Turk J Elec Engin, VOL.8, NO.1 2000, © TÜBİTAK

Comparison of Simulation Algorithms for the Hopfield Neural Network: An Application of Economic Dispatch

Tankut Yalçınöz and Halis Altun
Department of Electrical and Electronic Engineering,
Niğde University, 51100 Niğde-TURKEY

Abstract

This paper is mainly concerned with an investigation of the suitability of Hopfield neural network structures in solving the power economic dispatch problem. For Hopfield neural network applications to this problem three important questions have been answered: what the size of the power system is; how efficient the computational method; and how to handle constraints. A new mapping process is formulated and a computational method for obtaining the weights and biases is described. A few simulation algorithms used to solve the dynamic equation of the Hopfield neural network are discussed. The results are compared with those of a classical technique, Hopfield neural network approaches and an improved Hopfield neural network approach [1].

1. Introduction

Hopfield and Tank [2,3] presented the energy function approach in order to solve several optimization problems including the travelling salesman problem (TSP), analog to digital conversion, a signal processing problem and linear programming problems. Their results encouraged a number of researchers to apply this network to different problems such as object recognition, graph recognition and economic dispatch problems.

However, the Hopfield neural network, which will be called the original Hopfield neural network, can be unstable. This algorithm does not scale well to large problems either in solution quality or in reliable convergence to a feasible solution [4]. Other criticisms are as follows:

- The network solution always has a local minimum which depends on the initial conditions.
- The robustness of the algorithm cannot be relied on, and therefore the results may be unfeasible.
- It is necessary to develop an efficient mapping method to determine weights in the energy function.

However, later researchers improved the energy function and modified the algorithm to produce reliable feasible solutions, and various methods were developed to escape from local minima. Van den Bout and Miller [5] improved the performance of the original Hopfield neural networks using the mean field annealing algorithm with neural normalization. Abe [6] explained the convergence region of a local minimum in the Hopfield neural network. He determined weights in the energy function for the TSP such that the feasible solution becomes stable.

In addition, some researchers have developed better problem mapping to improve the overall quality of the solutions. Aiyer et al. [7] developed a mapping technique which identified the valid constraints in the energy function using a single constraint term. In the early mapping techniques, in contrast, the energy function has a constraint term for each constraint, and Aiyer [8] successfully solved the TSP. Gee, Aiyer and Prager [9] discussed a new methodology to improve the performance of Hopfield networks, by formalizing the mapping process and providing a computational method for obtaining the weights and biases. Gee and Prager [10] improved their mapping to solve quadratic 0-1 programming problems with linear equality and inequality constraints. They applied this mapping technique to a few specific problems, such as the travelling salesman, knapsack and the Hamilton path problems. This method is quicker and more accurate and thus more efficient than the original Hopfield neural network method [2].

The economic dispatch problem in a power system is to determine the optimal combination of power outputs for all generating units which will minimize the total fuel cost while satisfying load and operational constraints. Neural networks have been used for solving the economic dispatch problem and have been usually applied to small power systems [11-16]. The Hopfield neural network was first applied successfully to the economic dispatch problem by Park et al. [12]. The authors used a simple 3-unit system taken from reference [17]. Then, some researchers used the Hopfield NN for solving the economic-environmental dispatch problem [15] and the security-constrained optimal rescheduling problem [18]. References [12] and [15] showed that the Hopfield NN converges very slowly. Recently, some researchers proposed other approaches to solving this problem. Some kind of adaptive scheme has been used to update the slope or bias of the network to speed up the convergence of the Hopfield NN for the economic dispatch problem [16], using piecewise quadratic cost functions. The results for a 10-unit test system are compared with those of a numerical approach and the traditional Hopfield NN approach [12]. The number of iterations is reduced from about 40000 to less than 100. The slope adjustment and the bias adjustment methods take about 2 and 4 s respectively on a Compaq 90 MHz Pentium PC. An improved Hopfield NN has been proposed for solution of large-scale economic dispatch problems by Yalcinoz and Short [1]. The proposed method has achieved efficient and accurate solutions for different sizes of systems having 3 and 240 units, taking only 1.7 s on the Pentium 75 MHz PC for a 20-unit test system for the ED problem.

In this paper, a method is proposed using a new mapping technique, which has been described in [8,10], for Hopfield neural networks to solve the quadratic programming problems, subject to a number of inequality and equality constraints, which can be handled by adding corresponding terms to the energy function. For the mapping of quadratic programming problems, Aiyer's approach [8] has been combined with a slack variable technique for inequality constraints [10,19]. An efficient simulation algorithm has been used to solve the dynamic equation of the Hopfield NN in which the time step has been calculated. A few simulation algorithms for Hopfield neural networks are discussed. These approaches are applied to the economic dispatch problem, which is a large scale nonlinear allocation problem with both equality and inequality constraints.

2. Mapping Technique for Quadratic Programming Problems

The Hopfield model [20] has used a continuous nonlinear function to describe the output behaviour of the neurons, with the following procedure. First, construct an energy function by taking a linear combination of the objective and penalty functions. Then build a fully connected n vector of neurons and set the weights so that the network will take a strictly descending path on the energy function. Initialize the network to the starting point (that point which is the middle of all feasible solutions) and let the network run until it stabilizes at a local minimum. The lowest energy state corresponds to the optimum solution, and values of the network outputs are taken as the solution.

