Transactions on the Built Environment vol 24, © 1997 WIT Press, www.witpress.com, ISSN 1743-3509

Neural-genetic techniques for ship control systems

Jerzy Balicki & Zygmunt Kitowski
The Polish Naval Academy
81-919 Gdynia, Smidowicza St., POLAND
e-mail: jbal@amw.gdynia.pl

Abstract

In this paper, artificial neural networks and genetic algorithms for solving optimization problems in ship control systems have been proposed. The gradient Hopfield ANN for linear minimization is considered. Moreover, HANN for finding local Pareto-optimal solutions in multicriteria optimization of designed navigation systems has been considered. Finally, genetic-neural algorithm GNA for improving a quality of solutions have been introduced.

1. Introduction

Operating and configurations of ship control systems are very complex. So, for estimating of their quality, we need several parameters. If there are considered the set of feasible alternatives of navigation systems, we have to solve the instance of multicriteria optimization problem [1]. Optimization techniques have many disadvantages related with small efficiency for even easy problems. But, neural networks and genetic algorithms are means for improving optimization tools. Although, optimization techniques with neural or genetic acceleration can be used in several fields, but, in this paper, we concern on methods, which are worth for ship control applications.

Users and designers consider several criteria related with a particular problem, for instance, the static deviation of control, the dynamic deviation of control, the safety of the motion, the fuel consumption, the deviation from the given trajectory and the others. These criteria evaluate the control quality and properties of finding solutions. Therefore, the user should express his system of preferences during the multi criteria problem formulation.

In this paper, artificial neural networks and genetic algorithms for solving optimization problems in ship control systems have been proposed. The gradient

Hopfield ANN for linear minimization with continuos decision variables is considered. Moreover, HANN for finding local Pareto-optimal solutions in multicriteria optimization with discrete decision variables has been considered. Finally, genetic-neural algorithm GNA for improving a quality of solutions have been introduced.

2. Neural networks for linear optimization with continuos decision variables

Let us consider a linear minimization problem, as follows:

$$\min_{x \in X} \sum_{m=1}^{M} c_m x_m \tag{1}$$

where

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

c_m are given for $m = \overline{1, M}$

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

a_m are given for $m = \overline{1, M}$.

In above optimization problem all decision variables are continuous, an objective function is linear $f(x) = \sum_{m=1}^{M} c_m x_m$, and constraints are linear, too. Of course, several optimization techniques can be used based on standard optimization methods. For instance, SIMPLEX method or parallel COMPLEX method can be used for solving it, but we want to improve Tank-Hopfield neural network model for solving it.

2.1. Hopfield neural networks

For some NP-hard combinatorial problems the Hopfield networks can find suboptimal solutions. There are many combinatorial problems that can be solved by Hopfield ANN. For instance, the Traveling Salesman Problem, the Hamiltonian Cycle Problem, graphs problems and minimization problems are solved with using Hopfield's ANN, the Boltzman Machine or the Genetic Methods [10,11]. That is why, we considered „neural" approach for optimization of navigation systems. But using Hopfield networks for solving continuous optimization problem is not so common used. So, we want to fill this gap.

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:

$$\frac{du_m}{dt} = -\frac{u_m}{\eta_m} + \sum_{n=1}^{M} w_{nm} g_n(u_n) + I_m \quad \text{for} \quad m = \overline{1,M}; \tag{2}$$

where

M - a neuron number,

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

η_m - a 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 is symmetric. Moreover, $w_{mm} = 0$ for $m = \overline{1,M}$. External inputs are constant during network operating. Signals in a neuron are transformed according to a linear activation function, as follows:

$$g_m(u_m) = u_m \qquad m = \overline{1,M} \tag{3}$$

Hopfield found a Liapunov function for the differential system (2) in respect to the following formula:

$$E(u) = -\frac{1}{2}\sum_{n=1}^{M}\sum_{m=1}^{M} w_{nm} g_n(u_n) g_m(u_m) - \sum_{m=1}^{M} I_m g_m(u_m) + \sum_{m=1}^{M} \int_0^{g_m(u_m)} g_m^{-1}(\xi_m) d\xi_m \tag{4}$$

