$y_1=25+55+15=95$

$y_2=25+35=60$

$y_3=100+35+55=190$

$y_4=100+15=115$

$y_5=75$

The output value of the neural network is

$$Z_k=\min_j\{y_j\}=60$$

k=2

The weights of the second neuron (hidden layer) correspond to the following route:

$$D_o=\begin{bmatrix}D_{12}\\D_{13}\\D_{14}\\D_{23}\\D_{24}\\D_{32}\\D_{34}\end{bmatrix}\begin{bmatrix}1\\0\\0\\0\\1\\0\\0\end{bmatrix}=\begin{bmatrix}D_{12}\\D_{24}\end{bmatrix}$$

As a result we can receive the optimum route:

$1 \to 2 \to 4$

The considered neural network is characterized by significant complexity. Therefore it can be used on final stages for search of optimum variant. Another way is the adaptation of a neural network to the algorithm of optimization, wich is used for the solution of the task. So, for example, if the dynamic programming is used for definition of the shortest way, then is easy to decrease the dimension of the neural network.

Let's consider the general approach of application of dynamic programming and neural network for the computation of the shortest way.

Let possible routes between an initial point S and final point T be given as the graph (Fig. 2). Such graph consist of four layers. Let's divide the task to steps and for definition of the optimum solution at each step we shall use the neural network. The first step coveres last three layers of the graph. Then it is necessary to find the optimum ways from points A, B and C to point T. For each of these tasks is formed the neural network. The number of cities is 4. Therefore the structure of the neural network will be the same, as in example 1. As a result of performance of the first step, we can receive

$$Z_2(a) = 3;\ \ a \to l \to t,$$
$$Z_1(b) = 7;\ \ b \to k \to t,$$
$$Z_4(c) = 6;\ \ c \to p \to t$$

Let's present the graph as follows (Fig. 3).

The neural network for n=5 is used for the definition of the optimum route from S to T. In this case

$Z_7(S) = 8$.

and route will be the following:

$s \to a \to l \to t$

If only real routes are taken into account, then the

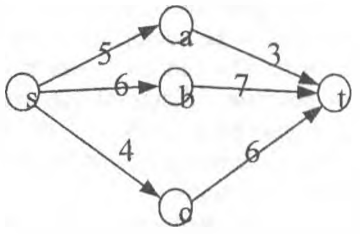

Figure 3. Reformed graph of ways

dimension of the hidden layer decreases. So, for n=5 the number of hidden neurons is

$m = 10$

If the real graph is taken into account (Fig. 2), then

$m=3$.

Thus, the use of the dynamic programming permits to decrease the dimension of the neural network.

4. The knapsack problem

The knapsack problem is NP-complete and is formulated as follows: (given a set of subjects $u = (u_1, u_2, ..u_n)$ and for each of them volume V(u) and cost C(u) are known. It is required to fill in the bag of limited volume T so, that to obtain maximum cost of the packed things or to make it more than K. The mathematical formulation of the knapsack problem can be presented as follows:

$$\sum_u V(u) \le T \quad \text{And} \quad \max_u \left\{ \sum_u C(u) \right\},$$
$$\sum_u V(u) \le T \quad \text{And} \quad \sum_u C(u) \ge K,$$

were N is a given cost. The general architecture of the neural network is shown on Fig 4.

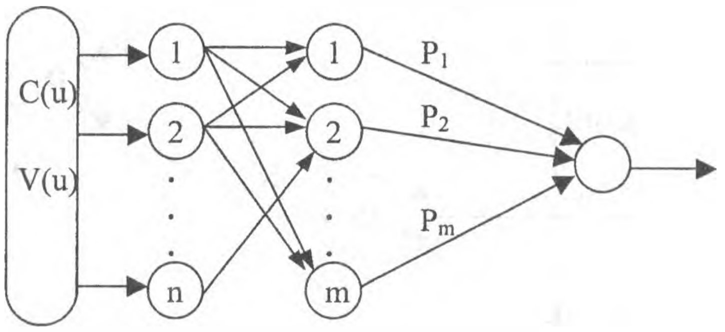

Figure 4. Neural network for solving the knapsack problem

It consist of three layers. The input information of the neural network is the vector of the cost C(u) and of the volume V(u). The hidden layers is intended for definition of possible variants of the decision of the knapsack problem. Each neuron of the hidden layer consists of three elementary neurons (Fig 5), which carry out various functions.

The element b_i generates i-th variant of the decision of the task. For this purpose the following function is calculated:

$$K_{bi} = \sum_{j=1}^{n} w_{ji} C(u_j) \tag{6}$$

The element d_i is the neural element with threshold function of activation. It analysis i-th variant of the decision as follows:

$$S_i = \sum_{j=1}^{n} w_{ji} V(u_j) - T \tag{7}$$

$$K_{ai} = \begin{cases} 1, & \text{if } S_i \leq 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

The element P_i performs the following function:

$$P_i = \begin{cases} K_{bi}, & \text{if } K_{ai} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

The output layer consists of one unit, which defines the optimum variant of the decision:

$$Z_k = \max_i \{P_i\} \tag{10}$$

where Z_k – the maximum cost of the subjects in the bag; k – the number of the unit, which identifies the optimum variant of the knapsack problem.

