Pareto Neural Model for Finding Task Allocation

JERZY BALICKI, ZYGMUNT KITOWSKI
Computer Science Laboratory, The Naval University of Gdynia
81-103 Gdynia, ul. Smidowicza 69, Poland

Abstract:- In this paper, the Hopfield model of artificial neural networks called HANNs for finding some task allocations in multiple computer systems have been proposed. A multiobjective optimisation problem with two criteria has been considered. Resource constraints have been assumed, too. Both the cost of parallel program execution and the cost of computers have been minimised. Two models of neural networks for minimisation of the computer cost and for minimisation of the cost of parallel program execution have been designed. Moreover, HANN for finding local Pareto-optimal solutions has been considered. Finally, some simulation results related to minimisation of the energy function for constructed neural networks have been included. Especially, a trajectory of energy function obtained during finding Pareto-optimal task allocation has been presented.

Key words:- Hopfield neural networks, multiobjective optimisation, task assignment

1. Introduction

Task allocation in multiple computer systems may decrease the total time of program execution by taking advantage of the specific efficiencies of some computers. In a network of workstations or personal computers, two or more program modules may execute concurrently for various periods. A program module is a collection of procedures or subroutines, or could be data files. A task is an execution of a program module.

In a distributed computer system, another way for the minimisation of the total time of program execution is a replacement of computers for data processing. So, the computer with a powerful floating-point unit is supposed to be dedicated for tasks with numerical procedures. Similarly, the graphic workstation is suitable for program modules with animation and chart processing.

In this paper, the Hopfield model of artificial neural network – called HANN – is proposed for finding a task allocation in distributed systems. The Hopfield models have been applied for solving some NP-hard optimisation problems [1, 6]. Two-objective optimisation problem with resource constraints is considered. The cost of parallel program execution and the cost of computers are minimised in this problem. The non-negative convex combination technique is developed for aggregate partial criteria. Two models of neural networks for minimising the computers cost and the cost of parallel program execution are constructed. Synaptic weights and external inputs are determined by comparison between an energy function of network and an objective function. Finally, HANN for finding a Pareto-optimal solution is considered.

2. Basic model of parallel processing

The standard problem of task assignment deals with finding the allocation of program modules to minimise the program execution cost [5]. An objective of optimisation can be the time of program execution, too. Another measure is the amount of computer resources utilized.

Figure 1 shows an example of a distributed program in a computer network consists of 5 computers. The program is divided into 11 modules M_1, M_2,..., M_{11}. Module M_7 is assigned to a computer with module M_8. The above task assignment is reasonable, if the number of interactions between pairs of these modules is relatively high.

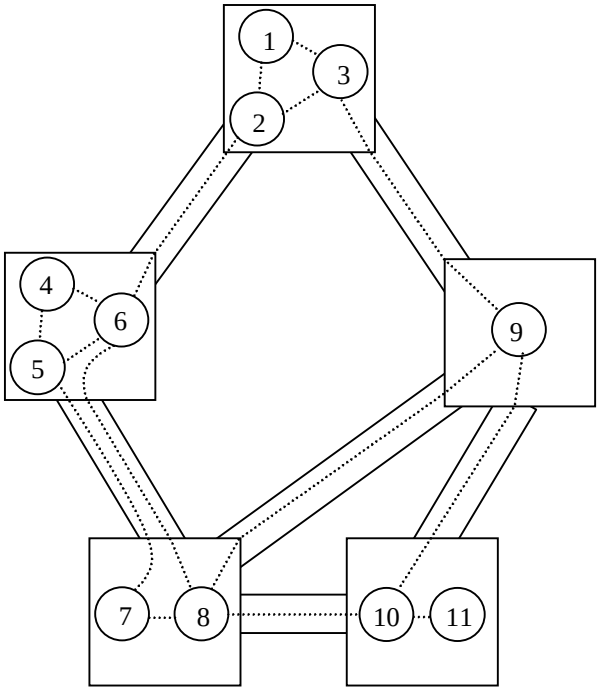

Figure 1: An example of distributed program that consists of 11 modules assigned in multiple computer system with 5 computers

A program module can be activated several times during the program lifetime. The process (operation, task) is the execution of one activated program module. Some processes can be associated with the module. As a result a set of program modules $\{M_1,...,M_m,...,M_M\}$ is mapped to a set of tasks $\{m_1,...,m_v,...,m_V\}$.