If neuron i has input u$_i$ and output x$_i$, and is connected to neuron j with a weight T$_{ij}$, and each neuron also has an input bias i$_i^b$, then the dynamic equations of the network are defined as follows:

$$\frac{du_i}{dt} = -\frac{u_i}{\eta_i} + \sum_{j=1}^{n} T_{ij} x_j + i_i^b \tag{1}$$

$$x_i = g(u_i)$$

where η_i is a passive decay term, n is the number of neurons and g(u$_i$) is the input-output function (often called the activation function) of neuron i, usually shown as

$$g(u_i) = \frac{1}{2}(1 + \tanh(u_i/\beta))$$

where β is a coefficient that determines the shape of the input-output function. Hopfield [20] discovered a Lyapunov function for a network of n neurons as identified by equation (1). He called this Lyapunov function the network's energy function and defined it as

$$E(x) = -\frac{1}{2} x^T T x - x^T i^b$$

The energy function, which is a quadratic function, is associated with the objective function for minimizing the optimization problem. Therefore, we must first decide how to set weights **T** and input biases $\mathbf{i}^b$ for any minimization problem. This process is called mapping. The sum of the constraints and an objective function are given as inputs to the energy function. The energy function comes from a similarity between the network's behaviour and the behaviour of the physical system. A physical system, such as a pendulum, may converge towards an energy minimum whereas a network of neurons always moves towards a minimum of the energy function.

In this paper, we combine an improved Aiyer's mapping technique, which has been described for quadratic 0-1 programming problems with linear equality and inequality constraints [8,10], with Abe's formulation [19] for inequality constraints. A similar mapping technique was proposed for the economic dispatch by Yalcinoz and Short [1].

The quadratic problem has equality and inequality constraints and can be written as

$$\text{minimize } E^{obj}(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T \mathbf{T}^{obj} \mathbf{x} - \mathbf{x}^T \mathbf{i}^{obj} \tag{2}$$

subject to:

$$\mathbf{A}^{eq}\mathbf{x} = \mathbf{b}^{eq} \tag{3}$$

$$\mathbf{A}^{in} x \leq \mathbf{b}^{in} \tag{4}$$

or

$$\mathbf{A}^{in} x \geq b^{in} \tag{5}$$

$$\mathbf{x}_{\min,i} \leq \mathbf{x}_i \leq \mathbf{x}_{\max,i} \qquad i = 1, \ldots, n \tag{6}$$

where $\mathbf{x}$ is the n dimensional variable vector, $\mathbf{T^{obj}}$ is an nxn symmetrical constant matrix, $\mathbf{i^{obj}}$ is the n dimensional constant vector, and $\mathbf{x}_{min,i}$ and $\mathbf{x}_{max,i}$ are the lower and upper bounds respectively.

Gee and Prager [10] have first considered the simple quadratic problem without inequality constraints. The feasible solution for equality constraints can be described as

$$\mathbf{x} = \mathbf{T^{constr}x} + \mathbf{s} \tag{7}$$

where

$$\mathbf{T^{constr}} = \mathbf{I} - \mathbf{A^{eq}}^{\mathbf{T}}(\mathbf{A^{eq}A^{eq}}^{\mathbf{T}})^{-1}\mathbf{A^{eq}} \tag{8}$$

and

$$\mathbf{s} = \mathbf{A^{eq}}^{\mathbf{T}}(\mathbf{A^{eq}A^{eq}}^{\mathbf{T}})^{-1}\mathbf{b^{eq}} \tag{9}$$

Equation (7) defines a valid subspace on which x must lie if it is to satisfy the equality constraints. For this case, the energy function can be written as

$$E = E^{obj} + \frac{1}{2}c_o||\mathbf{x} - (\mathbf{T^{constr}x} + \mathbf{s})||^2 \tag{10}$$

By setting c_o to a large enough value, we can make the $c_o||x - (T^{constr}x + s)||^2$ term dominant over E^{obj}. Thus to minimize E, x remains in the valid subspace. The equality constraints have been combined into a single penalty term in the energy function. Let us put the E^{obj} (2) into energy function (10), as follows:

$$E = -\frac{1}{2}\mathbf{x}^T[\mathbf{T}^{obj} + c_o(\mathbf{T^{constr}} - \mathbf{I})]\mathbf{x} - \mathbf{x^T}(\mathbf{i^{obj}} + c_o\mathbf{s}) + \frac{1}{2}c_o\mathbf{s}^T\mathbf{s} \tag{11}$$

The network's weights and input biases are set for satisfying energy function (11) as

$$\mathbf{T} = \mathbf{T}^{obj} + c_o(\mathbf{T}^{\text{constr}} - \mathbf{I}) \tag{12}$$

$$\mathbf{i}^b = \mathbf{i}^{obj} + c_o\mathbf{s} \tag{13}$$

The mapping technique can be extended to include the inequality constraints which are converted to equality constraints by introducing slack variables [10,19]. The slack variable technique as presented by Abe et al. [19] was applied to inequality constraints where equation (4) can be converted for the i-th inequality constraint as follows using a slack variable $\mathbf{y}_i$:

$$\mathbf{b}_i^{\mathbf{in}}\mathbf{y}_i - \mathbf{A}_i^{\mathbf{in}}\mathbf{x}_i = 0 \quad \text{where } \mathbf{y}_i \leq 1 \tag{14}$$