Using number k it is possible to define the final decision:

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \begin{bmatrix} w_{1k} \\ w_{2k} \\ \vdots \\ w_{mk} \end{bmatrix} \tag{11}$$

The number of the hidden units depends on the technology of the decision knapsack problem. Generally, if all variants of the decision are analyzed, then the hidden layer contains the following quantity of neural elements:

$$m = 2^n - 1. \tag{12}$$

Then the weights vector of i-th neuron corresponds to binary code of the number i: so, if i=5 and n=6

$W_5 = (00101)$

The weight matrix is formed so, that each column was able to characterize the weights of the certain neural element. So for n=3 and m=7 weight matrix is defined as follows:

$$W = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{13}$$

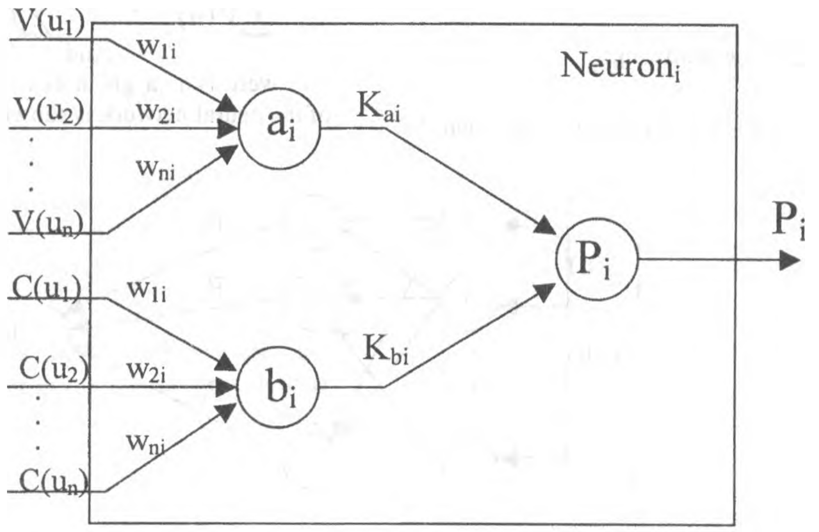

Figure 5. Scheme of neural element

The disadvantage of the above mentioned neural network is a great number of hidden units. For elimination of this disadvantage the algorithm of optimization to the neural network can be adapted, as it was in the previous section. Another way is the increase of the tacts of the work of the network. Suppose, there is a neural network consisting of 2^n neural elements. It is necessary to solve on such a neural network the task, which has dimension r, where r>n. Let

m=r-n

and the neural network, which contains r input units and 2^n-1 hidden units is given.

For the decision of knapsack problem (size r>n) on such a network it is necessary to spend 2^m tacts. The weight of neural elements will be changed in each tact according to the weight matrix.

Example 2

Let's consider the example of the decision of knapsack problem of dimension n=3. Let

$$V(k) = \{5,7,10\},$$
$$C(k) = \{3,2,1\},$$
$$T=17.$$

The quantity of neurons of the hidden layer is

$$m = 2^3 - 1 = 7$$

The weight matrix is determined according to expression(13). Then the results of calculations of neural elements of the hidden layer can be presented in the table 2.

Tab.2

№	Ka	Kb	P
1	1	1	1
2	1	2	2
3	1	3	3
4	1	3	3
5	1	4	4
6	1	5	5
7	0	6	0

Let's define the output value of the neural network:

$$Z_k = \max_i\{P_i\} = 5$$

k = 6.

According to number k of a neuron and its weight vector W_k we identify the decision of the knapsack problem:

$$U = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Thus optimum decision is achieved by packing in bag subjects u1 and u2.

5. Conclusion

In this paper the neural network for the optimization problems is described. It consists of 3 layers and allow to find global optimum. The weight matrix of the neural network is computed, using a priori knowledge. Such approach can be used for different combinatorial problems of optimization.

Literature

1. Lin, S.&B.W.Kernighan, "An effective heuristic for the travelling-salesman problem," *Optimization Research* 21(1973), 498-516.
2. Padberg, M.W.&G.Rinaldi, "Optimization of a 532-City Symmetric Travelling Salesman Problem by Branch and Cut," *Operations Research Letters* 6(1987), 1-7.
3. Burr, D.J. "An Improved Elastic Net Method for the Travelling Salesman Problem," in *IEEE International Conference on Neural Networks*#1, IEEE, New York, 1988, 69-76.
4. Hopfield, J.J.and Tank, D.W. "Collective computation with continuos variable." *Disordered systems and Biological organization* (Springer-Verlag), 1986, 155-170.
5. Melamed, J.J. "Neural network and combinatorial optimization," in *Proceedings of the 16th IFIP Conf. Syst. Model Optim.* (Compieqne, France), 1993, 537-542.
6. V. Golovko, H. Suhodolsky, V. Dimakov. "Neural Net for Combinatorial Optimization" *Proc. of intern. conf. "High Performance Computing"* (april, Boston) San Diego: The Computer Simulation International, - 1998.