2.1. Computer allocation constraints

Let the task m_v be executed on several computers taken from the set $\Pi = \{\pi_1,...,\pi_j,...,\pi_J\}$. Computers are supposed to be assigned to the fixed nodes that belong to the set $W = \{w_1,...,w_i,...,w_I\}$. Computers located in different nodes can be characterised by the same sort. For example, two computers classified as the sort π_j can be assigned both to the node w_i and w_k.

Computers situated in nodes can communicate to each other to support program module interactions. Furthermore, one and only one computer should be allocated in each node. This implies the computer allocation constraints, as follows:

$$\sum_{j=1}^{J} x_{ij}^{\pi} = 1,\ i = \overline{1,I} \tag{1}$$

where

$$x_{ij}^{\pi} = \begin{cases} 1 \text{ if } \pi_j \text{ is assigned to } w_i \\ 0 \text{ in the other case,} \end{cases} \quad i = \overline{1,I},\ j = \overline{1,J}.$$

A vector x^{π} describes an allocation of computers:

$$x^{\pi} = [x_{11}^{\pi},...,x_{1j}^{\pi},...,x_{1J}^{\pi},...,x_{i1}^{\pi},...,x_{ij}^{\pi},...,x_{iJ}^{\pi},...,x_{I1}^{\pi},...,x_{Ij}^{\pi},...,x_{IJ}^{\pi}]^T \tag{2}$$

2.2. Task allocation constraints

An optimal task allocation should be found for minimising the parallel program execution time. A vector can determine the task allocation as follows:

$$x^m = [x_{11}^m,...,x_{1i}^m,...,x_{1I}^m,...,x_{vi}^m,...,x_{VI}^m]^T \tag{3}$$

where

$$x_{vi}^m = \begin{cases} 1 \text{ if task } m_V \text{ is assigned to } w_i, \\ 0 \text{ in the other case,} \end{cases} \quad v = \overline{1,V},\ i = \overline{1,I}.$$

Because each task ought to be allocated to any node, then the task allocation constraints are formulated, as below:

$$\sum_{i=1}^{I} x_{vi}^m = 1,\ v = \overline{1,V} \tag{4}$$

Now, the following vector can represent the task allocation to computers:

$$x = [x_{11}^m,...,x_{1i}^m,...,x_{1I}^m,...,x_{vi}^m,...,x_{VI}^m, x_{11}^{\pi},...,x_{1j}^{\pi},...,x_{1J}^{\pi}, ...,x_{i1}^{\pi},...,x_{ij}^{\pi},...,x_{iJ}^{\pi},...,x_{I1}^{\pi},...,x_{Ij}^{\pi},...,x_{IJ}^{\pi}]^T \tag{5}$$

2.3. Resource constraints

Each computer is supposed to have required amount of resources for a program execution. After loading, a program module reserves an operational memory. If the reserved memory size is changed during a module run, then a maximal amount is estimated. Another resource is the capacity of hard discs. If modules share the other sort of memories (a tape memory, the ZIP memory, etc.), then the capacities of memories cannot be exceeded. The following memories $z_1,...,z_r,...,z_R$ are available in the distributed computer system. Computers can be equipped with different amounts of memories. Let d_{jr} be the capacity of memory z_r in the computer π_j. The value d_{jr} is nonnegative and limited. We assume that the task m_v reserves c_{vr} units of memory z_r and holds it during a program execution. The value c_{vr} is nonnegative and limited, too.