Equation 5 can be rewritten as

$$\mathbf{b}_i^{\mathbf{in}}\mathbf{y}_i - \mathbf{A}_i^{\mathbf{in}}\mathbf{x}_i = 0 \quad \text{where } \mathbf{y}_i \geq 1 \tag{15}$$

After converting inequality constraints to equality constraints we can extend the mapping technique for the inequality constraints. For this case, variables $\mathbf{x}$ are set as $\mathbf{x}^{new^T} = [\mathbf{x^T y^T}]$, where $\mathbf{y}$ is the vector of slack variables $[\mathbf{y_1 y_2} \ldots \mathbf{y}_{\text{min}}]^T$: therefore, the size of $\mathbf{x^{new}}$ is $\mathbf{n}^{new} = \text{n} + \text{m}^{in}$ where n and m^{in} are the number of variables and inequality constraints respectively. The optimization problem may be stated as

$$\text{minimize } E^{obj}(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{\mathbf{new}^{\mathbf{T}}}\mathbf{T}^{\mathbf{new}^{\mathbf{obj}}}\mathbf{x}^{\mathbf{new}} - \mathbf{x}^{\mathbf{new}^{\mathbf{T}}}\mathbf{i}^{\mathbf{new}^{\mathbf{obj}}} \tag{16}$$

subject to

$$\mathbf{A}^{new}\mathbf{x}^{\mathbf{new}} = \mathbf{b}^{new} \tag{17}$$

where $\mathbf{b}^{\mathbf{new}}$ is a vector $(m^{eq}+m^{in})$, $\mathbf{T}^{\mathbf{new}^{\mathbf{obj}}}$ is equal to $\begin{bmatrix} \mathbf{T}^{obj} & 0 \\ 0 & 0 \end{bmatrix}$ and $\mathbf{i}^{\mathbf{new}^{\mathbf{obj}}}$ is equal to $\begin{bmatrix} \mathbf{i}^{obj} \\ 0 \end{bmatrix}$. The equality constraints (17) consist of both original equality (3) and inequality constraints (4 and 5). We can set the network's parameters as equations (12) and (13). Hence the Hopfield NN is created with n neurons for variables and m^{in} neurons for slack variables.

2.1. Side Constraints

The variables may be restricted to be within certain limits. The generator outputs which are variables for solving the economic dispatch problem should be between the lower and upper generation limits. Therefore, side constraints should be taken into account. Aiyer [8] and Ogier and Beyer [21] studied steepest descent dynamics and later, Gee and Prager [10] proposed steepest descent dynamics for quadratic 0-1 programming problems. The variables are limited to the range from zero to one. Similar dynamics can be used for solving the economic dispatch problem in order to handle generation limits.

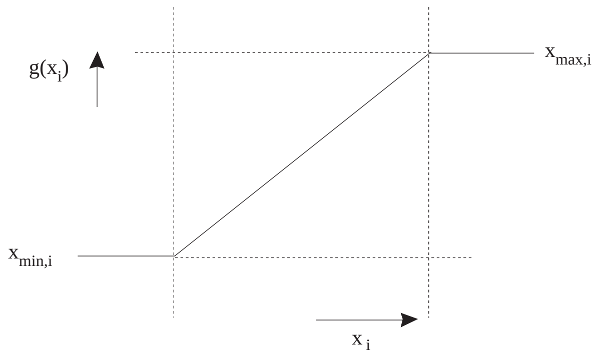

Figure 1. Input output function of the variable i

A symmetric ramp function is chosen for the input-output function and is applied to each element of **x**. Figure 1 shows the activation function of the variable (neuron) i, which is described as

$$\mathbf{x}_I = g(\mathbf{x}_i) = \begin{cases} \mathbf{x}_{\min,i} & \text{if } \mathbf{x}_{\min,i} > \mathbf{x}_i \\ \mathbf{x}_i & \text{if } \mathbf{x}_{\min,i} \le \mathbf{x}_i \le \mathbf{x}_{\max,i} \\ \mathbf{x}_{\max,i} & \text{if } \mathbf{x}_i > \mathbf{x}_{\mathrm{m}ax,i} \end{cases}$$

The activation function of each neuron is modified to limit the output value between lower and upper bounds.

3. An Efficient Simulation Algorithm

The differential equation (1) of the Hopfield NN can be solved using numerical techniques for initial value problems if each processing element is given an initial value, u$_i$(0). Here, a few simulation algorithms are discussed.

