Appl. Math. and Comp. Sci., 1997, Vol.7, No.3, 567–592

GENETIC ALGORITHMS AND HOPFIELD NEURAL NETWORKS FOR SOLVING COMBINATORIAL PROBLEMS†

JERZY BALICKI*, ANDRZEJ STATECZNY*
BOGDAN ŻAK*

In this paper, a Hopfield neural-network population for solving NP-hard multiobjective optimization problems with zero-one decision variables is proposed. The initial states of the Hopfield models in this population are modified by a genetic algorithm and the energy functions are constructed by using a non-negative convex combination method. Accordingly, optimization neural networks for the related optimization problem are designed. Simulation results are also presented to illustrate the effectiveness of the approach.

1. Introduction

Genetic algorithms can be used to solve many optimization problems. Holland (1975) developed this approach and its theoretical foundations. In recent years a lively interest has been observed in the application of genetic algorithms to combinatorial optimization problems (Bac and Perov, 1993; Reeves, 1995). Simultaneously, Tank and Hopfield (1986) considered the neural approach to solving the Travelling Salesman Problem (TSP). Moreover, Sun and Fu (1993) proposed a hybrid neural-network model consisting of multiprocessor systems for finding solutions of TSP and discovering solutions to the Hamiltonian Cycle. A review of applications of neural networks can be found in (Korbicz *et al.*, 1994). Some optimization networks for the related optimization problems are considered in (Abe, 1996; Kaznachey and Jagota, 1997).

In this paper, two optimization problems of resource allocation are formulated as multiobjective optimization problems (Ameljańczyk, 1986) with zero-one decision variables. In the first problem, Pareto-optimal solutions are generated. In the other problem, compromise solutions with parameter $p = 1$ are obtained. For solving these problems, Hopfield artificial neural networks (HANN) are designed. Energy functions are constructed for the related multiobjective optimization problems. Formulae to determine the values of the external inputs and synaptic weights are presented. The HANN for finding Pareto-suboptimal solutions (PHANN) is thus obtained. Similarly, the neural network called C1HANN which gives subcompromise solutions at the

† A preliminary version of this paper was presented at the 2nd National Conference on *Neural Networks and Their Applications*, 30 April–4 May, 1996, Szczyrk, Poland.

* Polish Naval Academy, ul. J. Śmidowicza 19, 81–919 Gdynia, Poland, e-mail: {jbali, astat, bzak}@amw.gdynia.pl.

equilibrium points for $p = 1$ is proposed. However, the neural approach to solve optimization problems has a serious disadvantage, consisting in the fact that at the equilibrium points only local minima of the energy function can often be attained. The resulted solution can even be non-feasible. To avoid this drawback, a genetic algorithm using the designed HANN is proposed.

2. A Standard Genetic-Neural Algorithm SGNA

Genetic algorithms can be used in solving some NP-hard optimization problems. Although there are many versions of the basic genetic algorithm (Bac and Perov, 1993; Goldberg and Lingle, 1985), the underlying mechanism operates on a population of individuals. The most important is a suitable representation of chromosomes which in optimization problems represent the solutions. If the solutions are represented by binary numbers, the standard genetic algorithm can be used. For real numbers in the solution representation, some coded binary numbers are taken in most cases. If real numbers are directly processed by the algorithm, then this algorithm performs evolutionary computations (Holland, 1975).

Moreover, the crossover operator with a correct value of the crossover probability p_c has to be chosen. Apart from the standard crossover operator, Goldberg and Lingle (1985) suggested a crossover operator called the "partially-mapped crossover", and Bac and Perov (1993) proposed a crossover operator created by the non-Abel group theory.

During selection of parents, the fitness function plays the main role. For maximization problems without constraints, the objective function $f(x)$ is the fitness function $F(x)$. If the objective function should be minimized, then the fitness function can be chosen as $F(x) = -f(x)$. If there are constraints, then the optimization problem should be converted into an unconstrained problem by one of the well-known methods for one-criterion problems. For multicriteria optimization problems one of the scalarization method (Ameljańczyk, 1986) can be used to transform it into an optimization problem with one criterion. In particular, the non-negative convex combination method (Balicki and Kitowski, 1996) or the hierarchical method (Ameljańczyk, 1986) can be applied.

Reproduction of offspring depends on the values of the fitness-function for chromosomes (solutions). If a value of the fitness function is larger, then the chance of selection for a given solution is greater. Some copies of the solution are cloned according to the fitness function and random selection of parents is carried out. Then the crossover of parent chromosomes with a high probability p_c is made. From two solutions two new solutions are created by exchanging parts of chromosomes at a randomly chosen point.

After the reproduction and the parent crossover, a mutation with a low probability p_m for each gene (one bit in the binary representation of the solution) is applied. According to the decrease in the mutation probability p_m, the influence of mutation on the chromosomes decreases, too. But its too small value causes that optimal or suboptimal solutions cannot be found. Hence a suitable value of the mutation probability is fixed by the simulated trial-and-error method. Reproduction, crossover, and

mutation are repeated until a stopping condition is satisfied. The stopping condition is related to the maximal number or to the "patient" condition, where after a given number of iterations the algorithm is stopped if there is no improvement in the fitness function sense.

Genetic algorithms and evolutionary computations have the same disadvantage. Namely, they fail to find optimal solutions in many cases. Neural networks have difficulties when they start from bad initial points. But from another point of view, genetic algorithms and neural networks have been successfully applied for several optimization problems. This is the reason why a combination of genetic algorithms and neural networks can give additional possibilities to reach "good" solutions. It is necessary to take advantage of them to a great extent.

Standard genetic algorithms do not require many iterations during one reproduction, crossover, and mutation for the whole population of solutions. However, Hopfield analog networks demand $O(knM^2)$ iterations to obtain an equilibrium point, where k denotes the number of increases in the penalty parameter, n is the number of network-state updates, and M stands for the number of decision variables. For a parallel implementation of the Hopfield model, the complexity is $O(kn)$.

It is possible to combine genetic algorithms and neural networks in several ways. First of all, neural networks prepared to optimize of a problem can be used as solvers for local optimization in each basic step of the standard genetic algorithm. In such an approach, the genetic algorithm usually operates on the initial activation level of Hopfield networks, since for given synaptic weights and external inputs the initial activation level has a strong influence on the solution quality. This algorithm can be called the standard genetic-neural algorithm SGNA. In Fig. 1 the diagram of an SGNA is presented. In the SGNA the crossover operator picks up K randomly chosen pairs of HANN from the network population to obtain K pairs of offspring by the Goldberg operator (Goldberg and Lingle, 1985). The termination condition $i < I$ is satisfied when during I trials the maximum of the fitness function from all chromosomes in the first population cannot be increased.

If the next population has a larger maximal fitness value, then it is labelled with the number $i = 1$. Experimental results confirm the usefulness of genetic algorithms in solving combinatorial optimization problems, and especially multiobjective optimization problems. The main disadvantage of SGNA is the fact that we have to know the parameters of the HANN prepared to solve optimization problems.

3. Multicriteria Optimization Problem with Zero-One Decision Variables

Let us consider the following example of the multiobjective optimization problem (X, F, P) consisting in designing Pareto-optimal allocations of program modules and

