Neural Algorithms for Solving Some Multi-Criterion Optimization Problems

Jerzy Balicki †,

Naval University of Gdynia, ul. Smidowicza 69, Gdynia, Poland

Summary
In this paper, artificial neural networks for solving multiobjective optimization problems have been considered. The Tank-Hopfield model for linear programming has been extended, and then the neural model for finding Pareto-optimal solutions in the linear multi-criterion optimization problem with continuous decision variables has been discussed. Furthermore, the model for solving quasi-quadratic multiobjective optimization problems has been studied. What is more, some models of the Hopfield neural network for solving NP-hard combinatorial multi-criterion optimization problems have been proposed. Finally, the family of extended Hopfield models for finding Pareto-optimal solutions has been developed..

Key words:
Neural networks, efficient solutions, multi-criterion optimization.

1. Introduction

A Hopfield neural network model HNN for linear programming has been proposed by Tank and Hopfield [14]. It can be modified for solving nonlinear optimization problems with continues decision variables [6]. So, we can apply the extended neural model for finding Pareto solutions to the multiobjective optimization problems.

Besides optimization problems with continuous decision variables, we can consider questions with binary variables. The first neural algorithm for solving the Traveling Salesman Problem has been proposed by Tank and Hopfield in [14], too. Moreover, the hybrid neural network model consisted of multiprocessor systems has been presented for finding solutions of TSP and successfully discovers solutions to the Hamiltonian Cycle by Sun and Fu in [13]. Some developments of Hopfield models for solving combinatorial optimization problems are proposed in [9]. For some other NP-hard problems, neural algorithms can be modified to find suboptimal solutions [15]. For instance, graphs problems and flows minimization problems are solved with using Hopfield models [7].

In this paper, the extended Hopfield models for solving some multi-criterion optimization problems are presented. To obtain this purpose some network models for standard constraints and objective functions are designed, separately. Then, synaptic connections of these networks are developed in one global network prepared for solving the special case of multiobjective optimization problem to obtain an efficient solution. This meta-heuristics is rather general and gives solution that is close to an optimum.

2. The Hopfield model of neural networks

In gradient models of HNN the neural activation states are changed from the initial state $u(t_0)=[u_1(t_0),...,u_m(t_0),...,u_M(t_0)]^T$ according to the below differentiable equations [2]:

$$\frac{du_m}{\mathrm{d}t} = -\frac{u_m}{\eta_m} + \sum_{n=1}^{N} w_{nm} g_m(u_m) + \mathrm{I_m} \quad \text{for } \mathrm{m} = \overline{1,\mathrm{M}}; \qquad (1)$$

where

M - the number of neurons,

u_m - the activation level of mth neuron, $m = \overline{1, M}$,

η_m - the positive passive suppress coefficient for the neuron with the output x_m ,

w_{nm} - the synaptic weight from the neuron x_n to the neuron x_m ,

I_m - the external input to the neuron x_m .

Matrix of synaptic weights W is *symmetric*. An important assumption, that

$$w_{mm} = 0, \quad m=\overline{1,M}, \qquad (2)$$

causes that the energy function there has no minimum in the interior of the output space. Moreover, external inputs and synaptic weights are constant during the process of energy function minimization. For optimization problems with continuous decision variables, the signals in neurons are transformed according to the linear activation function g_m , as follows:

$$x_m = u_m \quad \text{for} \quad m=\overline{1,M}; \qquad (3)$$

Manuscript received June 16, 2007
Manuscript revised June 25, 2007

Hopfield proposed the Liapunov function for the differential system (1) in respect to the below formula [8]:

$$E(u) = -\frac{1}{2}\sum_{n=1}^{M}\sum_{m=1}^{M} w_{nm} x_n x_m - \sum_{m=1}^{M} I_m x_m + \sum_{m=1}^{M} \int_0^{x_m} g_m^{-1}(\xi_m) d\xi_m , \quad (4)$$

where

g_m^{-1} - a reverse function for an activation function g_m, $u_m = g_m^{-1}(x_m) = x_m$.

3. Linear problem with continuous variables

Let the linear minimization problem be consider in the following form:

$$\min_{x \in X} \sum_{m=1}^{M} c_m x_m , \quad (5)$$

where

x_m - decision variables, $m = \overline{1, M}$,

c_m - cost coefficients, $m = \overline{1, M}$,

$$X = \left\{ x \in R^M : \sum_{m=1}^{M} a_{nm} x_m \geq b_n, \; n = \overline{1, N} \right\},$$

a_{nm} - constraint coefficients, n, $m = \overline{1, M}$.