An Euler approximation of the continuous differential equation (Eq. 1) has been used for computer simulations of the Hopfield NN [2] but it requires a very small time step Δt at each iteration. If the time step is increased too much it is possible for the algorithm to become unstable. The continuous Hopfield NN is shown in Figure 2 where the change of u is given by

$$\Delta\mathbf{u} = \Delta t(-\frac{\mathbf{u}}{\eta} + \mathbf{Tx} + \mathbf{i^b})$$

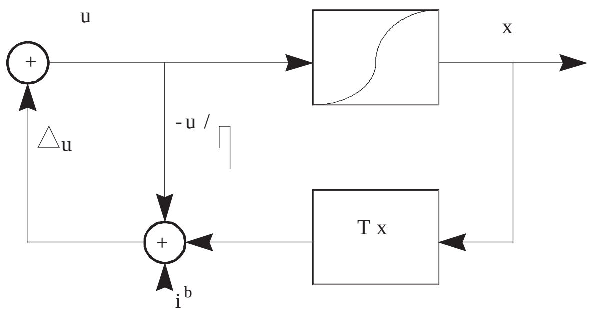

Figure 2. Simulation of the continuous Hopfield NN

Aiyer [8, 9] proposed the projection network implementation in order to overcome the continuous Hopfield network's problems. This modified network requires fewer iterations than the continuous Hopfield network and has one parameter Δt. In contrast, the Hopfield network [2] has five parameters (A, B, C, D and Δt) for the travelling salesman problem. The projection network implementation is shown in Figure 3. Firstly, in box 2, **x** is updated over a finite time step Δt using the objective term E^{obj} alone. Then, in box 1, updated **x** is directly forced to the valid region.

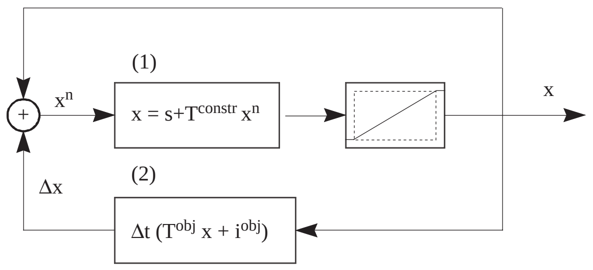

Figure 3. Aiyer's simulation algorithm

Here we modify Aiyer's simulation algorithm as shown in Figure 4, based on the projected gradient technique. Each iteration has two separate stages. Firstly, in box 1, **x** is updated over a finite time step Δt using the objective term E^{obj} alone where the time step is usually determined empirically by running some trial simulations. In this study, we calculate the time step instead of doing a number of trials in order to find the time step. Time step Δt depends on the objective function and constraints. In the next section, time step Δt will be mathematically derived. Then, in box 2, updated x is directly forced into the valid region. The slack variables $\mathbf{y}_i$ are forced into the valid region according to equations (14) and (15). For example, slack variables $\mathbf{y}_i$ are equal to or less than 1 for inequality constraints (4). In the convergence box in Fig.

4, the difference between the values of the cost function calculated by the present iteration and the previous iteration is checked. If the difference is smaller than the tolerance selected, the calculation will be stopped; otherwise the new cost value will be calculated. This is an iterative process.

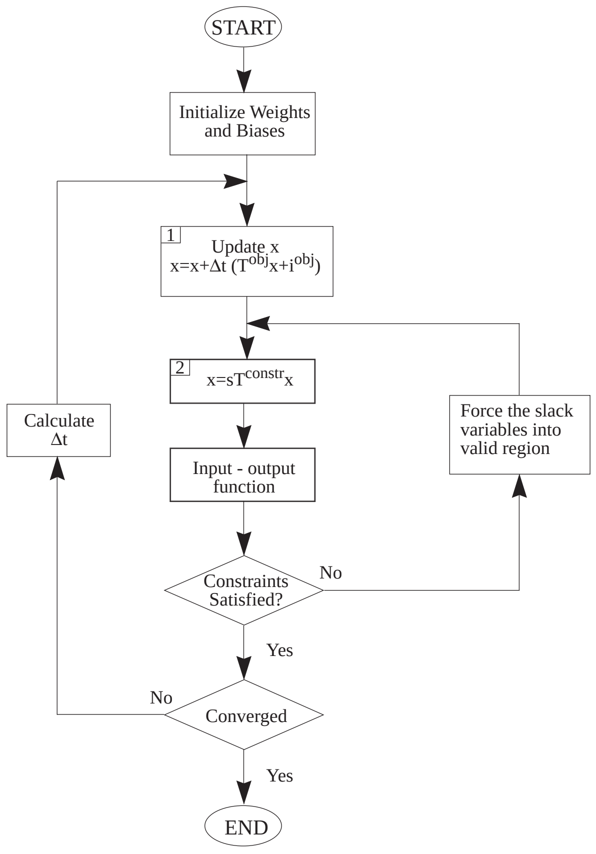

Figure 4. Simulation algorithm

3.1. Calculation for the Time Step

The important question is, how big a time step should we take? From basic calculus, time step Δt minimizes energy function E when the directional derivative $\frac{d}{d(\Delta t)}E(\mathbf{x}_{(1)})$ is equal to zero. By the chain rule,

$$\frac{d}{d(\Delta t)}E'(\mathbf{x}_{(1)}) = E'(\mathbf{x}_{(1)})^T \frac{d}{d(\Delta t)}\mathbf{x}_{(1)} = E'(\mathbf{x}_{(1)})^T \mathbf{r}_{(0)} \tag{18}$$

where $\mathbf{r}$ is a residual. Let us start from the energy function (11)

$$E = -\frac{1}{2}\mathrm{x}^T[\mathbf{T}^{obj} + c_o(\mathbf{T}^{constr} - \mathbf{I})]\mathbf{x} - \mathbf{x}^T(\mathbf{i}^{obj} + c_o\mathbf{s}) + \frac{1}{2}c_o\mathbf{s}^T\mathbf{s}$$