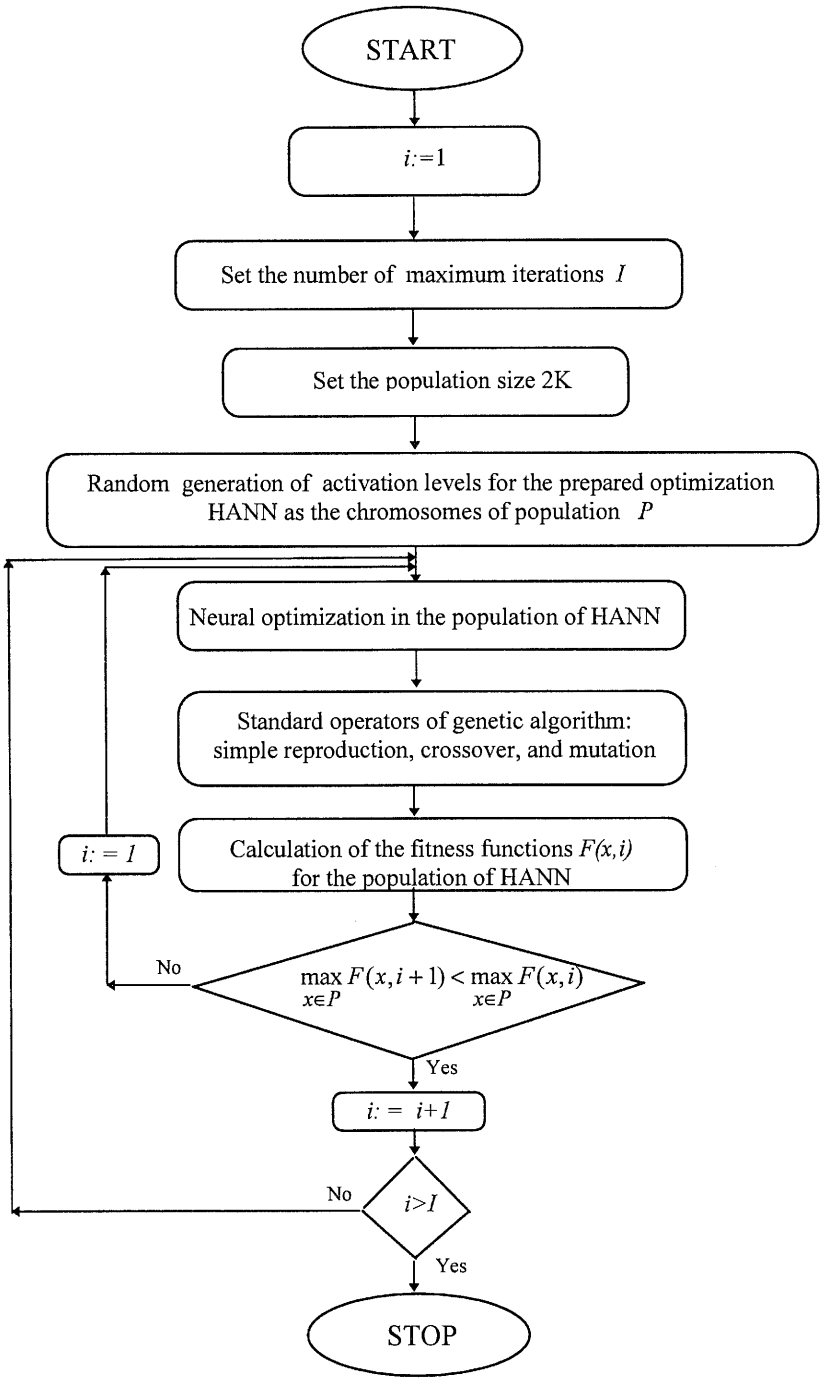

Fig. 1. A diagram of a standard genetic-neural algorithm SGNA.

processor types:

- X —a feasible set,

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

- F —a vector performance index,

$$F : X \to \mathbb{R}^2, \qquad F(x) = \big[F_1(x),\ F_2(x)\big]^T, \qquad x \in X$$

$$F_1(x) = \sum_{i=1}^{2} \sum_{j=1}^{J} \kappa_j x^{\pi}_{ij}$$

$$F_2(x) = \sum_{j=1}^{J} \sum_{v=1}^{V} \sum_{i=1}^{2} t_{vi} x_{vi} x^{\pi}_{ij} + \sum_{v=1}^{V} \sum_{u=1}^{V} \sum_{i=1}^{2} t_{vu} x_{vi} (1 - x_{ui}) \tag{1}$$

- P —the Pareto relationship,

where $B = \{0,1\}$, V is the number of program modules $m_1, \ldots, m_v, \ldots, m_V$, J is the number of processor types $\pi_1, \ldots, \pi_j, \ldots, \pi_J$, I stands for the number of processing nodes w_1, w_2, $I = 2$,

$$x_{vi} = \begin{cases} 1 & \text{if program module } m_v \text{ is} \\ & \text{assigned to processing node } w_i, \quad v = \overline{1,V},\ i = \overline{1,I}, \\ 0 & \text{otherwise} \end{cases}$$

$$x^{\pi}_{ij} = \begin{cases} 1 & \text{if processor type } \pi_j \text{ is} \\ & \text{assigned to processing node } w_i, \quad i = \overline{1,I},\ j = \overline{1,J}, \\ 0 & \text{otherwise} \end{cases}$$

F_1 is the cost of allocated processors, κ_j denotes the cost of processor type π_j, F_2 is the performance time for allocated program modules, t_{vj} stands for the processing time of module m_v on processor π_j, τ_{vu} denotes the communication time between program modules m_v and m_u $(v < u)$ which are processed on different processors, $v, u = \overline{1,V}$.

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

The general form $\sum_{m=1}^{M} x_m = L$ includes several constraints from the collection of combinatorial problems. For instance, in the Travelling-Salesman Problem (Aiyer *et al.*, 1990) during L steps (L is the number of all cities) a salesman should pass through each city c_i exactly once, which can be written down as $\sum_{k=1}^{L} x_{ik} = 1$ for $k = \overline{1, L}$, where x_{ik} is equal to 1 if in the k-th step the salesman is in the city c_i. The trip of the salesman is characterized by a bounded-time condition. He should make exactly L steps, and if the constraints $\sum_{k=1}^{L} x_{ik} = 1$ are satisfied for $k = \overline{1, L}$, then that requirement is met, too.

We deal with a similar situation in the vehicle routing problem VRP (Reeves, 1995), where V vehicles make deliveries to C customers. In VRP we have to allocate customers to vehicles and find the order in which each vehicle visits its customers so as to minimize the global distance covered by the vehicles. Because each customer has to be assigned to exactly one vehicle (the vehicle has enough capacity to deliver units of resource to one customer), there are the constraints $\sum_{v=1}^{V} x_{vc} = 1$ for $c = \overline{1, C}$, where x_{vc} is equal to 1 if the v-th vehicle delivers units to the c-th customer.

Similar constraints are in the 0–1 knapsack problem, the set covering problem, or the standard assignment problem (Reeves, 1995). They can be expressed in the general form $\sum_{m=1}^{M} x_{vi} = L$. To satisfy them and other sorts of constraints, Hopfield neural networks HANN can be used (Tank and Hopfield, 1986). Moreover, for minimization of a linear function or a quasi-quadratic function with binary variables Hopfield networks can also be useful.

4. Hopfield Neural Networks

John Hopfield proposed a special case of artificial recurrent neural networks for several differentiable applications, e.g. optimization, association memory, or A/C converters (Tank and Hopfield, 1986). A review can be found in (Tadeusiewicz, 1993). Hopfield networks are still criticized (Aiyer *et al.*, 1990; Lillo *et al.*, 1991) or developed successfully (Balicki and Kitowski, 1996; Ciepliński and Jędrzejek, 1994). From the former point of view, they are not flexible enough for several optimization problems, e.g. genetic algorithms, evolutionary methods, tabu search, or simulated annealing. But, from the latter point of view, they can be applied for solving some cases of optimization problems. In particular, problems with a linear or a quasi-quadratic objective function and constraints with binary decision variables are preferred.

Decision problems which have a known *polynomial algorithm* for finding the decision 'yes' or 'no' are said to be of class P. If some decision problems have a known *non-deterministic polynomial algorithm* for finding the decision 'yes' or 'no', then they are said to be of class NP. The decision version of a problem and the optimization version of the same problem are closely related, since it is intuitively obvious that an algorithm for the decision version of a problem can be employed to solve its optimization version. But there are decision versions which are of class NP and optimization versions which are not in the class NP. Let us consider a *tree-search algorithm* where all branches can be searched simultaneously. If the maximum time taken by a branch is polynomially bounded, then the problem is of class NP. If, in addition, a problem P

is such that there is another problem in the class NP which is polynomially transformable to P, then P is NP-hard (Reeves, 1995). If an NP-hard problem P belongs to the class NP, then this problem is NP-complete. Some combinatorial problems which were solved by Hopfield network are NP-hard. The class of NP-hard problems is now known to be quite large. The list of NP-hard problems is contained in (Reeves, 1995), and is periodically updated in the Journal of Algorithms.

The main advantage of the HANN is parallel processing in neurons. The neurons process simultaneously signals obtained from other neurons. This cooperation is ordered to minimize computational energy of the whole network. When the network attains a local minimum of the energy function, then it stops. A Hopfield brain behaves similarly to a lazy man who wants to be at rest. It prefers to do nothing if the network energy could increase.

In analog models of HANN the neural activation states are changed from the initial state $u(t_0) = [u_1(t_0), \ldots, u_m(t_0), \ldots, u_M(t_0)]^T$ according to the ordinary differential equations

$$\frac{\mathrm{d}u_m}{\mathrm{d}t} = -\frac{u_m}{\eta_m} + \sum_{n=1}^{M} w_{nm} g_n(u_n) + I_m, \qquad m = \overline{1, M} \tag{2}$$

where M is the number of neurons, u_m denotes the global activation level of the m-th neuron, $m = \overline{1, M}$, η_m stands for the positive passive coefficient for the neuron with the output x_m, w_{nm} is the synaptic weight from x_n to x_m, and I_m denotes the external input to the neuron x_m.

The matrix of synaptic weights is symmetric. Moreover, $w_{mm} = 0$ for $m = \overline{1, M}$. At the equilibrium points of the HANN ($\mathrm{d}u_n/\mathrm{d}t = 0$, $m = \overline{1, M}$ or $\lim_{t\to\infty} \mathrm{d}u_m/\mathrm{d}t = 0, m = \overline{1, M}$) we have $u_m/\eta_m = \sum_{n=1}^{M} w_{nm} g_n(u_n) + I_m$, $m = \overline{1, M}$.