Decision variables are continuous and their values is the same as outputs from neurons in the stable state of the network. The objective function $f(x)$ is linear. Constraints are linear with lower limits.

A linear programming problem can be transformed into the standard form with constraints modeled as equations. Some optimization techniques can be used for finding solutions to the problem [11]. For instance, the SIMPLEX method or the parallel COMPLEX method can be used for solving it. These methods are the fundaments for preparing some techniques in nonlinear optimization as well as multi-criterion optimization.

4. Neural model for linear optimization

Let us consider a constraint, as below:

$$\sum_{m=1}^{M} a_m x_m \geq b_n \quad \text{for given } n. \quad (6)$$

We introduce a denotation for a positive inequality resource:

$$r_n = \sum_{m=1}^{M} a_m x_m - b_n . \quad (7)$$

If $r_n \geq 0$, then inequality constraint is satisfied. In the other case, inequality constraint is not satisfied.

Unfeasible solution should be punished during network relaxation. It can be made by the following penalty function:

$$P_n(a_n^t x - b_n) = \begin{cases} 0.5(a_n^t x - b_n)^2 \; if \; a_n^t x - b_n < 0 \\ 0 \quad if \; a_n^t x - b_n \geq 0 \end{cases}, \quad n = \overline{1, N} \quad (8)$$

An energy function can be written, as follows:

$$E = c^t x + \sum_{n=1}^{N} P_n(a_n^t x - b_n) + \sum_{m-1}^{M} \frac{1}{\eta_m} \int_0^{x_m} \xi_m d\xi_m \quad (9)$$

The motion system can be obtained, as follows:

$$\frac{du_m}{dt} = -\frac{\partial E}{\partial x_m} \quad (10)$$

Above formula gives an analogy between the deepest descent method and Hopfield network. But, in Hopfield network a data processing is parallel, because each neuron plays a role of processor in computer. If another formula is applied, then the constant C_m is used on the left side of above formula. However, it does not influence on the equilibrium point of motion equations [1]. From (9) and (10), we get, as below:

$$\frac{du_m}{dt} = -\frac{u_m}{\eta_m} + \sum_{n=1}^{N} (-a_{nm}) \frac{\partial P_n}{\partial r_n} - c_m , \quad (11)$$

where

$$\frac{\partial P_n}{\partial xm} = \frac{\partial P_n}{\partial r_n} \frac{\partial r_n}{\partial x_m} = a_{nm} \frac{\partial P_n}{\partial r_n} .$$

Because of formula (8) the partial derivatives are given, as follows:

$$\frac{\partial P_n}{\partial r_n} = \begin{cases} (a_n^t x - b_n) \; for \, a_n^t x - b_n < 0 \\ 0 \quad for \, a_n^t x - b_n \geq 0 \end{cases}, \quad n = \overline{1, N} , \quad (12)$$

According to the Tank-Hopfield model for solving linear optimization problem, above function can be used as an activation function in constraint neurons. This model is denoted as LTH model.

Figure 1 shows the network for solving linear minimization problem can be presented. In this network two groups of neurons are considered. The first one consists of *M* neurons with linear activation functions. These neurons are called decision neurons. Decision neurons change their states according to the formula (11).

The output values x_m from mth neurons are sent to the second group of N neurons with activation functions expressed by formula (12). These neurons are called constraint neurons and are responsible for constraint satisfaction. Constraint neurons do not work as dynamic decision neurons, but they give output value on inputs force without delay. Indeed, only marked decision neurons are Hopfield neurons.

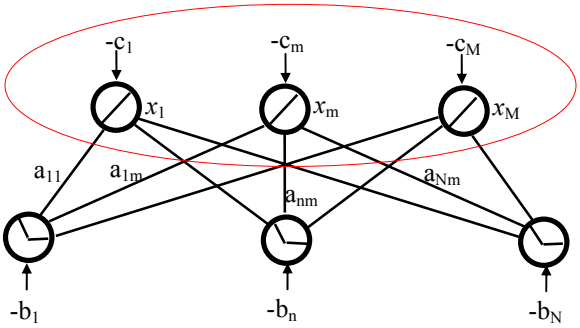

Fig. 1. The LTH model for linear minimization.

If the associated constraint is not performed, then the constraint neuron generates positive value proportional to the resource non-satisfaction of its. Gradient component of penalty function is given on the input of decision neurons, because output $-r_n(x)$ is multiplied by the mth constraint partial derivative a_{nm} . In this way to mth decision neuron the term $-\sum_{n=1}^{N} a_{nm}\frac{\partial P_n}{\partial r_n}$ is given. This term is the component of penalty function gradient, and leads to the growth of $r_n(x)$ for nth unsatisfied constraint. Constraint neurons and synaptic connections to decision variables from constraint neurons introduce a competition mechanism for neural networks. It transforms the requirement of constraint satisfaction into activation level modifications of decision neurons to satisfy all constraints. This undirected competition mechanism tends the network trajectory to the feasible solution set.