If the network's weights and input biases are set as equations (12) and (13), then we can rewrite the energy function as

$$E = -\frac{1}{2}\mathbf{x}^T\mathbf{T}\mathbf{x} - \mathbf{x}^T\mathbf{i}^b + \frac{1}{2}c_o\mathbf{s}^T\mathbf{s}$$

If the directional derivative $\frac{d}{d(\Delta t)}E(\mathbf{x}_{(1)})$ is equal to zero, we can calculate the residual as

$$\mathbf{r}_{(i)} = -E'\mathbf{x}_{(i)}) = \mathbf{T}\mathbf{x}_{(i)} + \mathbf{i}^b \tag{19}$$

Rewriting equation (18) using (19) in order to determine the time step Δt,

$$\frac{d}{d(\Delta t)}E(\mathbf{x}_{(1)}) = E'(\mathbf{x}_{(1)})^T\mathbf{r}_{(0)} = -\mathbf{r}_{(1)}^T\mathbf{r}_{(0)} = 0$$

and $(\mathbf{T}\mathbf{x}_{(1)} + \mathbf{i}^b)^T\mathbf{r}_{(0)} = 0$, where $\mathbf{x}_{(1)}$ can be calculated as $\mathbf{x}_{(1)} = \mathbf{x}_{(0)} + \Delta t\mathbf{r}_{(0)}$. Then we have

$$\begin{aligned}&(\mathbf{T}(\mathbf{x}_{(0)} + \Delta t\mathbf{r}_{(0)}) + \mathbf{i}^b)^T\mathbf{r}_{(0)} = ((\mathbf{T}\mathbf{x}_{(0)} + \mathbf{i}^b) + \boldsymbol{\Delta}\mathbf{t}\mathbf{T}\mathbf{r}_{(\mathbf{0})})^T\mathbf{r}_{(0)} = 0 \\ &(\mathbf{r}_{(0)} + \boldsymbol{\Delta}\mathbf{t}\mathbf{T}\mathbf{r}_{(\mathbf{0})})^T\mathbf{r}_{(0)} = 0\end{aligned} \tag{20}$$

According to equation (20), the time step Δt can be calculated as

$$\Delta t = -\frac{\mathbf{r}_{(0)}^T\mathbf{r}_{(0)}}{\mathbf{r}_{(0)}^T\mathbf{T}^T\mathbf{r}_{(0)}} \tag{21}$$

Δt may be interpreted as the time step or, more precisely, the convergence rate.

4. Formulation of Economic Dispatch Problem

The standard economic dispatch minimizes the total thermal unit operating cost which can be described mathematically as

$$\min_{p_i}\sum_{i=1}^{n} F_i(P_i) = \min_{p_i}\sum_{i=1}^{n}(a_i + b_i\mathbf{P}_i + c_i\mathbf{P}_i^2) \tag{22}$$

subject to the following constraints:
Load demand:

$$\sum_{i=1}^{n}\mathbf{P}_i - \mathbf{P}^D - \mathbf{P}^L = 0 \tag{23}$$

$$\mathbf{P}^L = \sum_{i=1}^{n}\mathbf{B}_i\mathbf{P}_i^2 \tag{24}$$

Capacity limits of generators:

$$\mathbf{P}_{\min,i} \leq \mathbf{P}_i \leq \mathbf{P}_{\max,i} \tag{25}$$

where
P_i : Power output of the i-th generator
a_i, b_i, c_i : Cost coefficients of the i-th generator
n : Number of units

$F_i(P_i)$: Cost of generation P_i
P^D : Demand
P^L : Transmission losses
B : Coefficients of transmission losses
$P_{\text{min,i}}$: Minimum generation output of the i-th generator
$P_{\text{max,i}}$: Maximum generation output of the i-th generator

5. Mapping of Economic Dispatch Problem

We can set weights and input biases for the cost of electricity using the mapping technique which was described in Section 2. The Hopfield NN is created with n neurons for generators and m^{in} neurons for inequality constraints. The cost function of the economic dispatch problem (22) is considered the energy function of the Hopfield NN. Therefore the cost function is mapped as

$$\mathbf{T}^{\text{obj}} = -2 \cdot \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ . & . & . & . \\ 0 & 0 & 0 & c_n \end{bmatrix}$$

$$\mathbf{i}^{\text{obj}} = -[b_1 b_2 .. b_n]$$

where b_i and c_i are the cost coefficients of the i-th generator. The constraints of the economic dispatch problem can be handled by adding corresponding terms to the energy function. We can convert the inequality constraints to equality constraints using equations (14) and (15), and then $A^{\mathbf{new}}$ and $b^{\mathbf{new}}$ can be written as

$$A^{\mathbf{new}} = \begin{bmatrix} A^{\mathbf{eq}} \\ A^{\mathbf{in}} \end{bmatrix} \; and \; b^{\mathbf{new}} = \begin{bmatrix} b^{\mathbf{eq}} \\ b^{\mathbf{in}} \end{bmatrix}$$

where $A^{\mathbf{eq}}$ and $b^{\mathbf{eq}}$ are defined from equation (23) for the economic dispatch problem as follows:

$A^{\mathbf{eq}}$ is a vector which contains ones for units and zeros for slack variables and $\mathbf{b}^{\mathbf{eq}} = P^D + P^L$. $\mathbf{A}^{\mathbf{in}}$ and $\mathbf{b}^{in}$ are defined from inequality constraint equations. Inequality constraints are converted to equality constraints by introducing slack variables.

Yalcinoz and Short [1] have treated side constraints (generation limits) of the ED problem as inequality constraints. Therefore, a neuron for each generator and two neurons for generation limits of each generator have been used for solution of the economic dispatch problem.

Here, side constraints are handled by a modified activation function as explained in Section 2.1. The activation function of each neuron is changed to limit the output value of each neuron between the minimum and maximum generating capacity of each unit as shown in Figure 5. The activation function of neuron i (generator i) is given as

$$\mathbf{P}_i = g(\mathbf{P}_i) = \begin{cases} \mathbf{P}_{\min i} & \text{if} \quad \mathbf{P}_{\min\, i} > \mathbf{P}_i \\ \mathbf{P}_i & \text{if} \quad \mathbf{P}_{\min,\, i} \le \mathbf{P}_i \le \mathbf{P}_{\max i} \\ \mathbf{P}_{\max i} & \text{if} \quad \mathbf{P}_i > \mathbf{P}_{\max\, i} \end{cases}$$

After finding $\mathbf{A}^{\text{new}}$ and $\mathbf{b}^{\text{new}}$, $\mathbf{T}^{\text{constr}}$ and $\mathbf{s}$ can be determined using equations (8) and (9). Afterwards we can set new weights and new input biases using (12) and (13), and then the Hopfield NN is created for solving the economic dispatch problem. At this stage, we have to solve the dynamic equation of the Hopfield network using numerical techniques.

In this study, the dynamic equation of the network can be solved using the algorithms described in Section 3. The first method is an Euler approximation of the continuous differential equation (1) which has been used for solving the dynamic equation by Hopfield and Tank [2]. Figure 2 shows the continuous Hopfield NN, which will be called standard Hopfield NN (SHN). The second method is projection network implementation, which has been proposed in order to overcome the continuous Hopfield network's problems [8,9]. The projection network implementation is shown in Figure 3 and will be called Aiyer's improved Hopfield NN (AHN). Finally, the slightly modified Aiyer's simulation algorithm shown in Figure 4 is used for computer simulations. Side constraints of the ED problem are handled using modified activation functions as described in Section 2.1 for solving the dynamic equations. This process will be called the proposed Hopfield neural network algorithm (PHN).

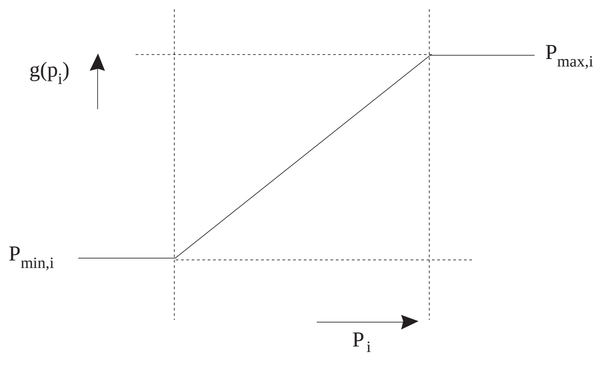

Figure 5. The input-output function of the generator i

In this study, transmission losses are calculated using equation (24) at the end of every period, and then assumed to be constant over the next period.

6. Simulation Results of Economic Dispatch

The proposed method is tested on 3- and 20- units systems and is compared with the classical optimization technique (CM), which is based on the Sequential Quadratic Programming method, standard Hopfield NN (SHN), Aiyer's improved Hopfield NN (AHN) and an improved Hopfield NN approach IHN [1]. The classical optimization program was written using the Matlab Optimization Toolbox, and the other programs which implemented PHN, IHN, SHN and AHN were also written in Matlab. The programs were executed on a Pentium 75 MHz PC with 8 Mb RAM.