where

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

2.2. Hopfield neural networks for linear optimization

Let us consider the constraint $\sum_{m=1}^{M} a_m x_m \geq b_n$ for given n. We take a denotation for a positive inequality resource $R_n = g_n(x) = \sum_{m=1}^{M} a_m x_m - b_n$. If $R_n \geq 0$, then inequality constraint is satisfied. In the other case $R_n < 0$, and inequality constraint is not satisfied. So, unfeasible solution should be punish during calculation. It can be made by the following penalty function:

$$P_n(a_n^t x - b_n) = \begin{cases} 0.5(a_n^t x - b_n)^2 & \text{for } a_n^t x - b_n < 0 \\ 0 & \text{for } a_n^t x - b_n \geq 0 \end{cases} \quad \text{for } n = \overline{1,N} \tag{5}$$

For Hopfield model of analog neural networks an energy function can be written, as follows:

$$E = c^t x + \sum_{n=1}^{N} P_n(a_n^t x - b_n) + \sum_{m-1}^{M} G_m \int_0^{x_m} g_m^{-1}(x_m) dx_m \tag{6}$$

where $G_m=1/\eta_m$.
From above energy function, the motion system can be obtained, as follows:

$$\frac{du_m}{dt} = -\frac{\partial E}{\partial x_m} \tag{7}$$

Very often on the left side of above formula the constant C_m is used, but it does not influence on the equilibrium point of motion equations. So, we get:

$$\frac{du_m}{dt} = -\frac{u_m}{\eta_m x} + \sum_{n=1}^{N}(-a_{nm})\frac{\partial P_n}{\partial R_n} - c_m \tag{8}$$

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 (5) the partial derivatives are given, as follows

$$\frac{\partial P_n}{\partial R_n} = \begin{cases} (a_n^t x - b_n) & for\ a_n^t x - b_n < 0 \\ 0 & for\ a_n^t x - b_n \geq 0 \end{cases} \quad for\ \ n = \overline{1,N} \tag{9}$$

Finally, the network for solving linear minimization problem can be presented on the fig. 1.

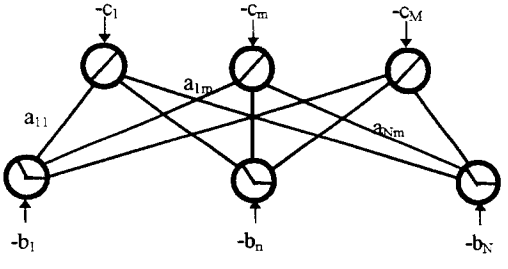

Fig. 1. Neural network for linear minimization.

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. The output values x_m from these neurons are sent to the second group of N neurons with a bit strange activation functions (9). These neurons are called constraint neurons and are responsible for constraint satisfaction. If associated constraint is not performed, then the constraint neuron generates positive value proportional to non satisfaction degree. The motion system can be solved by the linear Euler method, or the first order Runge-Kutty method.

3. Neural networks for optimization problem with zero-one variables

Algorithms for solving anti-collision situations are known. But, the problem is how to implement these algorithms in effective way. One of the promising solution is using two processor system to accelerate these calculations, because a time of decision making is the key for avoiding indicated collision. After solving

these problem, we know how to share program modules to two processors such, that cost of anti-collision system is the lowest, and the answer what to do for a captain was prepared almost at once. To solve above design problem we can use new optimization technique based on artificial methods, for instance neural networks or genetic algorithms.

Constraints in discrete optimization problems can be written in a general form $\sum_{m=1}^{M} x_m = L$, where x_m is a typical binary variables, and it should be satisfied M≥L. It means, that only L variables can be equal to 1, and the other variables M.-L should have value 0. There is no preferences which decision variables should be taken.