In decision neurons, the activation functions are linear, because the dynamic system (11) can be written in a simpler form, as below:

$$\frac{dx_m}{dt} = -\frac{x_m}{\eta_m} + \sum_{n=1}^{N} a_{nm}\left(-\frac{\partial P_n}{\partial r_n}\right) - c_m, \qquad (13)$$

This motion system can be solved by the linear Euler method or the first order Runge-Kutty method.

5. The extended LTH model

The considered LTH model has two disadvantages for some problem instances. During several numerical experiments we observe that for few cases LTH model converges to an interior of feasible solution set X, where an optimal solution is a vertex of a convex polyhedron. Moreover, for the other few cases the trajectory of network did not converge to a point from X. The amount of failed cases was no greater than few percent, but these disadvantages of the LTH model can be removed.

If the state $x(t)$ is in a feasible solution set X, then the dynamic system has a following form:

$$\frac{dx_m}{dt} = -\frac{x_m(t)}{\eta_m} - c_m, \qquad m = \overline{1, M}, \qquad (14)$$

because all constraints are satisfied and the term $-\sum_{n=1}^{N} a_{nm}\frac{\partial P_n}{\partial r_n}$ is equal to 0. So, on each decision neuron its external input $I_m=-c_m$ has an influence, only. If the trajectory $x(t)$ is in X, then passive suppress coefficients and costs coefficients exist, and they are such, that network obtains an equilibrium point in X, according to the following formula:

$$\frac{dx_m}{dt} = -\frac{x_m(t)}{\eta_m} - c_m = 0, \qquad m = \overline{1, M}, \qquad (15)$$

and

$$x_m^* = -\eta_m c_m, \quad m = \overline{1, M}, \qquad (16)$$

Let $\eta = \eta_m$, $m = \overline{1, M}$. Because the set X is constrained, then there is a passive suppress coefficient such, that

$$\eta c \notin X, \qquad (17)$$

For the given initial value of passive suppress coefficient the above condition is tested. If it is not satisfied, then this parameter is iterative increases until condition (17) is performed.

Another disadvantage of the LTH model is a fact, that for few cases the trajectory of network did not converge to a point from X. This non feasible way of network operating can be omitted by introducing a minimal penalty $A_n>0$ for non-satisfaction of nth constraint, which perform the below inequality estimation:

$$A_n \geq -\eta\frac{|-c|}{|\nabla_x r_n|}, \quad m = \overline{1, M}, \qquad (18)$$

where $|x|$ denotes the length of the vector x.

Let $A=\max\{A_n,\ n=1,...,N\}$. Figure 2 shows the transfer function for constraint neurons in the extended

LTH model. It is right-site continuous, and can be expressed as follows:

$$\frac{\partial P_n}{\partial r_n} = \begin{cases} A + (a_n^t x - b_n) \; if \; a_n^t x - b_n < 0 \\ 0 \; \; if \; a_n^t x - b_n \geq 0 \end{cases}, \quad n = \overline{1, N}, \quad (19)$$

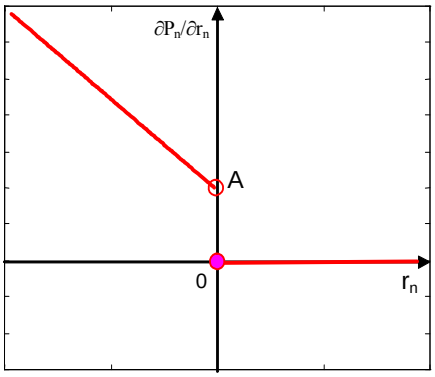

Figure 2. Transfer function for constraint neurons in extended LTH model

6. Multi-criterion linear optimization by PLNN

.

The PLNN model can be considered for solving multiobjective optimization linear problems, where K linear criteria $F_1,...,F_k,...,F_K$ are simultaneously minimized. The Pareto solutions, domination solutions, or compromise solutions can be found. For aggregation of all criteria in one global criterion the nonnegative convex combination function can be used, as below:

$$\min_{x \in X} \{ \sum_{k=1}^{K} \alpha_k F_k(x) \}, \quad (20)$$

where $\alpha_k \geq 0 \;\; k = \overline{1, K}, \;\; \sum_{k=1}^{K} \alpha_k = 1$.