The memory limit in any computer assigned to the ith node cannot be exceeded. This constraint is formulated as bellows:

$$\sum_{v=1}^{V} c_{vr} x_{vi}^m \le \sum_{j=1}^{J} d_{jr} x_{ij}^{\pi},\ \ i = \overline{1,I},\ r = \overline{1,R} \tag{6}$$

A program module may require the subroutine library, a specific software environment, a DVD driver, a high-resolution monitor, or the other components. Let $k_1,...,k_s,...,k_S$ denote the required components. We assume that the following component coefficients are given:

$$c'_{vs} = \begin{cases} 1 \text{ if } m_V \text{ requires the component } k_s, \\ 0 \text{ in the other case,} \end{cases} \quad v = \overline{1,V},\ s = \overline{1,S}$$

$$d'_{js} = \begin{cases} V \text{ if } \pi_j \text{ has the component } k_s, \\ 0 \text{ in the other case,} \end{cases} \quad j = \overline{1,J},\ s = \overline{1,S}$$

Operational requirements related to the access to computer components can be formulated as below:

$$\sum_{v=1}^{V} c'_{vs} x_{vi}^m \le \sum d'_{js} x_{ij}^{\pi},\ \ i = \overline{1,2},\ s = \overline{1,S} \tag{7}$$

3. Multiobjective optimisation problem

A cost of computers can be calculated according to the following formula:

$$F_2(x^{\pi}) = \sum_{i=1}^{I} \sum_{j=1}^{J} \kappa_j x_{ij}^{\pi} \tag{8}$$

where κ_j represents the cost of computer π_j.

Another criterion used for an allocation assessment is the cost of a parallel program execution, which can be calculated, as below:

$$F_1(x) = \sum_{j=1}^{J}\sum_{v=1}^{V}\sum_{i=1}^{I} t_{vj} x_{vi}^{m} x_{ij}^{\pi} + \sum_{v=1}^{V}\sum_{u=1}^{V}\sum_{i=1}^{I} \tau_{vuik} x_{vi}^{m} x_{uk}^{m} \quad (9)$$

where

$x \in \mathcal{B}^M$,

t_{vj} – the cost of executing the task m_v by the computer π_j,

τ_{vj} – the cost of communications between the task m_v assigned to the ith node and the task m_u assigned to the kth node,

$\mathcal{B}$ – the set {0, 1}.

Let the case of multiobjective optimisation problem for finding the Pareto-optimal allocations of tasks in a distributed computer system be considered. This problem is formulated as ($\mathcal{X}$, F, P), where:

1) $\mathcal{X}$ - a feasible solutions set

$$\mathcal{X} = \{x \in \mathcal{B}^{I(V+J)} \mid \sum_{i=1}^{I} x_{vi}^{m} = 1 \text{ for } v = \overline{1,V}; \quad (10)$$

$$\sum_{j=1}^{J} x_{ij}^{\pi} = 1 \text{ for } i = \overline{1,I};$$

$$\sum_{v=1}^{V} c_{vr} x_{vi}^{m} \le \sum_{j=1}^{J} d_{jr} x_{ij}^{\pi}, \; i = \overline{1,I}, \; r = \overline{1,R};$$

$$\sum_{v=1}^{V} c'_{vs} x_{vi}^{m} \le \sum d'_{js} x_{ij}^{\pi}, \; i = \overline{1,I}, \; s = \overline{1,S} \}$$

2) F - a vector quality criterion

$$F : \mathcal{X} \to \mathcal{R}^2 \quad (11)$$

$F(x) = [F_1(x), F_2(x)]^{\mathrm{T}}$ for $x \in \mathcal{X}$

where $F_1(x)$ is calculated by (9), $F_2(x)$ is calculated by (8).

3) P - the Pareto relation [2, 9]

4. Neural model for objective function

There are some combinatorial optimisation problems that are convenient for solving by an Hopfield ANN. For instance, the Travelling Salesman Problem [8], some graphs problems [4], and linear minimisation problems [7] can be solved by the optimisation techniques based on the Hopfield ANN. That is why, we consider a neural network approach for optimisation of task allocation to multiprocessor systems. The modified Hopfield models for solving optimisation problems with linear objective function or quasi-quadratic objective function can be design [3]. By introducing nonnegative convex combination method for solving considered combinatorial problem with several linear objective functions, the Hopfield model can be constructed. A main advantage of the above approach consists in the parallel effect in numerical computations.