For hardware implementation of HANN, the following well-known Hopfield motion equation is more useful:

$$C_m \frac{\mathrm{d}u_m}{\mathrm{d}t} = -\frac{u_m}{\eta_m} + \sum_{n=1}^{M} \widehat{w}_{nm} g_n(u_n) + \widehat{I}_m, \qquad m = \overline{1, M} \tag{3}$$

where R_m and C_m are positive coefficients such that

$$\eta_m = R_m C_m \quad \text{for} \quad m = \overline{1, M}, \quad w_{nm} = C_m \widehat{w}_{nm}, \quad I_m = C_m \widehat{I}_m$$

Coefficients R_m and C_m have a physical meaning in the electrical model of the HANN (Tank and Hopfield, 1986): C_m is the capacity of an input capacitor and R_m denotes the parallel combination of ρ_i (the input resistor of the m-th neuron) and R_{mn} (the resistor in the synaptic connection n–m), i.e.

$$\frac{1}{R_m} = \frac{1}{\rho_m} + \sum_{n=1}^{M} \frac{1}{R_{mn}}$$

This formula can be transformed as follows:

$$\tau_m \frac{\mathrm{d}u_m}{\mathrm{d}t} = -u_m + \sum_{n=1}^{M} \widetilde{w}_{nm} g_n(u_n) + \widetilde{I}_m, \qquad m = \overline{1, M} \tag{4}$$

where $\tau_m = \eta_m = R_m C_m$ for $m = \overline{1,M}$, $w_{nm} = R_m \widetilde{w}_{nm} = \tau_m \widehat{w}_{nm}$, $I_m = R_m \widetilde{I}_m = \tau_m \widehat{I}_m$. Here $\tau_m = \eta_m$ denotes either a passive coefficient or the adaptive time constant.

The activation function in a neuron can be modelled in the form

$$g_m(u_m) = \frac{1}{2}\left[1 + \tanh(\alpha u_n)\right], \qquad n = \overline{1,M} \tag{5}$$

where α_m is the gain coefficient in the m-th neuron ($\alpha_m \geq \alpha_{\mathrm{gr}} > 0$ for $m = \overline{1,M}$).

Hopfield found a Lyapunov function for the differential system (3):

$$E(u) = -\frac{1}{2}\sum_{n=1}^{M}\sum_{m=1}^{M} w_{nm} g_n(u_n) g_m(u_m) - \sum_{m=1}^{M} I_m g_m(u_m) + \sum_{m=1}^{M} \int_0^{g_m(u_m)} g_m^{-1}(\xi_m)\, \mathrm{d}\xi_m \tag{6}$$

where g_m^{-1} is an inverse of the activation function g_m, $u_m = g_m^{-1}(x_m)$.

The basic property of the analog HANN is the fact that if gain coefficients α_m, $m = \overline{1,M}$ are large enough, then the equilibrium points of (3) can usually find a local minimum of the energy function E. Moreover, $x_m \in \{0 + \varepsilon,\ 1 - \varepsilon\}$ for $m = \overline{1,M}$, where ε decreases to zero according to increasing the gain coefficient. Hopfield networks navigate towards equilibrium points while decreasing the energy function. Because at an equilibrium point it is not possible to decrease the energy function, the HANN finds either a local minimum of its energy function or its saddle point, where the energy gradient is equal to zero. This results from the proof that the Hopfield energy function is a Lyapunov function.

The above-mentioned standard model of the HANN can be developed for solving optimization problems. A general method transforms the optimization problem into an energy function. This means that optimization problems with constraints are reduced to optimization problems without constraints by the penalty-function method or the Lagrange-multiplier method, and then these functions are compared with the Hopfield energy function to determine the synaptic weights and external inputs in the HANN. Since the solutions obtained in these HANNs are related to the initial states of the activation levels, genetic processing is used to get the best solutions.

5. Uniform Hopfield Networks for Satisfying of Linear Constraints

Uniform Hopfield networks UHANN play an important role in satisfaction of the special class of constraints expressed in a general form as $\sum_{m=1}^{M} x_m = L$. For the uniform Hopfield networks all main parameters have the same value for each neuron, i.e. $w_{nm} = w$ for $n, m \in \overline{1,M}$, $I_m = I$ for $m = \overline{1.M}$, $\eta_m = \eta$ for $m = \overline{1,M}$, $\alpha_m = \alpha$ for $m = \overline{1,M}$.

To solve numerically the motion equations of the UHANN, the Euler method can be used. Then the following iterative procedure to find the active levels at the next moment is applied:

$$u_m(t_k + \Delta t) = \left(1 - \frac{\Delta t}{\eta_m}\right) u_m(t_k) + w\Delta t \sum_{\substack{n=1 \\ n \neq m}}^{M} g_n\big(u_n(t_n)\big) + I\Delta t \quad \text{for } m = \overline{1, M} \quad (7)$$

where Δt is the integration step length and t_k stands for the time of the k-th iteration, $k = 0, 1, 2, \ldots$.

In the Euler method the integration step Δt should be taken as small as possible to avoid errors related to the approximation of differential equations by several partitions. However, if Δt is too small, the number of iterations is excessively large.

For a uniform Hopfield network the main part of the energy function called the basic energy function, where the influence of synaptic weights and external inputs is important, can be expressed as follows:

$$E^*(u) = -\frac{w}{2} \sum_{n=1}^{M} \sum_{m=1}^{M} g_n(u_n) g_m(u_m) + I \sum_{m=1}^{M} g_m(u_m) \quad (8)$$

If in the UHANN there is only one neuron with the external input equal to 1, then the basic energy function is the activation function multiplied by -1 (Fig. 2). It decreases if the activation level increases.

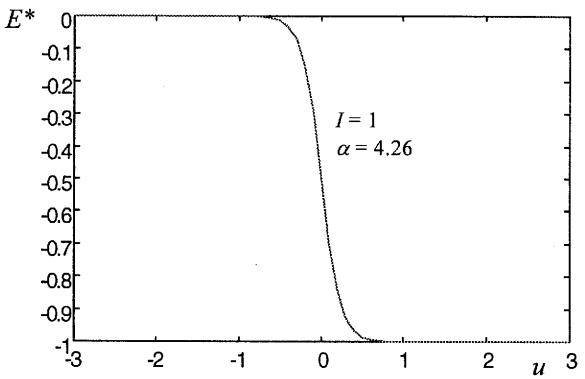

Fig. 2. The basic energy function for one neuron with external input equal to 1 and gain coefficient equal to 4.26.

We set $g^{-1+}(x) = -g^{-1}(x)$ if $g^{-1}(x) < 0$ and $g^{-1+}(x) = g^{-1}(x)$ if $g^{-1}(x) \geq 0$. In the analog UHANN the additional term $E^{\alpha}(u) = \sum_{m=1}^{M} \int_0^{g_m(u_m)} g_m^{-1}(\xi_m)\, \mathrm{d}\xi_m$ of the energy function has to be considered. For $M = 1$ and a positive value of the gain coefficient, $E^{\alpha}(u) = \int_0^{g(u)} g^{-1}(\xi)\, \mathrm{d}\xi$ can be interpreted as the area under the curve

$u^+ = g^{-1+}(x)$ (Fig. 2). If the output is changed from 0 to 1, then the activation energy can be expressed in the form

$$E^{\alpha}(u) = \int_0^{g(u)=1} g^{-1}(\xi)\,\mathrm{d}\xi = \int_0^{g(u)=0.5} g^{-1}(\xi)\,\mathrm{d}\xi + \int_{0.5}^{g(u)=1} g^{-1}(\xi)\,\mathrm{d}\xi$$
$$= 2\int_0^{g(u)=0.5} g^{-1}(\xi)\,\mathrm{d}\xi = 2\int_{0.5}^{g(u)=1} g^{-1}(\xi)\,\mathrm{d}\xi$$

Hence, if the activation level of the neuron u has a large value (positive or negative), the neural output x is close to 0 or 1, and its own activation energy is large. But if $u = 0$ and $x = g(u) = 0.5$, then the neural activation energy attains its minimum. That is why a single neuron without any external inputs and synaptic connections gets $x = 0.5$ at its equilibrium point, although its starting activation level u has a large absolute value. A single separate neuron tends to a calm state, where $u^c = 0$ and $x^c = 0.5$. If an external input I is added and I is constant during a neuron's relaxation, then the calm state for this neuron is $u^c = I$ and $x^c = g(I)$. Then the external input I has the main influence on the calm state which is an equilibrium point, too. The same result can be produced from the motion equation analysis.

In neural optimization the designed networks should avoid saddle points (false attractors) of the energy function, because even feasible solutions cannot be then obtained in some cases. In Fig. 4 a saddle point of the basic energy function is presented. Only two neurons are taken. Synaptic weights are equal to -2 and external inputs are equal to 1. For this case the basic energy function has two global minimizers and one saddle point. If the UHANN starts from an initial state, then after relaxation it reaches an equilibrium point. If this equilibrium point is a global minimizer of the basic energy function, then the optimal solution of the optimization problem can be found. But if at an equilibrium point of motion equations a saddle point of the basic energy function is reached, then $x_1 = x_2 = 0.5$ and formal constraints $x_1, x_2\{0,1\}$ are not satisfied. Hence the UHANN for optimization should avoid saddle points. In (Balicki and Kitowski, 1996) the following result to settle this problem has been stated.