Additional constraints can be modified $\alpha_k \geq 0 \;\; k = \overline{1, K+1}, \;\; \sum_{k=1}^{K+1} \alpha_k \geq 1$, where α_{K+1} denotes dummy variables. So, we consider M decision variables x_m, K unknown nonnegative convex combination coefficients α_k, and dummy variables α_{K+1}, which are represented by $M+K+1$ decision neurons with linear activation function. Neuron x_m is connected with neuron α_k by synaptic weights $-0.5c_{km}$ denoting cost coefficient in kth criterion for mth variable. Additional constraints are respected in network by the same way as linear constraints in the extended LTH model.

7. Multiobjective quasi-quadratic programming by the PQNN model

The PQNN model of neural network is proposed for solving multiobjective optimization problems, where K criteria $F_1,...,F_k,...,F_K$ are quasi-quadratic functions defined, as follows:

$$F_k(x) = -0.5 x^T D_k x - c_k^T x, \quad k = \overline{1, K}, \quad (21)$$

where D_k - $M \times M$ matrix of real numbers for function F_k.

This function is called quasi-quadratic, in opposite to quadratic, because $c_{mm}=0$ for $m=1,...,M$.

Figure 3 shows the scheme of the PQNN for finding a Pareto solution. The interactive mechanism for minimization $-0.5\alpha_k x^T D_k x$ is missed [3].

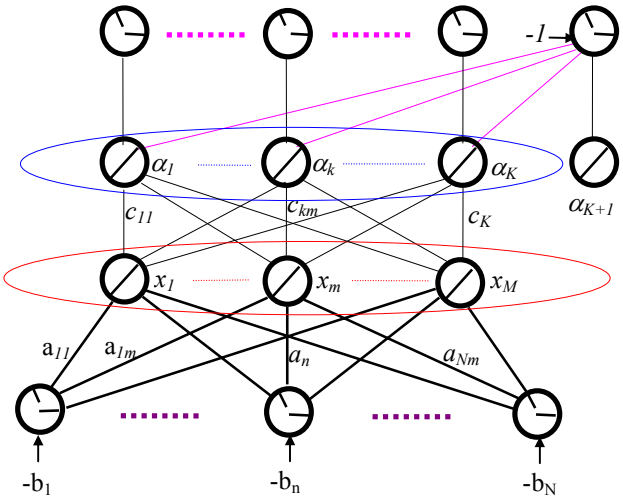

Fig. 3. The PQNN model for finding Pareto solutions.

8. Some optimization problems with zero-one decision variables

Let consider the multiobjective optimization problem (X,F,P) for finding the Pareto-optimal solutions, which are some allocations of program modules and processor types in a computer network. This problem seems to be very representative for the other combinatorial problems as it

was shown in [4]. In this problem some denotations are used, as follows:

1) X - a feasible solutions set

$$X=\{x, B^{2V+J} \mid x=(x_{11},...,x_{vi},...,x_{V2}, x_{11}^{\pi},...,x_{ij}^{\pi},...,x_{2J}^{\pi})^{T};$$
$$\sum_{i=1}^{2} x_{vi}=1, v=\overline{1,V}, \sum_{j=1}^{J} x_{ij}^{\pi}=1, i=\overline{1,2};\}$$

2) F - a vector quality criterion

$$F: X \rightarrow R^2, \quad F(x)=[F_1(x), F_2(x)]^T \text{ for } x \in X$$
$$F_1(x)=\sum_{i=1}^{2}\sum_{j=1}^{J} \kappa_j x_{ij}^{\pi},$$
$$F_2(x)=\sum_{j=1}^{J}\sum_{v=1}^{V}\sum_{i=1}^{2} t_{vi} x_{vi} x_{ij}^{\pi}+\sum_{v=1}^{V}\sum_{u=1}^{V}\sum_{i=1}^{2} \tau_{vu} x_{vi}(1-x_{ui}),$$

3) P - the Pareto relation [3]

In the constraint $\sum_{i=1}^{2} x_{vi}=1$ for $v=\overline{1,V}$ the following denotations are used. x_{vi} is the decision variable, which is equal to 1 if the program module m_v is assigned to the processing node w_i. V is the number of all program modules. Above constraint can be written in a general form $\sum_{m=1}^{M} x_m = L$, where x_m is a typical binary variable, and it should be satisfied for $M \geq L$. It means that only L variables can be equal to 1, and the other variables M-L should have value 0. There are no preferences which decision variables should be taken. This constraint is related with the requirement to program modules should be assigned or to the node w_1 or to the node w_2.