The general form $\sum_{m=1}^{M} x_m = L$ includes several constraints from a lot of combinatorial problems. For instance, in the Traveling Salesman Problem [12] 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 trip of the salesman has strict bounded time condition. He should make exactly L steps, and if the constraints $\sum_{k=1}^{L} x_{ik} = 1$ for $k = \overline{1,L}$ are satisfied, then that require is performed, too.

4. Uniform Hopfield neural networks HNN/L/M for linear constraint satisfaction

Uniform Hopfield networks UHNN play important role for satisfaction the special class of constraints expressed in a general form $\sum_{m=1}^{M} x_m = L$. For 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 $m = \overline{1,M}$.

In neural optimization the designed networks should avoid saddle points (false atractors).of energy function, because then even feasible solutions can not be obtained in some cases. On the fig. 2 the saddle point of a basic energy function is presented. Only two neurons are considered. Synaptic weights are equal to -2 and external inputs are equal to 1. For this case a basic energy function has two global minimizers and one saddle point. If UHNN starts form a initial state, then after relaxation it obtain an equilibrium point. If equilibrium point is a global minimizer of a basic energy function, then optimal solution of an

optimization problem can be found. But, if in an equilibrium point of motion equations a saddle point of a basic energy function is reached, then x1=x2=0.5 and formal constraints $x1,x2 \in \{0,1\}$ are not satisfied. So, the UHNN for optimization should avoid saddle points. In [4] special theorem for solving this problem has been stated.

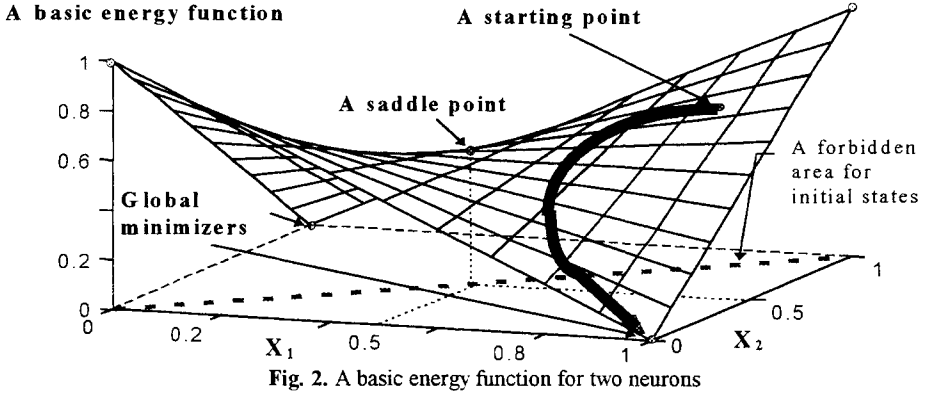

Fig. 2. A basic energy function for two neurons

So, if $u_1(t_0)=u_2(t_0)$, then $x_1(t_0)=x_2(t_0)$ in UHNN with two neurons (fig. 2). Because Hopfield networks minimize their energy functions according with the deepest descent method, then the trajectory of the state converges to a saddle point from each balanced starting point (c,c), where $c \in (0,1)$. On the fig. 2 the unfeasible area in zero-one optimization for initial states of UHANN is marked.

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 says about main parameters of UHNN such as thr neuron number, synaptic wages, and external inputs.

Theorem. 2 [4].

If $\sum_{m=1}^{M} x_m = L$ for $x_m \in \{0,1\}, L \le M, L = 0,1,2,\ldots,M$, and uniform Hopfield network has following parameters

$$w_{mj} = -2 \qquad m, j = \overline{1, M}, \qquad m \neq j$$
$$I_m = 2L - 1 \qquad m = \overline{1, M} \tag{10}$$

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 for their design the pair (L,M) have to be known, only. So, 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*=I in an equilibrium point.

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

We assume, that $\alpha=100$, $\Delta t/\tau=0.2$, $\eta=1$, k_{stop} is the condition $E(t_k) \le \varepsilon$ for $\varepsilon=0.01$. 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 numerical problems. Especially, for the network UHNN/L/M the neuron number M does not influence on the increasing of $K_{max}(E(t_k) \le \varepsilon)$.