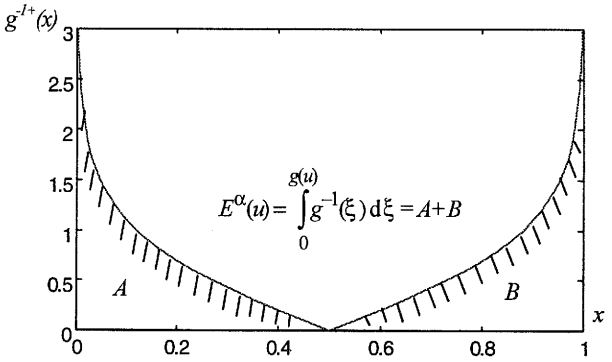

Fig. 3. An activation energy for a single neuron.

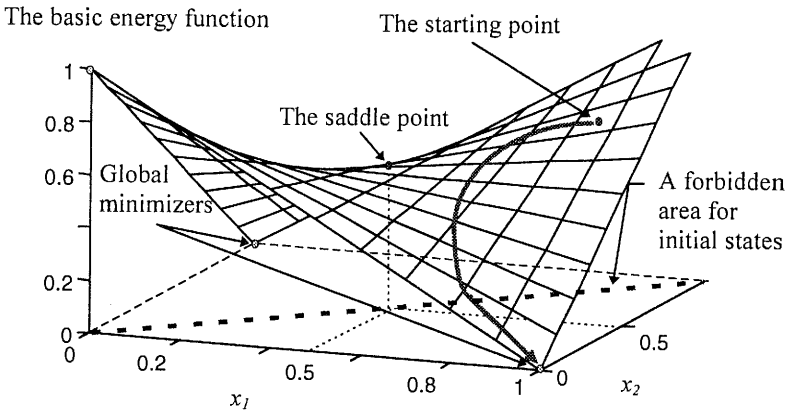

Fig. 4. The basic energy function.

Theorem 1. *Assume that* $\Delta t \to 0$ *and* $w \leq 0$. *If the initial activation levels* $u_1(t_0), \ldots, u_m(t_0), \ldots, u_M(t_0)$ *in a uniform analog Hopfield network have different values* $u_m(t_0) \neq u_n(t_0)$, $m = \overline{1, M}$, $n = \overline{1, M} \backslash \{m\}$ *and the Euler procedure (7) to find equilibrium points is used, then the activation levels during the convergence to the equilibrium points have different values.*

Proof. If the initial activation levels $u_1(t_0), \ldots, u_m(t_0), \ldots, u_M(t_0)$ in a uniform analog Hopfield network have different values $u_m(t_0) \neq u_n(t_0)$, $m = \overline{1, M}$, $n = \overline{1, M} \backslash \{m\}$ and the Euler procedure (7) is used to find equilibrium points, then the activation levels after the first iteration $k = 1$ at $t_0 + \Delta t$ ($\Delta t \to 0$) are calculated as follows:

$$u_m(t_k + \Delta t) = \left(1 - \frac{\Delta t}{\eta}\right) u_m(t_k) + w \Delta t \sum_{\substack{n=1 \\ n \neq m}}^{M} x_m(t_k) + I \Delta t, \quad m = \overline{1, M}$$

Let us change the indices of neurons to hold the decreasing adjustment:

$$u_1(t_0) > \cdots > u_m(t_0) > \cdots > u_M(t_0)$$

Then

$$x_1(t_0) > \cdots > x_m(t_0) > \cdots > x_M(t_0)$$

and

$$\sum_{\substack{n=1 \\ n \neq 1}}^{M} x_n(t_0) < \cdots < \sum_{\substack{n=1 \\ n \neq m}}^{M} x_n(t_0) < \ldots < \sum_{\substack{n=1 \\ n \neq M}}^{M} x_n(t_0)$$

If $w_{mn} = w \leq 0$ for $n \neq m$, and $n, m = \overline{1, M}$, then for $t_1 = t_0 + \Delta t$

$$u_1(t_0 + \Delta t) > \cdots > u_m(t_0 + \Delta t) > \cdots > u_M(t_0 + \Delta t)$$

Since at the moment $t_1 = t_0 + \Delta t$ the decreasing adjustment of activation levels is kept, similarly at $t_2 = t_1 + \Delta t$ the decreasing adjustment of activation levels holds, too. At any time t_k during the convergence to equilibrium points the activation levels have different values $u_1(t_k) > \cdots > u_m(t_k) > \cdots > u_M(t_k)$ which completes the proof. ■

Therefore, if $u_1(t_0) = u_2(t_0)$, then $x_1(t_0) = x_2(t_0)$ in the UHANN with two neurons (Fig. 4). Since Hopfield networks minimize their energy functions according to the steepest-descent method, the state trajectory converges to a saddle point from each balanced starting point (c, c), where $c \in (0, 1)$. In Fig. 4 the unfeasible area in zero-one optimization for initial states of the UHANN is indicated. Because in combinatorial optimization problems the solutions should be situated at a vertex of the hypercube $[0, 1]^M$, different values of the initial activation levels in the UHANN are preferred: $u_m(t_0) \neq u_n(t_0)$, $m = \overline{1, M}$, $n = \overline{1, M} \backslash \{m\}$.

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

Theorem 2. (Balicki and Kitowski, 1996) *If* $\sum_{m=1}^{M} x_m = L$ *for* $x_m \in \{0, 1\}$, $l \leq M$, $L = 0, 1, 2, \ldots, M$, *and a uniform analog Hopfield network has the following parameters:*

$$\begin{aligned} w_{mj} &= w = -2, \qquad m, j = \overline{1, M}, \qquad m \neq j \\ I_m &= I = 2L - 1, \qquad m = \overline{1, M} \end{aligned} \tag{9}$$

where M *is the number of neurons, then*

$$E(x) = h(x) + L^2$$

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 the basic energy function of this UHANN, and

$$h(x) = \left(L - \sum_{m=1}^{M} x_m \right)^2 + \sum_{m=1}^{M} x_m (1 - x_m)$$

is the penalty function for the constraints.

The UHANN with synaptic weights equal to -2 and non-negative external inputs calculated according to the rule $I = 2L - 1$ can be called UHANN/L/M, because for their design the pair (L, M) has to be known, only. Consequently, the signals from other neurons are converted and their absolute values are increased. Moreover, each neuron has its non-negative constant input which forces the activation level $u^* = I$ at an equilibrium point.

5.1. Minimization of the Basic Energy Function in UHANN/L/M

In Fig. 5 a minimization of the basic energy function in the UHANN/1/50 is presented. The trajectory of the energy decreasing versus the iteration number of the Euler method is considered. The following parameters are taken: $u(t_0) = [50, 49, 48, \ldots, 3, 2, 1]^T$, $\alpha = 10$, $\eta = 1$, $\Delta t = 0.01$, k_{stop}, i.e. the stopping condition $E(t_k) \leq \varepsilon$, $\varepsilon < 0.001$.

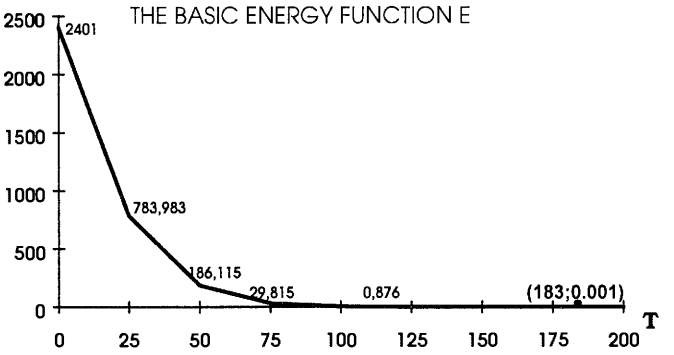

Fig. 5. Minimization of an energy function.

The considered equation is satisfied when the value of the basic energy function does not exceed a small threshold ε. Then $E(x) = -(1/2)\sum_{n=1}^{M}\sum_{m=1}^{M} x_n x_m - I\sum_{m=1}^{M} x_m \leq \varepsilon$ and $h(x) = (L - \sum_{m=1}^{M} x_m)^2 + \sum_{m=1}^{M} x_m(1 - x_m) \leq \varepsilon$. If x_m is equal to 0 or 1 with a smaller accuracy than ε_1, then $(L - \sum_{m=1}^{M} x_m)^2 \leq \varepsilon + M\varepsilon_1$. Thus the general constraint $\sum_{m=1}^{M} x_m = L$ is satisfied with an accuracy no worse than $\varepsilon + M\varepsilon_1$.

For the parameters of the UHANN/L/M the basic energy function is minimized from the initial value $E(t_0) = 2401$ to the value $E(t_k) \leq \varepsilon$ ($\varepsilon = 0.1$) during 146 iterations. For a smaller value ($\varepsilon = 0.001$) a larger number of iterations (183) is needed. In Fig. 5 we can see that the trajectory of the basic energy function decreases exponentially. At the beginning it decreases very fast, and then it approaches zero very slowly.

In Fig. 6 the trajectories of the chosen activation levels $u_1(t)$ and $u_2(t)$ during the energy minimization are presented. The activation levels of 49 neurons decrease from the given initial values to the values $-1 \leq u(t) \leq -1 + \varepsilon$ at an equilibrium point.