The general form of constraints $\sum_{m=1}^{M} x_m = L$ represents several constraints of combinatorial problems. For instance, in the Traveling Salesman Problem [14] during L (L is the number of all cities) steps a salesman should come through each city c_i exactly ones, what can be written as $\sum_{k=1}^{L} x_{ik}=1$ for $k=\overline{1,L}$, where x_{ik} is equal to 1 if the salesman in the kth step is in the city c_i. The length of the salesman travel is supposed to be equal to L steps, and if the constraints $\sum_{k=1}^{L} x_{ik}=1$ for $k=\overline{1,L}$ are satisfied, then this requirement is performed, too.

Similar constraints are in the vehicle routing problem VRP, the 0-1 knapsack problem, the set covering problem or the standard assignment problem [8]. They can be expressed in a general form $\sum_{m=1}^{M} x_m = L$.

The Hopfield model of artificial neural networks (called HNN) can be used for performing above constraint and the other sorts of constraint [10]. This time, signals in a neuron are (1) transformed according to an activation function:

$$g_m(u_m)=\frac{1}{2}[1+\tanh(\alpha_m u_n)], m=\overline{1,M}, \qquad (22)$$

where $\propto_m$ is the gain coefficient in the mth neuron ($\alpha_m \geq \alpha_{gr} > 0, m=\overline{1,M}$).

Uniform Hopfield networks called UHNN perform important role for satisfaction of the special class of constraints expressed in a general form [5]. For uniform Hopfield networks, all main parameters have the same value for each neuron, as below:

$$w_{nm}=w \text{ for } n,m \in \overline{1,M},$$
$$I_m=I \text{ for } m=\overline{1,M},$$
$$\eta_m=\eta \text{ for } m=\overline{1,M},$$
$$\alpha_m=\alpha \text{ for } m=\overline{1,M}.$$

The Euler method can be used for solving the motion equations of UHNNs. The following iterative procedure for finding active level value in the next moment is applied:

$$u_m(t_k+\Delta t)=(1-\Delta t)\frac{u_m(t_k)}{\eta_m}+w\Delta t\sum_{\substack{n=1 \\ n\neq m}}^{M} x_n(t_n)+I\Delta t, m=\overline{1,M}, \qquad (23)$$

In the Euler method, the integrity step length Δt should be taken as small as possible to avoid errors related with the approximation of differentiable equations. But, if Δt is too small, then the iterations number should be large to obtain the equilibrium point or saddle points.

To satisfy the considered constraint $\sum_{m=1}^{M} x_m = L$ the special case of UHNN can be used according to the below theorem. This theorem gives values of main parameters of UHNN such as the neurons number, synaptic wages, and external inputs.

Theorem. 1 [5]
If

$$\sum_{m=1}^{M} x_m = L \text{ for } x_m \in \{0,1\}, L \le M, L = 0,1,2,...,M,$$

and the uniform Hopfield network has following parameters:

$$\begin{aligned} w_{mj} &= -2\,m,\ j = \overline{1,M},\ m \ne j \\ I_m &= 2L - 1m = \overline{1,M} \end{aligned} \tag{24}$$

where M is the number of neurons,
then

$$E(x)=h(x)$$

where

$E(x) = -\frac{w}{2}\sum_{n=1}^{M}\sum_{m=1}^{M} x_n x_m - I\sum_{m=1}^{M} x_m$ is a basic energy function of this UHNN,

$h(x) = (L - \sum_{m=1}^{M} x_m)^2 + \sum_{m=1}^{M} x_m(1 - x_m)$ is a penalty function for constraint.

UHNN with synaptic weights equal to -2 and nonnegative external inputs calculated according to the rule I=2L-1 can be called UHNN/L/M, because the pair (L,M) is required for the designing of UHNN/L/M, only. Signals from the other neurons are converted and their absolute value is increased. Moreover, each neuron has its nonnegative constant input, which forces the activation level $u^*=\eta I$ in the equilibrium point for the winner neurons.

All cases of a network UHNN/L/M (M≤100) were studied by numerical experiments. For generating of initial states, the following formula was used:

$$u_m(t_0) = \frac{M - i + 1 - \frac{M}{2}}{10},\ m = \overline{1,M}, \tag{25}$$

Moreover, we assume, that α=100, Δt=0.2, and η=1. A stop criterion k_{stop} is the condition $E(t_k) \le \varepsilon$, where ε=0.01. For the worst case, the iteration number $K_{max}(E(t_k) \le \varepsilon)$ for solving a general equation is equal to 5. These experimental results confirm that neural networks can be designed as a very efficient method for solving some numerical problems. For the network UHNN/L/M the neurons number M does not influence on the increasing of $K_{max}(E(t_k) \le \varepsilon)$. For 1000 decision variables, and L=600, $K_{max}(E(t_k) \le \varepsilon)$=4.