5. Network HNN/F1/C for linear constrained minimization

A following optimization problem is studied:

$$\min_{x \in X} \sum_{i=1}^{2} \sum_{j=1}^{J} \kappa_j x_{ij}^{\pi} \tag{11}$$

For above problem two separated modified networks HANN/1/J for the satisfaction of the constraint $\sum_{j=1}^{J} x_{ij}^{\pi} = 1$, $i = \overline{1,2}$ can be used. 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 $\Delta I(x_{ij}^{\pi}) = \frac{\kappa_j}{\kappa_{max}}$, κ_{max} is the cost of the most expensive processor.

Above formula is related with the notice, 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 a 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.

6. Network HNN/F2/C for quasi-quadratic constrained minimization

If a more complex optimization problem is considered, when a time criterion $F_2(.)$ is minimized with respect to constraints $\sum_{i=1}^{2} x_{vi} = 1$, $v = \overline{1,V}$ and $\sum_{j=1}^{J} x_{ij}^{\pi} = 1$, $i = \overline{1,2}$. 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:

$$E(x\beta) = F_2(x) + \sum_{v=1}^{V} \beta_v E_v(x) + \sum_{i=V+1}^{V+2} \beta_1 \mathrm{E}_i(x) \qquad (12)$$

where

β_v, β_i- penalty coefficients

E_v- an energy function of a network UHNN/1/2 for a satisfaction of the constraint $\sum_{i=1}^{2} x_{vi} = 1$

E_i- an energy function of a network UHNN/1/J for a satisfaction of the constraint $\sum_{j=1}^{J} x_{ij}^{\pi} \leq 1$

7. Genetic algorithms for quality solution improving

Genetic algorithms can be used for solving several optimization problems. Holland [11] developed this approach and its theoretical foundation. Rosenberg noticed abilities of GA for development many criteria [15]. Then, Schaffer [16] considered GA for solving multiobjective optimization problems by an vector evaluated genetic algorithm VEGA. VEGA is an extension of system GENESIS prepared by Grefenstete [9]. VEGA uses ranging system for non-dominated individuals as Baker's ranging system for one function [3]. An overview of evolutionary algorithms for multiobjective optimization problems is presented by Fonseca and Flaming [7]. In recent years interest has risen in the application of genetic algorithms to combinatorial optimization problems [2,14].

It is possible to combine genetic algorithm and neural networks in several ways. First of all, neural networks prepared to optimization of a problem can be used as solvers for local optimization in each basic step of standard genetic algorithm. In a such approach, the genetic algorithm usually operates on the activation level initial values of optimization Hopfield network, because for given

synaptic weights and external inputs, the activation level initial values have a main influence on the obtained solutions. A gain coefficient in an activation function of neurons plays a less role. It should be large enough. Similarly, a passive suppress coefficient should be large enough. This standard genetic-neural algorithm SGNA assumes the parallel operating of neural networks. They exchange information to fit initial states in maximizing of fitness function. Fitness function is related with a global function created for scalarization of multiobjective problem. It is important to make sure, that this global function is prepared for minimizing of problems.

In [4] some experimental results are discussed. A standard genetic algorithm SGA and SNGA are compared. SGA uses 67 new populations to obtain two suboptimal in Pareto sense solutions for studied combinatorial optimization problem. Each population consists of 200 solutions. A probability of crossovering for solutions from a gene pool was taken p_c=0,95. A mutation probability for one bit was chosen p_m=0,001. A distribution of solutions in an initial population is uniform. Reached solutions are good, because they are feasible and close to Pareto-optimal solutions. Then, we used a proposed standard genetic-neural algorithm SGNA to improve obtained solutions.