4.1. The standard Hopfield model

In the gradient model of standard HANN, 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 following differentiable equations [8]:

$$\frac{\mathrm{d}u_m}{\mathrm{d}t} = -\frac{u_m}{\eta_m} + \sum_{n=1}^{M} w_{nm} g_n(u_n) + I_m, \; m = \overline{1,M}, \quad (12)$$

where

M – the number of neurons,

u_m – the global activation level of mth neuron,

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.

A matrix of synaptic weights is symmetric. Moreover, $w_{mm} = 0$ for $m = \overline{1,M}$. External inputs are constant during a network operation. Signals in a neuron are transformed according to the logistic activation function f_m as follows:

$$x_m = f_m(u_m) = \frac{1}{1 + e^{-\alpha_m u_m}}, \; m = \overline{1,M}, \quad (13)$$

where α_m is a gain coefficient in mth neurone ($\alpha_m > 0, m = \overline{1,M}$).

Theorem 1. [3]

If the non-linear activation function is substituted by the logistic activation function, then the neural activation state equations (13) can be transformed into the following equation system, for $m = \overline{1,M}$:

$$\frac{\mathrm{d}x_m}{\mathrm{d}t} = x_m(t)(1 - x_m(t)) \left(\frac{1}{\eta_m} \ln \frac{1 - x_m(t)}{x_m(t)} + \alpha_m \left(\sum_{n=1}^{M} w_{nm} x_n(t) + I_m \right) \right) \quad (14)$$

Figure 2 shows the logistic activation functions for different values of gain coefficients α. If the gain coefficient increases, then activation function tends to the binary function with the threshold in the point 0.

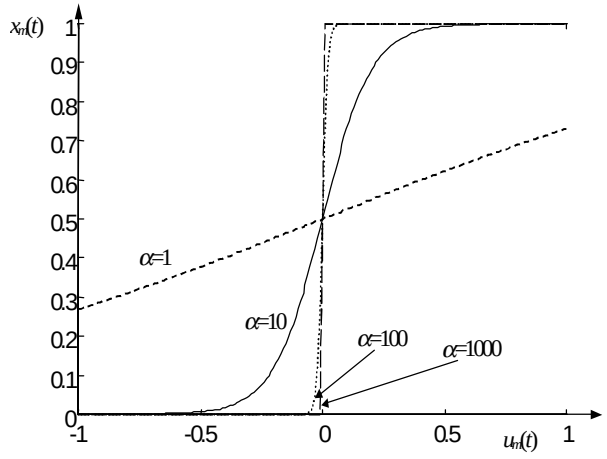

Figure 4: Logistic activation functions for different values of gain coefficients

Hopfield found the Liapunov function for the differential system (13), which is given by the formula:

$$E(x)=-\frac{1}{2}\sum_{m=1}^{M}\sum_{n=1}^{M}w_{mn}x_m x_n-\sum_{m=1}^{M}I_m x_m+\sum_{m=1}^{M}\frac{1}{\eta_m}\int_0^{x_m}g_m^{-1}(\xi_m)d\xi_m \quad (15)$$

Constraints (1) and (4) can be formulated as a general combinatorial constraint. The above constraint can be represented in a general form as follows:

$$\sum_{m=1}^{M} x_m = L \quad (16)$$

where x_m is a binary variable ($M \geq L$).

