European Journal of Operational Research xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Invited Review

A review of Hopfield neural networks for solving mathematical programming problems

Ue-Pyng Wen [a,1], Kuen-Ming Lan [a,1], Hsu-Shih Shih [b,*]

[a] Department of Industrial Engineering & Engineering Management, National Tsing Hua University, Hsinchu 30013, Taiwan, ROC
[b] Graduate Institute of Management Sciences, Tamkang University, Tamsui, Taipei 25137, Taiwan, ROC

ARTICLE INFO

Article history:
Received 1 October 2007
Accepted 4 November 2008
Available online xxxx

Keywords:
Hopfield neural networks
Energy function
Mathematical programming
penalty function
Lagrange multiplier
Primal and dual functions

ABSTRACT

The Hopfield neural network (HNN) is one major neural network (NN) for solving optimization or mathematical programming (MP) problems. The major advantage of HNN is in its structure can be realized on an electronic circuit, possibly on a VLSI (very large-scale integration) circuit, for an on-line solver with a parallel-distributed process. The structure of HNN utilizes three common methods, penalty functions, Lagrange multipliers, and primal and dual methods to construct an energy function. When the function reaches a steady state, an approximate solution of the problem is obtained. Under the classes of these methods, we further organize HNNs by three types of MP problems: linear, non-linear, and mixed-integer. The essentials of each method are also discussed in details. Some remarks for utilizing HNN and difficulties are then addressed for the benefit of successive investigations. Finally, conclusions are drawn and directions for future study are provided.

1. Introduction

In the past two decades, artificial neural networks or neural networks (NNs) have been widely developed in solving mathematical programming (MP) or optimization problems. Among various NNs, a couple of networks have been utilized for optimization (Looi, 1992; Kumar, 2005), such as Hopfield neural netwoksHNNs) (Hopfield, 1982), self-organizing feature maps (Kohonen, 1982), and Boltzmann machines (Ackley et al., 1985). As HNNs and general NN structure have been developed for almost all classes of optimization problems, it is worth reviewing their inside techniques, benefits and drawbacks, and future perspectives so that the presentation will enrich the theories and applications of HNNs in the future.

Research studies on neurons and their networks for information processing have drawn much attention by scholars in the fields of physics, electronics, and electricity. They have tried to formulate the problems mimicking the behavior of biological NNs to abstract principles of computation employed by the brain. Hopfield and Tank made a momentous contribution for neural computation by solving a traveling salesman problem (TSP) and a linear programming (LP) problem (Hopfield, 1982, 1984; Hopfield and Tank, 1985; and Tank and Hopfield, 1986). Although Shirazi and Yih (1989) pointed out some drawbacks on the approach, further improvements and verifications have been made by later investigators (Ricanek et al., 1999).

Constrained optimization problems in general assume that the minimization (or maximization) of some objective cost (or benefit) function is subject to various constraints with independent variables, and these problems are popular in various areas such as in science, engineering, and business (Bazaraa et al., 2005, 2006). Technically, the HNN approach to optimization is to handle a dynamic system in which its energy function, or a Lyapunov function (see Appendix A), characterizes the behavior of the networks and represents the problems to be solved. Its architecture can be realized by the use of an electronic circuit, possibly on a VLSI circuit, as an on-line solution with a parallel-distributed process (Cichocki and Unbehauen, 1993). These special characteristics are beneficial for real-time optimization and have been applied in many areas such as pattern recognition, scheduling, manufacturing, and numerous business applications (Looi, 1992; Sharda, 1994; Smith and Gupta, 2000; Wong et al., 2000; Zhang and Huang, 1995).

Due to renewed interests in NNs by extending HNNs, various NNs have been proposed. For instance, Kennedy and Chua (1988) extended the Tank and Hopfield's work to solve a non-linear programming (NLP) problem. Rodrı́guez-Vázquez et al. (1988, 1990) presented switched-capacitor NNs for solving LP and NLP problems. Later, Zhang and Constantinides (1992) introduced a Lagrange NNs for solving general NLP problems, and Cichocki and Unbehauen (1993) proposed the Lagrange multiplier method-based NN in solving the NLP problem. Xia et al. (2005) offered a

* Corresponding author. Tel.: +886 3 574 2653; fax: +886 3 572 2204.
E-mail addresses: upwen@ie.nthu.edu.tw (U.-P. Wen), hshih@mail.tku.edu.tw (H.-S. Shih).
1 Tel.: +886 2 8631 3221; fax: +886 2 8631 3214.