Neural-genetic algorithm SGNA is very efficient for solving such optimization problem example. In the considered example of problem it found two Pareto-optimal solutions by generating only 12 new populations. Each population consisted of 50 artificial neural networks PHNN dedicated for solving considered optimization problem. After selection, crossover, and mutation of initial activation levels 50 dedicated neural networks obtain their equilibrium points. Afterwards, fitness function values are calculated and new population of initial activation levels.

8. Concluding remarks

The recurrent ANN in equilibrium point can represent suboptimal points. Hopfiel networks were simulated in the PC environment without neural accelerators. It is possible to use the neural accelerators and improve the performance of the neural calculating. Designed 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.

The main advantage of SGNA is improving solutions by neural networks. If neural networks are implemented as VLSI chips, then time of obtaining equilibrium points can be approximately equal to one computer instruction time. Even for simulation of networks by program environment based on computers IBM PC presented SGNA method is very powerful. Neural networks PHNN for finding Pareto-suboptimal solutions developed in the SGNA case is described in [4,5].

References

[1] A. Ameljańczyk, *"Multicriteria optimization"*, WAT, Warszawa 1986, (in polish).

[2] F. Q. Bac, V.L. Perov, *"New evutionery genetic algorithms for NP-complete combinatorial optimization problems"*, Biological Cybernetics, v. 69, 1993, pp.229-234.

[3] J.E. Baker: *"Adaptive selection methods for genetic algorithms"*, Proc. of an Int. Conf. on Genetic Algo-rithms and Their Applications, 1985, 101-111.

[4] J. Balicki, Z. Kitowski, *"Genetic-neural multiobjective optimization for resource allocation problems"*, Proc. of the Int. Conf. on Intelligent Technologies in Human-Related Sciences, Leon, Spain, 5-7 July 1996, pp. 18-22.

[5] J. Balicki, *"Artificial neural networks for multicriteria optimization problems of program modules allocations"*, in Proceedings of the 8th International Symposium on System-Modeling-Control, vol. 3, Zakopane, Poland, May 1-3, 1995, pp. 1-6.

[6] E.K.P. Chong, S.H. Zak, *An introduction to optimization.* John Wiley&Sons, Inc., New York, 1996.

[7] C.M. Fonseca, P.J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization", Evolutionary Computation, vol. 3, No. 1, 1995, pp.1-16.

[8] D.E. Goldberg, R. Lingle, *"Alleles, loci, and the traveling salesman problem"*, in Proceedings of the International Conference on Genetic Algorithms and Their Applications, Carnegie Mellon University, Pitsburg, 1985, pp. 154-159.

[9] J.J. Grefenstette: GENESIS: A system for using ge-netic search procedures. Proc. of an Int. Conf. on Genetic Algorithms and Their Applications, 1984, 161-165.

[10] J. Hertz, A. Krogh, R. Palmer, *Introduction to the The-ory of Neural Computation*, Massachusetts: Addison-Wesley Publishing Company, Inc. Reading, 1991.

[11] J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, 1975.

[12] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, *"Sequencing and scheduling. algorithms and complexity"*, Report BSR8909, Center for Mathematics and Computer Science, Amsterdam, 1989.

[13] W.E. Lillo, S. Hui, S.H. Zak, *"Neural networks for constrained optimization"*, Problems. Int. J. of Circuit Theory and Applications, vol. 21, pp. 385-399, 1991.

[14] Z. Michalewicz, *"Genetic Algorithms + Data Structures = Evolutionary Programs"*, Springer Verlag, 1992

[15] R.S. Rosenberg, *"Simulation of genetic populations with biochemical properties. I. The Model"*, Mathe-matical Biosciences, vol. 7, pp. 223-257.

[16] J.D. Schaffer, *"Multiple objective optimization with vec-tor evaluated genetic algorithm"*, Proc. of an Int. Conf. on Genetic Algorithms and Their Applications, 1985, 93-100.

[17] K.T. Sun, H.C. Fu, *"A hybrid neural model for solving optimization problems"*, IEEE Trans. on Computers, vol. 42, no. 2, pp. 219-227, February 1993.

[18] 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.