9. HNN/F1/C for linear constrained minimization

A following optimization problem is studied:

$$\min_{x \in X} \sum_{i=1}^{2} \sum_{j=1}^{J} \kappa_j x_{ij}^{\pi} \tag{26}$$

Two separated modified networks HANN/1/J can be used. They satisfy the constraints $\sum_{j=1}^{J} x_{ij}^{\pi} = 1,\ i = \overline{1,2}$. In these networks external inputs are modified according to the formula

$$I(x_{ij}^{\pi}) = 2J + 1.5 - \Delta I(x_{ij}^{\pi}),\ j = \overline{1,J}, i = \overline{1,2},$$

where

$$\Delta I(x_{ij}^{\pi}) = \frac{\kappa_j}{\kappa_{\max}},$$

$\kappa_{\max}$ is the cost of the most expensive processor.

Above formula is related to the conclusion, that if in UHNN/L/M one neuron has the external input greater than the others, then this chosen neural output gets 1 in an equilibrium point. So, this is a way for preferring neurons related with the cheaper processors. Therefore, the additional term decreases the external inputs when the cost increases. If in a network UHNN/L/M all external inputs are increased about small value, then still L neurons are chosen in an equilibrium point. For L=0, there is I=2L-1=-1. For L=1, there is I=1. So, according to the bounder increasing L=0,1,2,3,4,... , there is an input increasing I=-1,1,3,4,6,... . For the bounder L is the interval for feasible external inputs (2L-2,2L). In this interval, the changed external inputs should have they value. For each node number, two separate networks UHNN/1/J are considered.

10. HNN/F2/C for quasi-quadratic constrained minimization

If a more complex problem is considered, when the time criterion F_2(.) is minimized with respect to constraints. This optimization problem can be transformed to the unconstrained optimization problem. Energy functions of neural networks designed for constraint satisfaction or for objective function minimization can be aggregated in a penalty function, as below:

$$E(x,\beta) = F_2(x) + \sum_{v=1}^{V} \beta_v E_v(x) + \sum_{i=V+1}^{V+2} \beta_i E_i(x), \tag{27}$$

where
β_v, β_i- penalty coefficients,
E_v- the energy function of the network UHNN/1/2 for a satisfaction of the constraint $\sum_{i=1}^{2} x_{vi} = 1$,
E_i- the energy function of a network UHNN/1/J for a satisfaction of the constraint $\sum_{j=1}^{J} x_{ij}^{\pi} \le 1$.

Figure 4 shows a network HNN/F_2/C for minimization for finding solution to the studied problem. Penalty coefficient can be found by systematically increasing from the initial value β_0 (usually equals 1). If in the equilibrium point one of the energy function related to the vth constraint is greater than 0, then this constraint is not satisfied, and its penalty parameter is increased about $\Delta\beta$ (usually about 0.05). This process is stopped, if all energy functions of constraints are equal to zero.

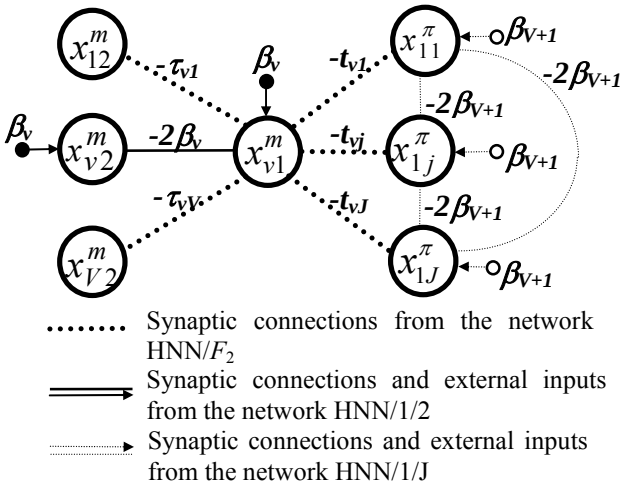

Fig. 4. Synaptic connections and external inputs for the neuron x_{v1}^{m} in HNN/F_2/C

If $\Delta\beta$ is too large, then a feasible solution will be found, but for smaller values the influence of a goal function is greater than the influence of penalties functions on the neural network trajectory. Then chances of finding an optimal solution or a suboptimal solution are much greater. From the other hand, smaller value of $\Delta\beta$ causes the longer time of a network simulation.

11. Some experimental examples

Model HNN/F_2/C was tested by several numerical examples. Figure 5 shows the energy function trajectories for the following optimization problem instance.