4.2. Network for linear constrained minimisation

An optimisation problem with one criterion is studied for finding the minimal cost of a distributed computer system:

$$\min_{x\in\mathcal{X}} \sum_{i=1}^{I}\sum_{j=1}^{J}\kappa_j x_{ij}^{\pi}. \quad (17)$$

Let a temporary assumption be made that resources are unlimited. For the above problem, the I separated modified networks UHANN/1/J [1] can be used to satisfy the constraint $\sum_{j=1}^{J} x_{ij}^{\pi} = 1,\ i=\overline{1,I}$. Values of the external inputs are calculated by $I(x_{ij}^{\pi}) = 2J + 1.5 - \Delta I(x_{ij}^{\pi}),\ j=\overline{1,J},\ i=\overline{1,I}$, where $\Delta I(x_{ij}^{\pi}) = \frac{\kappa_j}{\kappa_{\max}}$ and κ_{max} is the cost of the most expensive processor.

Figure 3 shows the minimisation of the energy function for J=5, α=100, η=1, Δt = 0.2, and κ=[5, 4, 3, 2, 1]^T. It is I=[0.5, 0.7, 0.9, 1.1, 1.3]^T. Two separate networks UHANN/1/J are considered for each node number. Only 5 steps are required to find the optimal solution with given accuracy 0.001.

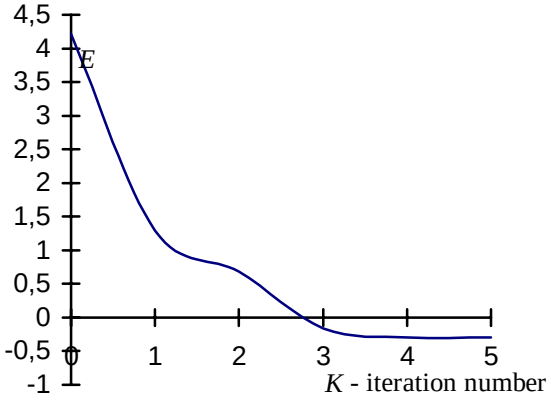

Figure 3: Minimization of the energy function E in HANN/F_1/R with 5 neurons

4.3. Network HANN/F_2/R for cost minimisation

If the cost criterion F_2 is minimised subject to constraints

$$\sum_{i=1}^{I} x_{vi}^{m} = 1,\ v=\overline{1,V};$$

$$\sum_{j=1}^{J} x_{ij}^{\pi} = 1,\ i=\overline{1,I};$$

$$\sum_{v=1}^{V} c_{vr} x_{vi}^{m} \leq \sum_{j=1}^{J} d_{jr} x_{ij}^{\pi},\ i=\overline{1,I},\ r=\overline{1,R};$$

then the energy functions of neural networks designed for constraint satisfaction and for objective function minimisation can be aggregated in a global function as below:

$$E(x,\beta) = F_2(x) + \sum_{v=1}^{V}\beta_v E_v(x) + \sum_{i=V+1}^{V+I}\beta_i E_i(x) + \sum_{i=1}^{I}\sum_{r=1}^{R}\beta_{ir} P_{ir}(x), \quad (18)$$

where

β_v, β_i, β_{ir} – positive penalty coefficients,

E_v – an energy function of the network UHANN/1/I for satisfying the constraint $\sum_{i=1}^{I} x_{vi} = 1$

E_i – an energy function of the network UHANN/1/J for satisfying the constraint $\sum_{j=1}^{J} x_{ij}^{\pi} = 1$.

Figure 4 shows synaptic connections from the neuron x_{v1}^{m} in the network HANN/F_2/R. There are two nodes, only. The neuron x_{v1}^{m} has an external input β_v. The synaptic weight between neurons (x_{v1}^{m}, x_{v2}^{m}) is equal to -2β_v. If more neurones are considered, then each pair of them has the same synaptic weight -2β_v. The above results are obtained from the network UHANN/1/I to satisfy the constraint $\sum_{i=1}^{I} x_{vi} = 1$.

The neuron x_{v1}^{m} is connected to the neuron x_{u2}^{m} by a synaptic connection with the weight -τ_{vu}. If tasks are allocated to different computers, then the synaptic weight between neurones (x_{vi}^{m}, x_{uk}^{m}) is equal to -τ_{vu} that is a negative value of cost communication between them. The neuron x_{v1}^{m} is connected to the neuron x_{1j}^{π} by a synaptic connection with the weight -t_{vj}. If task m_v is assigned to the computer π_j in the node w_i, then the synaptic weight between neurones $(x_{vi}^{m}, x_{ij}^{\pi})$ is equal to -t_{vj} negative value of execution cost of this task on

an allocated computer. Above results are obtained by the comparison between an objective function and the energy function.