An adequate trajectory for these neurons is $u_2(t)$ which tends to $\eta = -1$. If the initial values are greater than -1, the activation levels decrease. If the activation levels are less than -1, the activation levels increase. Only the first neuron with the greatest initial activation level has a distinct sort of trajectory $u_1(t)$. After it attains its minimizer at iterations 137, 138, and 139, its activation level increases to $\eta = 1$ (Fig. 7).

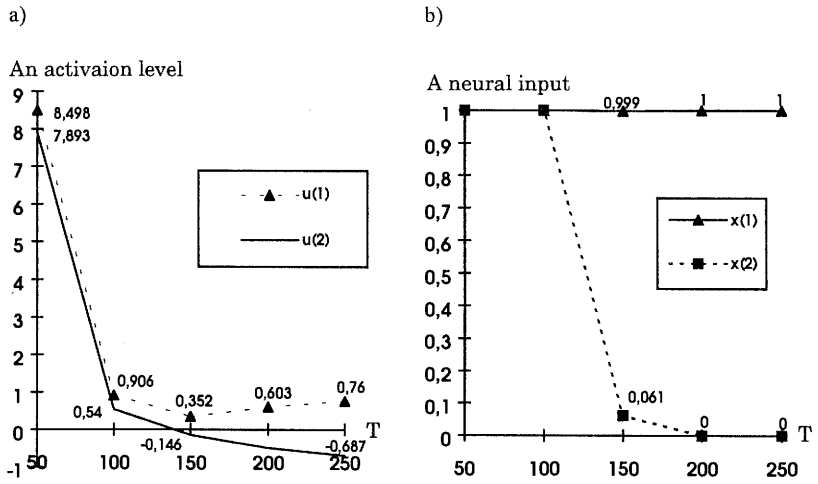

Fig. 6. Trajectories of the two first neurons in the UHNN/1/50: a) activation levels and b) neural outputs.

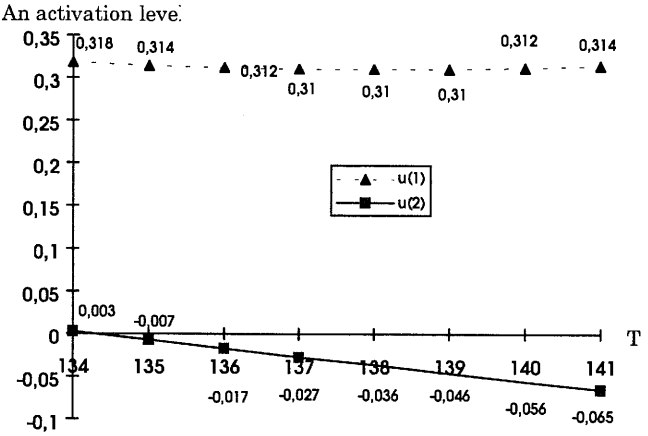

Fig. 7. A competition of two neurons on the minimizer's trajectory of the activation level for the first neuron in the network UHNN/1/50.

The neurons in the UHANN/L/M with feasible parameters compete, because they try to get final states to minimize the energy function. A neuron can obtain from given initial states either the state 'off' ($x = 0$) or the state 'on' ($x = 1$). For the first case the individual preferences argue with the whole network's preferences, and at iteration 140 the gradient of the activation level changes its sign to become 1 at iteration 97.

5.2. Initial States

There are two basic problems related to the minimization of the basic energy function of networks UHANN/L/M. Firstly, parameters for finding a global minimizer of the energy function have to be found. Secondly, the network UHANN/L/M should fix this minimizer as soon as possible, e.g. it needs a smallest number of iterations $K_{\max}$.

An initial state with distinguished activation levels guarantees for feasible values of the other network parameters that this network finds an equilibrium point with a local or a global minimizer. The shorter the distance to an equilibrium point, the faster the approach to it is (see Table 1).

Table 1. The influence of the initial state $u(t_0)$ on the number of iterations $K_{\max}$ for finding equilibrium points, where $\alpha = 10$, $\Delta t = 0.01$, $\varepsilon < 0.001$, $E(t_k) \leq \varepsilon$, $\eta = 1$.

No.	Initial state formula $u_i(t_0)$ for $i = \overline{1, M}$	$u(t_0)$	$K_{\max}$
1	$1000 + M - i + 1$	$[1050, 1049, \ldots, 1002, 1001]^T$	320
2	$100 + M - i + 1$	$[150, 149, \ldots, 102, 101]^T$	206
3	$M - i + 1$	$[50, 49, \ldots, 2, 1]^T$	183
4	$M - i + 1 - \frac{M}{2}$	$[25, 24, 23, \ldots, -23, -24]^T$	173
5	$\left(M - i + 1 - \frac{M}{2}\right)/10$	$[2.5, 2.4, 2.3, \ldots, -2.3, -2.4]^T$	75
6	$\left(M - i + 1 - \frac{M}{2}\right)/100$	$[0.25, 0.24, 0.23, \ldots, -0.23, -0.24]^T$	149
7	$1 - \frac{2(i-1)}{M}$	$[1, 0.96, 0.92, \ldots, -0.92, -0.96]^T$	73
8	$1 + \left(M - i + 1 - \frac{M}{2}\right)/100$ for k neurons $-1 + \left(M - i + 1 - \frac{M}{2}\right)/100$ for $M - k$ neurons	$[1.25, -0.76, -0.75, \ldots, -1.23, -1.24]^T$	1

If the largest interval for the initial state generation is used, then the maximal number of iterations for several formulae does not exceed 350 iterations. During numerical experiments real numbers from the EXTENDED data type in Turbo Pascal were used. This sensitivity with respect to the initial states is very similar to the same property of gradient optimization techniques.

5.3. The Gain Coefficient in the Activation Function

The gain coefficient α in the activation function exerts an important influence on the convergence of the UHANN/L/M to an equilibrium point. We have $\tanh(\alpha u_m) =$

$(e^{\alpha u_m} - e^{-\alpha u_m})/(e^{\alpha u_m} + e^{-\alpha u_m})$. The activation function $g_m(u_m) = (1/2)[1 + \tanh(\alpha_m u_n)]$, $m = \overline{1, M}$ has the domain $(-\infty, \infty)$ and range (0,1). It is continuous, odd, and increasing. Its graph $x = g(u)$ has two asymptotes $x = 0$ and $x = 1$. Owing to an increase in the gain coefficient, the shape of the activation function is very similar to the unit-step function. The relationship between the gain coefficients and the range of the visible value period $[-u_{\mathrm{gr}}, u_{\mathrm{gr}}]$ can be written as

$$u_{\mathrm{gr}} = \frac{u_{\mathrm{gr}}(e, 1)}{\alpha} \tag{10}$$

where $u_{\mathrm{gr}}(\varepsilon, 1)$ is the activation level such that for $\alpha = 1$ one has $\varepsilon = g(u_{\mathrm{gr}}(\varepsilon, 1))$.

In practical calculations the gain coefficients take on very large values: 10000 (Ciepliński and Jędrzejek, 1994) or 20000 (Hertz *et al.*, 1993). But for the UHNN/1/5 with the stopping criterion k_{stop} such that $E(t_k) \leq \varepsilon = 0.001$ it is enough to take $\alpha > \alpha_{\mathrm{gr}} = 4.26$. When the gain coefficient increases, the number $K_{\max}$ for finding equilibrium points decreases. For instance, for the UHNN/1/5 with $\alpha_{\mathrm{gr}} = 4.26$ we have $K_{\max} = 424$. If $\alpha = 5$, then $K_{\max} = 229$. Similarly, for $\alpha = 10$ we have $K_{\max} = 140$. If the gain coefficient is still increased, then an increase in the convergence is not so large, because for $\alpha = 100$ one has $K_{\max} = 101$, and for $\alpha = 1000$ we get $K_{\max} = 98$. If the gain coefficient is greater than 100, then it does not influence the speed of convergence to equilibrium points. However, if $\alpha < \alpha_{\mathrm{gr}}$, then networks UHANN/L/M cannot find the minimizers of the energy function.

5.4. The Length of the Integration Step

A recursive procedure of solving differential equations can be written in the form

$$u_m(t_k + \Delta t) = u_m(t_k) - \Delta t\left(\frac{u_m(t_k)}{\eta_m} - w \sum_{\substack{n=1 \\ n \neq m}}^{M} g(u_n(t_k)) - I\right), \quad m = \overline{1, M} \tag{11}$$

If the integration step Δt increases, then $K_{\max}$ decreases in UHANN/L/M. For instance, for $\Delta t = 0.001$ there is $K_{\max} = 1324$, for $\Delta t = 0.01$ we have $K_{\max} = 132$, and for $\Delta t = 0.1$ we get $K_{\max} = 13$. In particular, for $\Delta t = 0.5$, we get the fastest performance, i.e. $K_{\max} = 3$. However, for $\Delta t > 0.5$ this network *oscillates* between two states when none of the minimizers of the basic energy function is obtained. Consequently, for a given value of the adaptive time parameter τ, an integration step Δt should be determined.

If the stopping criterion is k_{stop} such that $E(t_k) \leq \varepsilon$ for $\varepsilon > 0$, then the number of iterations $K_{\max}(E(t_k) \leq \varepsilon)$ increases as the accuracy increases (ε decreases). But, if the stopping criterion is formulated as the condition that all neural outputs reach 0 or 1 with a given accuracy, then the number of iterations is larger, but still close to $K_{\max}(E(t_k) \leq \varepsilon)$, because from the formula for the basic energy we can find a relationship between the accuracy of the outputs and the energy. But, to obtain the same accuracy of neural activation levels, a few times larger number of iterations is required.