V=4, J=2, I=2, $T = \begin{bmatrix} 2,5 & 1,3 \\ 1,4 & 8,9 \\ 1,2 & 5,2 \\ 2,2 & 1,1 \end{bmatrix}$, $\tau = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

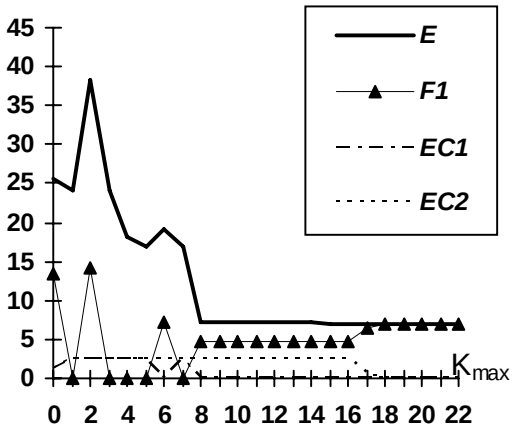

Fig. 5. Minimization the energy function

The number of decision variables is $M=I(V+J)=12$. Moreover, the following network parameters are taken: $\alpha \in \{100,200,300,400,500\}$, $\eta=1$, $\Delta t=0.2$, $\varepsilon<0.005$. Model started from the initial activation level state:

$$u(t_0)= [10^{-6}, 0, 10^{-6}, 0, 10^{-6}, 0, 10^{-6}, 0, 10^{-6}, 0, 10^{-6}, 0]^T.$$

An optimal value of the objective function $F_2(x^*)$=7.0 was obtained for β=[2.55, 2.45, 2.25, 2.25, 7.35, 7.30]^T , and for α=200 in an equilibrium point x^*=[1,0,0,1,0,1,1,0, 0,1,0,1,0,0]^T.

If the penalty coefficient of a partial network is increased, then the role of this network increases, too. Then, signals generated by this network try to dominate signals from the other network.

For the correct value of the penalty vector β the optimal solution was found after 18 steps of updating states in the neural model. Unfortunately, for each case the right value of the penalty vector β should be sought.

12. Neural model for Pareto allocations

For finding a local efficient point of that problem we can use the Hopfield ANN (PHANN). PHANN can represent one Pareto-optimal solution in equilibrium point. An energetic function for PHANN are constructed as below:

$$E(x,\beta)=\sum_{n=1}^{N}\alpha_n F_n(x)+\sum_{v=1}^{V}\beta_v E_v(x)+\sum_{i=V+1}^{V+2}\beta_i \mathrm{E_i}(x). \quad (28)$$

For N=2, the combination coefficients can be systematically changed from a period (0,1). The objective function and penalty functions can be presented by the separate partial energetic functions. For the objective function in non-negative convex combination method, we get the following formula of separate energetic function:

$$\sum_{n=1}^{N}\alpha_n F_n(x)=-\frac{1}{2}\sum_{r=1}^{M}\sum_{m=1}^{M}w_{rm}^{n}x_r x_m-\sum_{r=1}^{M}I_r^n x_r, \quad (29)$$

where

w_{rm}^{n} - the synaptic weight from rth neuron to mth neuron related with the multiplied objective function $E_n(x,\alpha_n)=\alpha_n F_n(x)$

I_r^n -the external input of rth neuron related with the multiplied objective function $E_n(x,\alpha_n)=\alpha_n F_n(x)$

In the similar way, we can obtain the formulas of partial energetic functions for the constraint satisfaction. Hence, we have the global basic energetic function of PHANN:

$$E=-\frac{1}{2}\sum_{r=1}^{M}\sum_{m=1}^{M}\Big(\sum_{n=1}^{N}\alpha_n w_{rm}^{g}+\sum_{l=1}^{V+2}\beta_l w_{rm}^{l}\Big)x_r x_m-\sum_{r=1}^{M}\Big(\sum_{n=1}^{N}\alpha_n I_r^n+\sum_{l=1}^{V+2}\beta_l I_r^l\Big)x_r, \quad (30)$$

In above formula, combination coefficients are systematically fixed from 0 to 1. If α_1=1 and α_2=0, then we get a network HNN/F_1/C. If α_1=0 and α_2=1, then we get a network HNN/F_2/C. Similarly, we can obtain the others synaptic weights and external inputs, which are related with the others constraints and the objective function.

13. Concluding remarks

Numerical experiments with presented neural models were carried out in the PC environment without neural accelerators. It is possible to use the neural accelerators and improve the performance of the neural calculating. Simulations of considered neural networks confirmed, that neural networks are competitive techniques with standard numerical methods.