Table 1. Data for 3-Generator system

	Unit ♯1	Unit ♯2	Unit ♯3
P_{max} / P_{min} (MW)	600 / 150	400 / 100	200 / 50
Cost coefficients a_i / b_i / c_i	561 / 7.92 / 0.00156	310 / 7.85/ 0.00194	78 / 7.97 / 0.00482

The first test system, which has three units - see reference [17], - is chosen because this example has been a popular problem used in the literature [11,12,14,15]. Details of this test system are given in Table 1. Transmission losses are calculated using (24). The transmission loss coefficients B are given by

$$B = [0.000030\ 000090\ 00012]$$

In Table 2, the results of the PHN method are compared with the results of the classical method (CM), SHN, AHN and IHN [1] for 3 load levels. It is seen that there is negligible difference in the values between the proposed method and the CM. The error is calculated as the percentage difference between the values of the cost functions of the neural networks methods and the classical method. We can formulate the errors or differences as

$$\text{Err} = \frac{\text{NN Method's Cost - CM's Cost}}{\text{CM's cost}} \cdot 100\%$$

The errors of SHN and AHN are 0.132% and 0.023% respectively but their execution times are slower than CM. For example, SHN is about 15 times slower than CM. On the other hand, the proposed method is about 2-9 times faster than CM. The errors of IHN and PHN are around zero.

Table 2. Results of CM, SHN, AHN, IHN and PHN for 3-unit system

Methods	P^D+P^L	Output of generators (MW)			Cost	Error	CPU	Iteration
	(MW)	P_1	P_2	P_3	($/h)	%	time (s)	no.
CM	342.762	152.18	140.57	50.00	3742.9	—	0.44	—
SHN	342.754	170.35	104.18	68.211	3748.5	0.15	12.91	7650
AHN	342.762	159.64	133.02	50.092	3743.1	0.0053	0.99	500
IHN	342.762	152.52	139.85	50.381	3742.9	0	0.16	11
PHN	342.77	152.23	140.54	50.00	3743.0	0.0026	0.38	11

Methods	P^D+P^L	Output of generators (MW)			Cost	Error	CPU	Iteration
	(MW)	P_1	P_2	P_3	($/h)	%	time (s)	no.
CM	867.14	401.22	341.08	124.84	8351.4	—	0.93	—
SHN	867.12	373.73	310.27	183.12	8370.6	0.23	16.75	10000
AHN	867.14	383.79	331.98	151.362	8355.4	0.0479	0.99	500
IHN	867.14	401.67	340.66	124.81	8351.4	0	0.11	9
PHN	867.14	401.66	340.66	124.82	8351.4	0	0.05	11

Methods	P^D+P^L	Output of generators (MW)			Cost	Error	CPU	Iteration
	(MW)	P_1	P_2	P_3	($/h)	%	time (s)	no.
CM	1179.1	592.33	400.00	186.77	11295	—	0.66	—
SHN	1179.07	583.55	397.93	197.58	11297	0.0177	16.86	10000
AHN	1179.1	582.96	398.77	197.36	11297	0.0177	0.99	500
IHN	1179.1	592.52	399.57	187.01	11296	0.0089	0.33	7
PHN	1179.12	591.33	400.00	187.79	11296	0.0089	0.33	11

The 20-unit test system is given in Appendix 1. The results of the proposed method are shown in Table 3 and compared against the results of CM; SHN, AHN and IHN for the 20-unit system with a 5500 MW load demand. The operation cost obtained by SHN is 16.4% more than CM; in contrast, the operation costs obtained by AHN and IHN are 0.5% and 0.19% more than the CM respectively. The IHN method is approximately 4-32 times faster than the CM, the SHN and AHN, but about 3 times slower than PHN.

PHN provides better and faster solutions than the other neural network methods. In addition, PHN is 28 times faster than CM, with 0.068% error. Oscillation is also extremely reduced from 10000 for the SHN to less than 20 iterations for the proposed method. In contrast, the slope adjustment and the bias adjustment methods [16] require about 100 iterations and takes about 2 and 4 s respectively on a Compaq 90 MHz Pentium PC for a 10-unit test system. PHN takes only 0.49 sec. on a Pentium 75 MHz PC for a 20-unit test system.

Table 3. Results of CM, SHN, AHN, IHN and PHN for 20-unit system

Methods	Cost ($/h)	Error %	CPU time (s)	Iteration no.
CM	78636	—	13.9	—
SHN	91531	16.4	45.37	10000
AHN	79031	0.5	6.31	1000
IHN	78785	0.19	1.7	10
PHN	78690	0.068	0.49	10

PHN provides significant improvements in performance over IHN, the main reason for this improvement being that PHN requires fewer neurons than IHN. Therefore sizes of the weights' matrix and the input biases' vector of PHN are smaller than those of IHN. Remembering that we use n neurons for generators and m^{in} neurons for inequality constraints, in the IHN model, side constraints of the ED problem are treated as inequality constraints. Hence, n neurons for generators and two neurons for side constraints of each generator are used for the solution of the ED problem.

7. Conclusions

In this paper, for quadratic programming problems with inequality and equality constraints, an improved Hopfield neural network has been outlined. Gee and Prager [10] discussed a new mapping technique for quadratic 0-1 programming problems with linear equality and inequality constraints and this special methodology improved the performance of Hopfield neural networks for solving combinatorial optimization problems. A similar mapping technique has been proposed for the economic dispatch problem. Constraints can be handled by adding corresponding terms to the energy function. For the mapping of the economic dispatch problem, Aiyer's approach [8] has been combined with a slack variable technique for inequality constraints [10,19]. We used Abe's slack variable technique [19] for inequality constraints. Simulation algorithms have been examined and a time efficient simulation algorithm has been presented. The time step of this algorithm has been calculated instead of undertaking a number of trials. Finally, a symmetric ramp function has been chosen for the input-output function and has been applied to each of the elements of x for handling side constraints.