5.5. The Passive Coefficient η

Incorrect values of the passive coefficient η can cause that a network UHANN/L/M cannot attain its equilibrium points. For instance, for $\eta = -10$ a minimizer of the basic energy function is reached after 20 iterations for the UHNN/3/5, but an equilibrium point of the motion equation is not obtained, and for the absolute values of the activation levels it tends to infinity. If η decreases to zero, then the activation levels increase faster. For $\eta = 0.1$ the network *oscillates*. Moreover, for $0.1 < \eta < 0.5$ the minimizer of the basic energy function is not obtained at equilibrium points.

A passive coefficient has feasible values for $\eta \geq 0.5$, and it permits to find a minimizer of the basic energy. For feasible values of the passive coefficient $\eta \geq 0.5$, the activation levels at an equilibrium point equal η or $-\eta$. A substantial increase in the feasible parameter η causes a small increase in the number of iterations required to find a minimizer of the energy function with accuracy ε. For example, for $\eta = 1$ we have $K_{\max}(E(t_k) \leq \varepsilon) = 12$, and for $\eta = 10^4$ we have $K_{\max}(E(t_k) \leq \varepsilon) = 17$. A considerable increase in the feasible parameter η causes a linear increase in the number of iterations required to find equilibrium points with a given activation level accuracy ε.

All cases of a network UHANN/L/M ($M \leq 100$) are studied. For the initial states, the following formula was used:

$$u_m(t_0) = \frac{1}{10}\left(M - i + 1 - \frac{M}{2}\right), \qquad m = \overline{1, M} \tag{12}$$

Moreover, the recommended parameters are $\alpha = 100$, $\Delta t = 0.2$, $\eta = 1$, k_{stop} is the condition $E(t_k) \leq \varepsilon$ for $\varepsilon = 0.01$. For the worst case, $K_{\max}(E(t_k) \leq \varepsilon)$ for solving a general equation is equal to 5. These experimental results confirm that neural networks can be designed as a very efficient method to solve numerical problems. In particular, for the network UHANN/L/M the number of neurons M does not influence the increase $K_{\max}(E(t_k) \leq \varepsilon)$. Several randomly chosen cases for $100 < M < 1000$ confirmed this rule.

Accordingly, the time complexity of the neural method for solving the considered equations is $O(K_{\max}M)$ for parallel algorithms or $O(K_{\max}M^2)$ for sequential algorithms. At each basic iteration, the formula for new states

$$u_m(t_k + \Delta t) = u_m(t_k) - \Delta t \left(\frac{u_m(t_k)}{\eta_m} - w \sum_{\substack{n=1 \\ n \neq m}}^{M} g\big(u_n(t_k)\big) - I \right)$$

is calculated. The most critical part of this calculation time is due to the activation procedure $g(u_n(tk))$. It has to be called $M-1$ times. If we assume that the activation procedure takes one time unit, then the other times may be neglected. Hence, in software implementations, a fast procedure for the activation function has to be used. Instead of the Euler method, the first-order Runge-Kutta method or another method for solving ordinary differential equations can be used. Żurada *et al.* (1996) propose a relaxation method for the HANN simulation.

6. Hopfield Networks for Solving Linear Inequalities

In the sequel, we consider the inequality constraint $\sum_{j=1}^{J} x_{ij}^{\pi} \leq 1$ for $i = \overline{1,2}$. In many optimization problems, its more general form can be considered:

$$\sum_{m=1}^{M} x_m \leq L \tag{13}$$

To solve this inequality, an extended uniform Hopfield network can be used. Based on Theorem 2, the following theorem can be proved (Balicki and Kitowski, 1996).

Theorem 3. (Balicki and Kitowski, 1996) *If* $\sum_{m=1}^{M} x_m \leq L$ *for* $x_m \in \{0,1\}$, $L \leq M$, $L = 0,1,2,\ldots M$, *and a uniform analog Hopfield network has the following parameters:*

$$\begin{aligned} &w_{mj} = -2, \qquad m,j = \overline{1,M+L}, \qquad m \neq j \\ &I_m = 2L-1, \qquad m = \overline{1,M+L} \end{aligned} \tag{14}$$

where $M+L$ *is the number of neurons, then*

$$E(x) = h(x) + L^2$$

where

$$E(x) = -\frac{1}{2} \sum_{n=1}^{M+L} \sum_{m=1}^{M+L} x_n x_m - I \sum_{m=1}^{M+L} x_m$$

is the basic energy function of this UHANN, and

$$h(x) = \Big(L - \sum_{m=1}^{M+L} x_m\Big)^2 + \sum_{m=1}^{M+L} x_m(1-x_m)$$

is the penalty function for this inequality.

From this theorem, a uniform Hopfield network for the inequality $\sum_{m=1}^{M} x_m \leq L$ can designed by adding L dummy neurons. Hence all $M+L$ neurons compete, since only L neurons can be chosen. That is why, in this UHANN/L/M+L, the other parameters can be taken similarly to the UHANN/L/M.

7. Neural Unconstrained Minimization

Let us consider the linear performance index $F_1(x) = \sum_{i=1}^{2} \sum_{j=1}^{J} \kappa_j x_{ij}^{\pi}$. At the beginning, we examine the following optimization problem:

$$\min_{x \in X} \sum_{i=1}^{2} \sum_{j=1}^{J} \kappa_j x_{ij}^{\pi} \tag{15}$$

If $F_1(x) = \sum_{i=1}^{2} \sum_{j=1}^{J} \kappa_j x_{ij}^{\pi}$ is minimized by an analog Hopfield network under the constraints $\sum_{i=1}^{2} x_{vi} = 1$ for $v = \overline{1,V}$, and $\sum_{j=1}^{J} x_{ij}^{\pi} \leq 1$ for $i = \overline{1,2}$, then only the constraints $\sum_{j=1}^{J} x_{ij}^{\pi} \leq 1$ for $i = \overline{1,2}$ are active, since the decision variables x_{vi} cannot change the performance index $\sum_{i=1}^{2} \sum_{j=1}^{J} \kappa_j x_{ij}^{\pi}$. Accordingly, we can use two networks HANN/1/J+1 to satisfy the constraint $\sum_{j=1}^{J} x_{ij}^{\pi} \leq 1$, $i = \overline{1,2}$. But in these networks the external inputs are modified according to the formula:

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

where $\Delta I(x_{ij}^{\pi}) = \kappa_j / \kappa_{\max}$, and $\kappa_{\max}$ is the cost of the most expensive processor.

To understand the above formula, let us notice that if in the UHANN/L/M one neuron has an external input greater than the others, then the corresponding neural output becomes 1 at an equilibrium point. Therefore, this is a way to prefer neurons related to cheaper processors. Thus the additional term decreases the external inputs when the cost increases. If in a network UHANN/L/M all external inputs are increased by a small value, then L neurons are still chosen at an equilibrium point. For $L = 0$, we have $I = 2L - 1 = -1$. For $L = 1$, we get $I = 1$. So, according to the increase $L = 0, 1, 2, 3, 4, \ldots$, we have $I = -1, 1, 3, 4, 6, \ldots$, respectively. For a bounded L, there exists an interval for the feasible external inputs $(2L - 2, 2L)$. The values of the changed external inputs should fall into this interval.

In Fig. 8, a minimization example for an energy function is given. We have $J = 5$, $\alpha = 100$, $\eta = 1$, $\Delta t = 0.2$. Moreover, the cost vector is $\kappa = [5, 4, 3, 2, 1]^T$, From the procedure $I(x_{ij}^{\pi}) = 2J + 1.5 - (\kappa_j / \kappa_{\max})$, $j = \overline{1,J}$, $i = 1, 2$, the external input vector is $I = [0.5, 0.7, 0.9, 1.1, 1.3]^T$. For each node number, a separate network UHNN/1/J+1 is considered. Let us denote this network by HNN/F1/C. The basic energy function decreases very fast, and in the fifth step it attains its minimum.

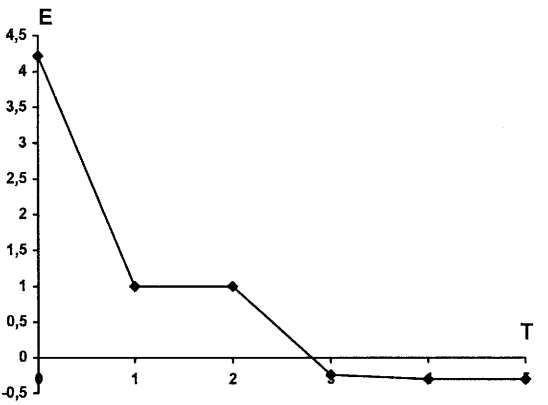

Fig. 8. Minimization of the basic energy function in an optimization network HNN/F1/C.