Above approach can be used for nonlinear programming with linear constraints by a simply modification of the LTH model. In this case the synaptic connections between decision neurons are added. For multiobjective optimization the PLNN model and the PQNN model can be adapted for solving hierarchical solutions or compromise solutions with parameter p.=1.

Moreover, neural algorithms for solving several optimization problems of operation allocation have been proposed. Formulas to determine values of synaptic weights and external inputs for networks, which satisfy basic constraints and objective functions, are presented.

The recurrent HNN for optimization can be combined with genetic algorithms. Therefore, a hybrid genetic-neural algorithm seems to be a very powerful tool for solving combinatorial problems. From that reason, we will focus on solving this problem in our future works.

References

1. S. Abe, *Convergence Acceleration of the Hopfield Neural Networks by Optimizing Integration Step Sizes*, IEEE Trans. on Systems, Man, and Cybernetics. Part B: Cybernetics. Vol. 28, No. 1, February 1996, pp.194-201.
2. S.V.B. Aiyer, H. Niranjan and F. Fallside, *A theoretical investigation into the performance of Hopfield model*. IEEE Tran. on Neural Networks, Vol. 1, No. 3, pp. 204-215, (1990).
3. A. Ameljañczyk, *Multicriteria optimization*, WAT, Warsaw 1986.
4. J. Balicki, *Negative selection with ranking procedure in tabu-based multi-criterion evolutionary algorithm for task assignment*, Lecture Notes in Computer Science. Vol. 3993, Springer-Verlag, Berlin Heidelberg New York, pp. 863-870, (2006).
5. J.Balicki, Z. Kitowski and A. Stateczny, *Extended Hopfield Model of Neural Networks for Combinatorial Multiobjective Optimization Problems*. Proceedings of 1998 IEEE World Congress on Computational Intelligence, Anchorage, May 4-9, 1998, Vol. 2, pp. 1646-1651, (1998).
6. A. Cichocki, R. Unbehauen, „*Neural networks for solving systems of linear equations and related problems"*, IEEE Trans. on Circuits and Systems, Vol. 39, No.2 1992, pp.124-137.
7. N. Funabiki, J. Kitamichi and S. Nishikawa, *An evolutionary neural network algorithm for max cut problems*. Proceedings of the 1997 International Conference on Neural Networks, Vol. 2, Houston, USA, June 9-12, pp. 1260-1265, (1997).
8. D. Koznachey, A. Jagota and S. Das, *Primal-target neural net heuristics for the hypergraph K-coloring problem.* Proceedings of the 1997 International Conference on Neural Networks, Vol. 2, Houston, USA, June 9-12, pp. 1251-1255, (1997).

9. W.E. Lillo, S. Hui and S.H. Żak, *Neural networks for constrained optimization problems.* Int. J. of Circuit Theory and Applications, Vol. 21 pp. 385-399, (1991).
10. J. Mandziuk, *Pulsed noise-based stochastic optimization with the Hopfield model.* Proceedings of the 1997 International Conference on Neural Networks, Vol. 2, Houston, USA, June 9-12, pp. 1315-1320, (1997).
11. C.R. Reeves, *Modern heuristic techniques for combinatorial problems*. McGraw-Hill Book Company, London, (1995).
12. D.W.Tank and J.J. Hopfield, *Simple "neural" optimization networks: an A/D converter, signal decision circuit, and linear programming circuit*, IEEE Trans. on Circuits and Systems, Vol. CAS-33, pp. 533-541, (1986).
13. S. Salcedo-Sanz and C. Bousoño-Calzón, *A hybrid neural-genetic algorithm for the frequency assignment problem in satellite communications*, Applied Intelligence, Vol. 22, No. 3, pp. 207-217, (2005).
14. K.H. Sun and H.C. Fu, *A hybrid neural model for solving optimization problems*. IEEE Trans. on Computers, Vol. 42. No. 2, pp. 219-227, (1993).
15. D.W. Tank, J.J. Hopfield, *Simple "neural" optimization networks: an A/D converter, signal decision circuit, and linear programming circuit,* IEEE Trans. on Circuits and Systems, vol. CAS-33, pp. 533-541, May 1986.
16. J. Żurada, *Introduction to artificial neural systems*. West Publishing Company, USA, (1993).

Jerzy Balicki received the M.S. and Ph.D. degrees in Computer Science from Warsaw University of Technology in 1982 and 1987, respectively. During 1982-1997, he stayed in Computer Center of High School of Gdynia to study management systems, mobile systems, and decision support systems. Then, he achieved habilitation from Technical University of Poznan in 2001. He was admitted as a professor at Naval University of Gdynia in 2002.