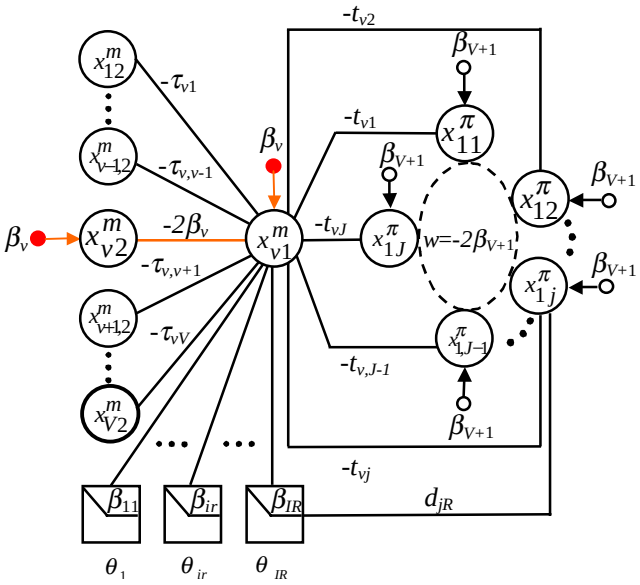

Figure 4: Synaptic connections from the neuron x_{v1}^m in the network HANN/F_2/R

The neuron x_{v1}^m is connected to the static neurone θ_{ir} by a synaptic connection with the weight $-c_{vr}$. If task m_v is assigned to the node w_i and the limit of the rth resource is exceeded, then signal value from the neuron θ_{ir} is equal to $\beta_{ir}\left(\sum_{j=1}^{J} d_{jr} x_{ij}^{\pi} - \sum_{v=1}^{V} c_{vr} x_{vi}^{m}\right)$ and it is multiplied by synaptic weight $-c_{vr}$. Simulation results presented the relaxation of network HANN/F_2/R are submitted in [2].

5. Finding a Pareto solution

The non-negative convex combination method can be applied for finding the Pareto-suboptimal solution. The multiobjective optimisation problem [2] is transformed to the minimisation problem with one criterion, as follows:

$$\min_{x \in X} \{\sum_{n=1}^{N} \lambda_n F_n(x)\}, \tag{19}$$

where

$$\sum_{n=1}^{N} \lambda_n = 1,\ \lambda_n \geq 0,\ n = \overline{1, N}.$$

The energy function of the neural network PHANN is constructed for finding a Pareto solution as below:

$$E(x,\lambda,\beta) = \lambda_1 F_1(x) + \lambda_2 F_2(x) + \sum_{v=1}^{V} \beta_v E_v(x) + \sum_{i=V+1}^{V+I} \beta_i E_i(x) + \sum_{i=1}^{I} \sum_{r=1}^{R} \beta_{ir} P_{ir}(x) \tag{20}$$

If λ_1=1 and λ_2=0, then the network PHANN becomes the network HANN/F_1/R and the Pareto-suboptimal solution can be obtained. This sort of the Pareto-optimal solution is called a hierarchical solution. Figure 5 shows examples of PHANN simulations for λ_1=1 and λ_2=0. In the equilibrium point, an energy E of the PHANN has constant value and it cannot be minimised. Values of an objective function F_1 increase and decrease during state modifications. However, a trajectory of an objective function F_1 converges to an optimal value F_1* in the equilibrium point. Values of the energy function E, the objective function as well as the energy function for constraints are expressed as the real numbers.