In Fig. 9 activation level trajectories are shown. The neurons compete around one state with a positive activation value. Since $\kappa = [5, 4, 3, 2, 1]^T$, we expect that the inputs $I = [0.5, 0.7, 0.9, 1.1, 1.3]^T$ cause the preferences for the fifth neuron. At the beginning, the fifth neuron has the lowest activation value. But, because of the highest external input, it gets the highest activation level, and it wins. It is interesting that the fifth neuron gets the activation level equal to $I_5 = 1.3$ at an equilibrium point. However, the other neurons have negative activation levels $u_m = -2 + I_m$ for $m \neq 5$. The above minimization example confirms that the Hopfield network can be applied as a numerical tool for solving some optimization problems. A small number of iterations suggests its large capabilities.

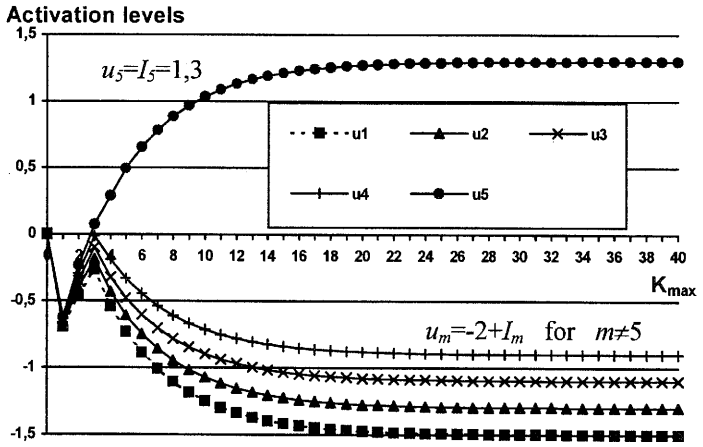

Fig. 9. A neurons competition in HNN/F1/C.

8. Hopfield Networks for Quasi-Quadratic Optimization

Let us consider the network HANN/F2 for the minimization of the function $F_2(x)$ and assume $x \in [\varepsilon, 1-\varepsilon]^M$, $\varepsilon \to 0^+$. In Fig. 10, the structure of this network is presented. The correct values of synaptic weights from the comparison of the basic energy function with the performance index

$$F_2(x) = \sum_{j=1}^{J} \sum_{v=1}^{V} \sum_{i=1}^{2} t_{vi} x_{vi} x_{ij}^{\pi} + \sum_{v=1}^{V} \sum_{u=1}^{V} \sum_{i=1}^{2} \tau_{vu} x_{vi} (1 - x_{ui})$$

were taken. In this network there are no external inputs (Balicki and Kitowski, 1996).

If a more complex optimization problem is considered, e.g. when a time criterion $F_2(\cdot)$ is minimized under the constraints $\sum_{i=1}^{2} x_{vi} = 1$, $v = \overline{1,V}$ and $\sum_{j=1}^{J} x_{ij}^{\pi} \leq 1$, $i = \overline{1,2}$. This optimization problem can be transformed into an unconstrained optimization problem. The energy functions of neural networks designed for constraint satisfaction or for minimization of a performance index can be

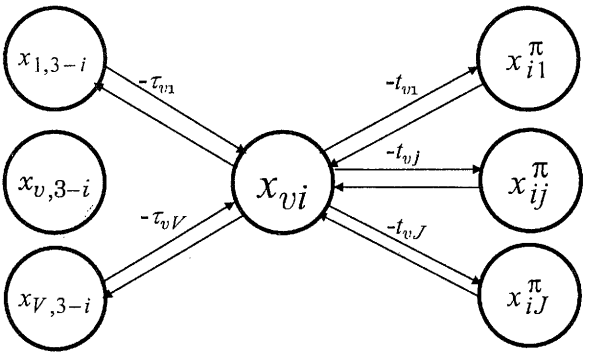

Fig. 10. Synaptic connections around the neuron x^m_{v1} in the analog Hopfield network HANN/F2 for minimization of the function $F_2(x)$.

aggregated in a penalty function:

$$E(x\beta) = F_2(x) + \sum_{v=1}^{V} \beta_v E_v(x) + \sum_{i=V+1}^{V+2} \beta_i E_i(x) \tag{16}$$

where β_v, β_i are penalty coefficients, E_v stands for the energy function of the network UHNN/1/2 designed to satisfy the constraint $\sum_{i=1}^{2} x_{vi} = 1$, and E_i denotes the energy function of the network UHNN/1/J designed to satisfy the constraint $\sum_{j=1}^{J} x^{\pi}_{ij} \leq 1$.

In Fig. 11, a network HNN/F2/C for minimization of the problem under consideration is presented. The penalty coefficient have to be known. They can be found by systematically increasing them from an initial value β_0 (usually chosen as 1). If one of the energy functions related to the v-th constraint is greater then 0, then this constraint is not satisfied, and the corresponding parameter is increased by $\Delta\beta$ (usually 0.05 or 0.1). This process is stopped if all the energy functions of the partial constraints are equal to zero.

In Fig. 12 experimental results with energy-function trajectories are presented. For $\beta = [2.55, 2.45, 2.25, 2.25, 7.35, 7.30]^T$ at the equilibrium point $x^* = [1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0]^T$ an optimal value of the performance index $F_2(x^*) = 7.0$ was obtained. Moreover, $\alpha = 200$. From this figure, we can see that the partial energy functions compete with one another. If a penalty coefficient of the partial network is increased, then the role of this network increases, too. Then the signals generated by this network try to dominate the signals from other networks.

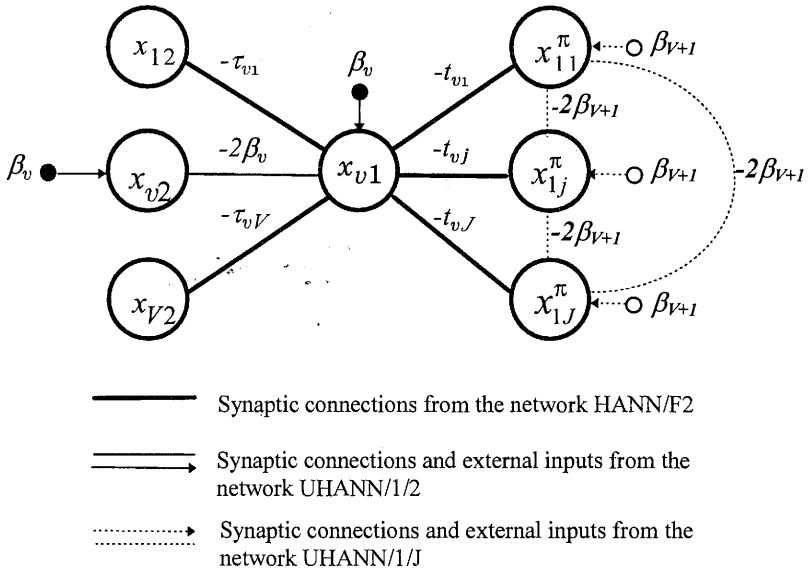

Fig. 11. Synaptic connections and external inputs for the neuron x^m_{v1} in HANN/F2/C.

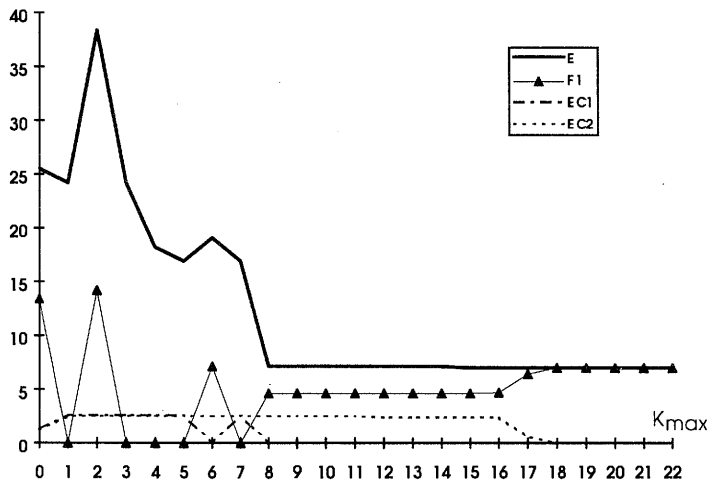

Fig. 12. Trajectories of the energy functions related to a performance index and constraints.

9. Hopfield Neural Networks for Generating Pareto-Optimal Allocations by the Non-Negative Convex Combination Method

The most widely-used method to generate Pareto-optimal solutions is a non-negative convex combination of the scalar partial functions:

$$\min_{x \in X} \left\{ \sum_{n=1}^{N} \alpha_n F_n(x) \right\} \tag{17}$$

under the constraints

$$\alpha_n \geq 0, \qquad n = \overline{1,N}, \qquad \sum_{n=1}^{N} \alpha_n = 1$$

where α_n is a combination coefficient.