The proposed method has been tested on 3-unit and 20-unit systems. The errors of the proposed method for solving the economic dispatch problem are around zero for the 3-unit system. Kumar and Sheble [14], in contrast, reported an approximately 1% error for 3-unit systems. The average errors incurred here are less than 0.1% even for the 20-unit system. The proposed method requires fewer iterations and achieves very fast solutions for the economic dispatch problem.

References

[1] T. Yalcinoz and M.J. Short, "Large-scale economic dispatch using an improved Hopfield neural network", IEE Proc. Gener. Transm. Distrib., Vol. 144, No. 2, pp. 181-185, March 1997.

[2] J.J. Hopfield and D.W. Tank, "Neural computations of decisions in optimization problems", Biological Cybernetics, Vol. 52, No. 3, pp. 141-152, 1985.

[3] J.J. Hopfield and D.W. Tank, "Simple neural optimization networks: An A/D converter, signal decision network, and a linear programming circuit", IEEE Trans. on Circuit and Systems, CAS-33, pp. 533-541, May 1986.

[4] G.V. Wilson and G.S. Pawley, "On the stability of the Traveling Salesman Problem algorithm of Hopfield and Tank" Biological Cybernetics, Vol. 58, pp. 63-70, 1988.

[5] D.E. Van den Bout and T.K Miller III, "Improving the performance of the hopfield-tank neural network through normalization and annealing", Biological Cybernetics, Vol. 62, pp. 129-139, 1989.

[6] S. Abe, "Global convergence and suppression of spurious states of the Hopfield neural networks", IEEE Trans. on Circuit and Systems-1: Fundamental theory and Applications, Vol. 40, No. 4, pp. 246-257, 1993.

[7] S.V.B. Aiyer, M Niranjan and F. Fallside, "A theoretical investigation into the performance of the Hopfield model", IEEE Trans. on Neural Networks, Vol. 1, pp. 204-215, 1990.

[8] A.H. Gee and R.W. Prager, "Polyhedral combinatorics and neural networks", Neural Computation, Vol. 6, pp. 161-180, 1994.

[9] C.Y. Maa and M.A. Shanblatt, "Linear and quadratic programming neural network analysis", IEEE Trans. on Neural Networks, Vol. 3 No. 4, pp. 580-594, 1992.

[10] J.H. Park, Y.S. Kim, I.K. Eom and K.Y. Lee, "Economic load dispatch for piecewise quadratic cost function using hopfield neural network", IEEE Trans. on Power Systems, Vol. 8, No. 3, pp. 1030-1038, August 1993.

[11] G. Singh, S.C. Srivastava, P.K. Kalra and D.M. Vinod Kumar, "Fast approach to artificial neural network training and its application to economic load dispatch", Electric Machines and Power Systems, Vol. 23, pp. 13-24, 1995.

[12] J. Kumar and G.B. Sheble, "Clamped state solution of artificial neural network for real-time economic dispatch", IEEE Trans. on Power Systems, Vol. 10, No. 2, pp. 925-931, May 1995.

[13] T.D. King, M.E. El-Hawary and F. El-Hawary, "Optimal environmental dispatching of electric power systems via an improved hopfield neural network model", IEEE Trans. on Power Systems, Vol. 10, No. 3, pp. 1559-1565, August 1995.

[14] K.Y. Lee, A. Sode-Yome and J.H. Park, "Adaptive Hopfield neural networks for economic load dispatch", IEEE Trans. on Power Systems, Vol. 13, No. 2, pp. 519-526, May 1998.

[15] S. Abe, J. Kawakami and K. Hirasawa, "Solving inequality constrained combinatorial optimization problems by the hopfield neural networks", Neural Networks, Vol. 5, pp. 663-670, 1992.

[16] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", Proceedings of National Academy of Sciences, Vol. 81, pp. 3088-3092, 1984.

[17] R.G. Ogier and D.A. Beyer, "Neural network solution to the link scheduling problem using convex relaxation", Proc. of the IEEE Global Telecommunications Conference, pp. 1371-1376, San Diego, 1990.

Appendix 1 : Data of 20 - Unit System

Unit	P_{max}	P_{min}	a_i	b_i	c_i
1	190	80	369.03	8.1817	0.00942
2	42	24	135.48	6.9467	0.08482
3	140	68	222.33	8.0543	0.01142
4	300	110	287.71	8.0323	0.00357
5	300	135	455.76	6.6020	0.00573
6	300	130	722.82	12.9080	0.00605
7	375	94	654.69	12.7960	0.00569
8	500	125	913.40	12.5010	0.00421
9	500	125	1728.3	9.1575	0.00708
10	500	125	1728.3	9.1575	0.00708
11	500	220	647.85	7.9691	0.00313
12	550	242	647.83	7.9691	0.00313
16	70	20	1207.8	13.0520	0.25098
17	70	20	810.79	21.8870	0.16766
18	60	18	641.43	26.2580	0.18362
19	60	25	832.24	16.3390	0.23915
20	60	25	834.24	16.3390	0.23915