An energy function E_1 of the network UHANN/1/I is responsible for satisfying the constraint $\sum_{i=1}^{I} x_{1i} = 1$. It converges to zero in the equilibrium point, because above constraint is satisfied. The other constraints are satisfied, too. Penalty coefficients are found by systematically increasing with $\Delta\beta$=0.05. Coefficients from non-satisfied constraints are taken in the equilibrium point. Initial values of penalty coefficients are equal to 1.

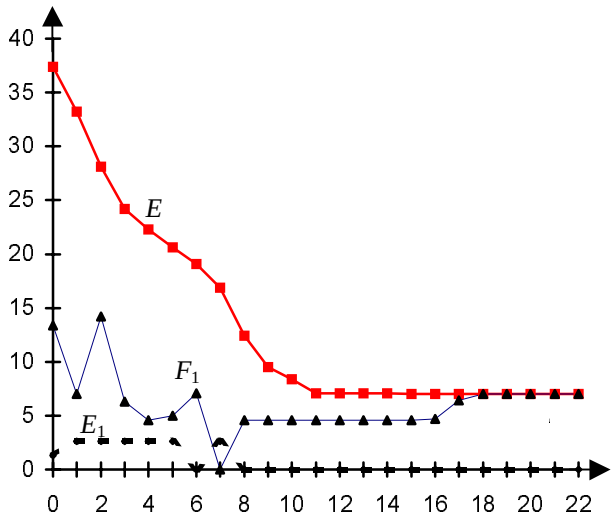

Figure 5: Minimisation of energy function in the network PHANN with λ_1=1

6. Numerical examples

The following parameters of optimisation problem are assumed: a task number of 4 (V=4), computer sorts of 2 (J=2), two computers (I=2), and three resources (R=3), Matrix of task execution times **T** is $\begin{bmatrix} 2.5 & 1.3 \\ 1.4 & 8.9 \\ 1.2 & 5.2 \\ 2.2 & 1.1 \end{bmatrix}$, matrix of

communication times τ is $\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, matrix of required resources **C** is $\begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & 1 \\ 3 & 5 & 4 \\ 4 & 2 & 0 \end{bmatrix}$, matrix of computer resources **D** is $\begin{bmatrix} 5 & 5 & 5 \\ 5 & 3 & 1 \end{bmatrix}$, vector of computer costs κ is [1, 2]. There are 12 decision variables for this test problem. The gain coefficient α is equal to 200, the passive coefficient η is 1, and an initial state vector $u(t_0)$ is $[10^{-6}, 0, 10^{-6}, 0, 10^{6}, 0, 10^{-6}, 0, 10^{-6}, 0, 10^{-6}, 0]^T$. New states of a network can be calculated by the differentiable equations (13) with the step length Δt=0.2.

If λ_1=1 and λ_2=0, then the network PHANN minimises its energy function as it is shown in Figure 9. A Pareto-optimal solution $x^*=[0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1]^T$ is reached. It is a hierarchical solution for preferences function F_1 before F_2. $F_1(x^*)$=7.0 and $F_2(x^*)$=3. For solution x^*, tasks m_2 and m_3 are assigned to the node w_1. Tasks m_1 and m_4 are assigned to the node w_2. The computer π_1 is located in the node w_1, and the computer π_2 is located in the node w_2. Penalty coefficients β_v, β_i are found as follows $[2.55, 2.45, 2.25, 2.25, 7.35, 7.30]^T$. Coefficients β_{ir} are elements of the matrix $\begin{bmatrix} 2.5 & 1.2 & 1.5 \\ 1.5 & 1.3 & 1.1 \end{bmatrix}$.