A point $x^* \in X$ is the *global efficient point* (Pareto-optimal solution) for the mapping $F : X \to \mathbb{R}^N$ if there exists no other point $x \in X$ such that

$$F_n(x) \leq F_n(x^*), \qquad n = \overline{1,N}$$

$$F_i(x) \leq F_i(x^*), \qquad \text{for some } j \in \overline{1,N}$$

A point $x^* \in X$ is the *local efficient point* (Pareto-suboptimal solution) for the mapping $F : X \to \mathbb{R}^N$ if there exists an $\varepsilon_0 > 0$, such that in the neighbourhood $N(x^*, \varepsilon_0)$ of x^* there exists no other point $x \in X$ such that

$$F_n(x) \leq F_n(x^*), \qquad n = \overline{1,N}$$

$$F_i(x) \leq F_i(x^*), \qquad \text{for some } j \in \overline{1,N}$$

When no distinction is made between local and global efficient points, this point is called the *efficient point*. The efficient set X^* contains all efficient points. The efficient boundary Y^* of the output set Y is related to the feasible-solution set X according to

$$Y^* = \left\{ y^* \in Y \mid y^* = F(x^*),\ x^* \in X \right\}$$

To find one local efficient point for the above problem, we can use a Hopfield ANN (PHANN). The PHANN can represent one Pareto-optimal solution at an equilibrium point. An energetic function for the PHANN is constructed according to the formula

$$E(x,\beta) = \sum_{n=1}^{N} \alpha_n F_n(x) + \sum_{v=1}^{V} \beta_v E_v(x) + \sum_{i=V+1}^{V+2} \beta_i E_i(x) \tag{18}$$

For $N = 2$, the combination coefficients can be systematically changed within the range (0,1). The performance index and penalty functions can be presented by

separate partial energetic functions. For the performance index in the non-negative convex combination method, we get the following formula for the separate energetic function:

$$\sum_{n=1}^{N} \alpha_n F_n(x) = -\frac{1}{2} \sum_{r=1}^{M} \sum_{m=1}^{M} w_{rm}^n x_r x_m - \sum_{r=1}^{M} I_r^n x_r \tag{19}$$

where w_{rm}^n is the synaptic weight from the r-th neuron to the m-th neuron related to the multiplied performance index $E_n(x, \alpha_n) = \alpha_n F_n(x)$, and I_r^n denotes the external input of the r-th neuron related to the multiplied performance index $E_n(x, \alpha_n) = \alpha_n F_n(x)$.

Similarly, we can obtain the formulae for the partial energetic functions of the constraint satisfaction. Hence we have the global basic energetic function of the PHANN:

$$\begin{aligned} E = & -\frac{1}{2} \sum_{r=1}^{M} \sum_{m=1}^{M} \left(\sum_{n=1}^{N} \alpha_n w_{rm}^n) + \sum_{l=1}^{V+2} \beta_l w_r m^l \right) x_r x_m \\ & - \sum_{r=1}^{M} \left(\sum_{n=1}^{N} \alpha_n I_r^n + \sum_{l=1}^{V+2} \beta_l I_r^l \right) x_r \end{aligned} \tag{20}$$

In the above formula, combination coefficients are systematically selected from 0 to 1. If $\alpha_1 = 1$ and $\alpha_2 = 0$, then we get a network HANN/F1/C. If $\alpha_1 = 0$ and $\alpha_2 = 1$, then we get a network HANN/F2/C. Similarly, we can obtain other synaptic weights and external inputs which are related to other constraints and performance indices.

10. HANN for Generating Compromise Solutions with $p = 1$

If an ideal point is normalized as follows:

$$\overline{y}_n = \frac{y_n}{y_n^0}, \qquad y_n^0 \neq 0, \qquad n = \overline{1, N} \tag{21}$$

then the compromise function for $p = 1$ is as shown below:

$$K_1 = \sum_{n=1}^{2} |1 - \overline{y}_n| \tag{22}$$

The problem of finding compromise solutions for $p = 1$ can be transformed into the minimization problem

$$\min_{x \in X} K_1(x) \tag{23}$$

To generate compromise solutions with $p = 1$, a Hopfield ANN (C1HANN) can be used. The C1HANN has $N = M + S$ neurons. The compromise criterion with

$p = 1$ can be written down as

$$K_1(x) = \frac{1}{\overline{y}_1^0} \sum_{i=1}^{2} \sum_{j=1}^{J} \kappa_j x_{ij}^{\pi} + \frac{1}{\overline{y}_2^0} \sum_{i=1}^{2} \sum_{v=1}^{V} \sum_{j=1}^{J} t_{vj} x_{ij}^{\pi} x_{vi}^{m}$$

$$+ \sum_{i=1}^{2} \sum_{v=1}^{V} \sum_{u=1}^{V} t_{vu}(1 - x_{ui}^{m}) x_{vi}^{m} - 2 \tag{24}$$

The above compromise criterion can be used to design a C1HANN, because it is taken as an energy function of the Hopfield network. Ideal points are given from the networks HANN/F1/C and HANN/F2/C. In this way, a special class of Pareto solutions can be found.

11. Concluding Remarks

In this paper, genetic methods and artificial neural networks for solving several operation allocation problems have been proposed. Formulae to determine the values of synaptic weights and external inputs for networks satisfying basic constraints and performance indices are presented. The most widely-used non-negative convex combination methods of finding efficient solutions have been considered. This method can be implemented as an artificial neural network PHANN. The recurrent ANN at an equilibrium point represents an efficient point. Moreover, the presented approach was applied to solving compromise optimization problems with parameter $p = 1$ and the results obtained have shown that this direction is very promising. Recent results confirm possibilities of finding suboptimal solutions for combinatorial problems by Hopfield models (Kaznachey and Jagota, 1997).

ANN's were simulated in a PC environment without neural accelerators. It is possible to use neural accelerators and to improve the performance of neural calculations. For the integration step length equal to a few μs, a real-time solution can be obtained.

The designed HANN 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. Apart from the paper (Balicki and Kitowski, 1996), a similar approach was independently proposed in (Funabiki *et al.*, 1997).

References

Abe S. (1996): *Convergence acceleration of the Hopfield neural networks by optimizing integration step sizes.* — IEEE Trans. Syst. Man and Cybern., Part B: Cybern., Vol.28, No.1, pp.194–201.

Aiyer S.V.B., Niranjan H. and Fallside F. (1990): *A theoretical investigation into the performance of Hopfield model.* — IEEE Trans. Neural Networks, Vol.1, No.3, pp.204–215.

Ameljańczyk A. (1986): *Multicriteria Optimization.* — Warsaw: WAT Press, (in Polish).

Bac F.Q. and Perov V.L. (1993): *New evolutionary genetic algorithms for NP-complete combinatorial optimization problems.* — Biol. Cybern., Vol.69, pp.229–234.

Balicki J. and Kitowski Z. (1996): *Genetic-neural multiobjective optimization for resource allocation problems.* — Proc. Int. Conf. *Intelligent Technologies in Human-Related Sciences*, Leon, Spain, July 5–7, pp.18–22.

Ciepliński L. and Jędrzejek C. (1994): *Statistical physics approach to optimization problems.* — Appl. Math. Comp. Sci., Vol.4, No.3, pp.423–431.

Cohen A. and Grossberg S. (1983): *Absolute stability of global pattern formation and parallel memory storage by competitive neural networks.* — IEEE Trans. Syst. Man Cybern., Vol.SMC–13, pp.815–825.

Funabiki N., Kitamichi J. and Nishikawa S. (1997): *An evolutionary neural network algorithm for max cur problems.* — Proc. Int. Conf. *Neural Networks*, Houston, Texas, pp.945–951.

Goldberg D.E. and Lingle R. (1985): *Alleles, loci, and the traveling salesman problem.* — Proc. Int. Conf. *Genetic Algorithms and Their Applications*, Carnegie Mellon University, Pitsbourgh, pp.154–159.

Hertz J., Krogh A. and Palmer R. (1993): *An Introduction to Neural Calculations Theory.* — Warsaw: WNT, (in Polish).

Holland J.H. (1975): *Adaptation in natural and artificial systems.* — University of Michigan Press, Ann Arbor.

Kaznachey D. and Jagota A. (1997): *Primal-target neural net heuristics for the hypergraph K-coloring problem.* — Proc. Int. Conf. *Neural Networks*, Houston, Texas, pp.965–971.

Korbicz J., Obuchowicz A. and Uciński D. (1994): *Artificial Neural Networks. Foundation and Applications.* — Warsaw: Academic Publishing House PLJ, (in Polish).

Lillo W.E., Hui S. and Żak S.H. (1991): *Neural networks for constrained optimization Problems.* — Int. J. Circ. Th. Appl., Vol.21, pp.385–399.

Reeves C.R. (1995): *Modern Heuristic Techniques for Combinatorial Problems.* — London: McGraw-Hill.

Tank D.W. and Hopfield J.J. (1986): *Simple "neural" optimization networks: An A/D converter, signal decision circuit, and linear programming circuit.* — IEEE Trans. Circ. Syst., Vol.CAS–33, pp.533–541.

Sun K.H. and Fu H.C. (1993): *A hybrid neural model for solving optimization problems.* — IEEE Trans. Comp., Vol.42, No.2, pp.219–227.

Tadeusiewicz R. (1993): *Neural Networks.* — Warsaw: Academic Publishing House PLJ, (in Polish).

Żurada J., Barski M. and Jędruch W. (1996): *Artificial Neural Networks.* — Warsaw: PWN, (in Polish).

Received: 25 October 1996
Revised: 21 April 1997
Re-revised: 4 July 1997