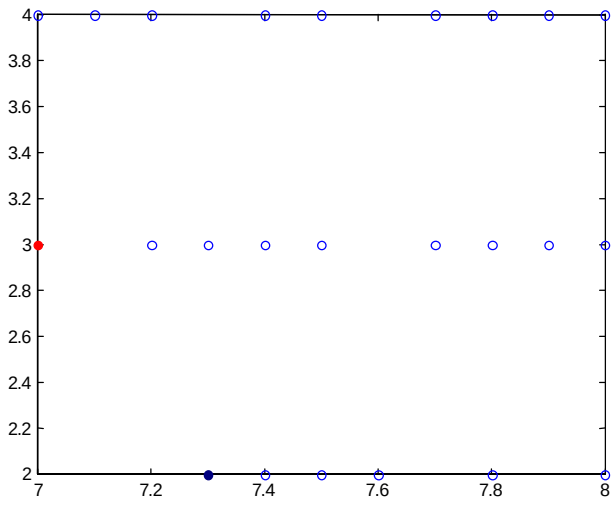

Figure 6: Results obtained by the PHANN for different values of parameter λ_1

Figure 6 presents results obtained by the PHANN model with $\lambda_1 \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Pareto-optimal evaluations are $y^*(1)$ and $y(0.2)$. The circles represent points close to the Pareto set in the objective set. We assume $y=F(x)$. The point $y(0.2)$ represents the evaluation obtained by the PHANN with λ_1=0.2 . For different values of parameter λ_1 the other evaluations are reached.

For λ_1=1, the point $y^*(1)$ represent the evaluation of a solution x^* described in this section. For λ_1=0.2, the point $y^*(0.2)$ has coordinates (7.3; 2). A Pareto-optimal task assignment is $x^{**}=[1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0]^T$. It is a hierarchical solution if function F_2 is preferred against F_1. $F_1(x^{**})$=7.3 and $F_2(x^{**})$=2. In the solution x^{**}, tasks m_1 and m_2 are assigned to the node w_1. Tasks m_3 and m_4 are assigned to the node w_2. The computers that are π_1-class are located both in the node w_1 and in the node w_2.

7. Concluding remarks

In this paper, the Hopfield model of artificial neural networks for finding some task allocations in multiple computer systems has been proposed. The cost of parallel program execution and also the cost of computers have been minimised. Two separated models of neural networks for minimisation of the computer cost and the cost of parallel program completing have been given. Moreover, the PHANN model for finding a Pareto-optimal solution has been considered.

To satisfy constraints, two rules are applied. According to the first rule, synaptic weights and external inputs are determined by comparison between an energy function and the penalty function. In this way, a task assignment constraint and a computer assignment constraint are considered. According to the second rule, resource constraints are satisfied by additional neurons, which generate positive response if constraints are not satisfied.

References

[1] Abe, S., "Convergence Acceleration of the Hopfield Neural Networks by Optimising Integration Step Sizes", IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics. vol. 28, No. 1, pp. 194 - 201, 1996.

[2] Balicki, J., "Evolutionary Neural Networks for Solving Multiobjective Optimization Problems", In S. Szczepaniak (Ed.): "Computational Intelligence and Applications", A Springer Verlag Company, Heidelberg-New York, pp. 108 - 118, 1999.

[3] Chong, E.K.P., Zak, S.H. "An Introduction to Optimisation", John Wiley&Sons, Inc., New York, 1996.

[4] Haykin, S. "Neural Networks – a Comprehensive Foundation", Prentice Hall Int., New Jersey, 1999.

[5] Kafil, M., Ahmad, I., "Optimal Task Assignment in Heterogeneous Distributed Computing Systems", IEEE Concurrency, vol. 6, No. 3, pp. 42 – 51, 1998.

[6] Lillo, W.E., Hui, S., Zak, S.H., "Neural Networks for Constrained Optimisation", International Journal of Circuit Theory and Applications, vol. 21, pp. 385-399, 1991.

[7] Sun, K.T., Fu, H.C., "A Hybrid Neural Model for Solving Optimisation Problems", IEEE Transactions on Computers, vol. 42, No. 2, pp. 219 - 227, 1993.

[8] Tank, D.W., Hopfield, J.J., "Simple 'Neural' Optimisation Networks: an A/D Converter, Signal Decision Circuit, and Linear Programming Circuit", IEEE Transactions on Circuits and Systems, vol. CAS-33, pp. 533 - 541, 1986.

[9] Weglarz, J. (ed.), "Recent Advances in Project Scheduling", Kluwer Academic Publishers, Dordrecht, 1998.