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.11.002

primal and dual method for solving linear and quadric programming problems. Nonetheless, the studies on NN on optimization have appeared in diversified journals. Interested readers can refer to *Computers and Operations Research* (No.34, Vol. 19, 1992), *Neurocomputing* (No. 13, Vol. 8, 1995), and *European Journal of Operational Research* (No. 2, Vol. 93, 1996). Some complete bibliography of this literature can be taken from books and papers (Klimasauskas, 1989; Looi, 1992; Sharda, 1994; Widrow et al., 1994; Wilson and Sharda, 1992).

Because there is no systematic review among plenty of articles on HNNs for optimization up until now, we intend to classify the essences of these HNNs for optimization as a guideline of future studies. Their special structures such as penalty function methods, Lagrange multiplier function methods, and primal and dual methods are also illustrated. In particular, energy functions of the networks characterize the behavior of the networks and support the directions to search out solutions for real-time optimization. The functions are the key to bridging the HNNs and the constraints optimization problems. On account of this significant importance, we review the literature based on three technical categories: penalty functions, Lagrange functions, and primal and dual functions. The contents are organized in the following. Section 2 discusses general functions and basic aspects of HNNs. The penalty function methods, Lagrange multiplier methods, and primal and dual functions are overviewed in Sections 3–5, respectively. The uses of HNNs for LP, NLP, and mixed-integer linear programming (MILP) are presented by each method. In Section 6, remarks are made for illustrating the possible difficulties when utilizing HNNs, and finally, conclusions and directions for future studies are given.

2. General function and basic aspects for HNNs

Neural networks have been characterized in various ways according to many relevant features (Basheer and Hajmeer, 2000; Sima and Orponen, 2001). In general, NNs have two kinds of structures (Zurada, 1992). Both have to be configured such that the application of a set of inputs produces (either direct or via a relaxation process) the desired set of outputs, and various methods exist to set out the strengths of the connections. One way is to place the weights explicitly, using a priori knowledge. The other way is to train the neural network by feeding it teaching patterns and letting it change its weights according to some learning rule. HNNs belong to non-training model. In particular, HNNs are considered as feedback (or recurrent) without learning from a pattern set (Zurada, 1992). The HNNs consist of a set of neurons and a corresponding set of unit delays, forming a multiple loop feedback system. Each feedback loops is equal to a neuron. Basically, the output of each neuron is feedback, via a unit delay element, to each of the other neurons in the system.

HNNs are not affected by additional inputs, and each neuron is fully connected with the remaining neurons by weights and always updates the connection weights. The associative memory or content-addressable memory is a memory organization that accesses memory by its contents and it always searches for the closest matching from all stored prototypes (Du and Swamy, 2006). This is analogous to the storage of information in an associative memory as biological neurons (Kohonen, 1982). In other words, the process of association and information retrieval is simulated by the dynamical behavior of a highly interconnected system of non-linear circuit elements with collective computational abilities (Hopfield, 1982).

Due to the above characteristics, HNNs are easily realized for their capability of parallel computation as a result of being a fully connected property. The networks use electric circuits to simulate the actions of biological neurons, and the basic model can be implemented by interconnecting an array or resistors, non-linear amplifiers with symmetrical output, and external bias current as shown in Fig. 1. There are n neurons, and each neuron is represented by a resistor g_i, a capacitance c_i, and a non-linear amplifier with an activation function $\psi(u_i)$, $i = 1,2,\dots,n$, respectively. The resistance–capacitance charging equation determines the rate change of u_i, $i = 1,2,\dots,n$, where u_i is the input voltage of the amplifier. The input is mapped to the output voltage v_i or $\overline{v_i}$ (normal or invert) through the function $\psi(u_i)$, $i = 1,2,\dots,n$, respectively. The choice of connecting the normal or invert output is dependent on the positive or negative value of the conductance or weight w_{ij}, $i = 1,2,\dots,n$, $j = 1,2,\dots,n$. The input of each neuron comes from two sources, external inputs I_i and inputs from other neurons with the interconnection strength or weights w_{ij} from neuron j to neuron i (Hopfield, 1982). In addition, Hopfield has shown that the sufficient condition for a stable network is that its synaptic weights are symmetric, i.e., $w_{ji} = w_{ij}$ with $w_{jj} = 0$, and, i.e. the system has a Lyapunov function (Hopfield, 1982). Here, the weights are derived from the function, and the neurons stay transit according to the local information (i.e., feedback) until a steady state is reached (Burke and Ignizio, 1992).

After a NN model is formulated, Hopfield and Tank (1985) tried to solve TSPs on a hardware circuit as illustrated above, and later they (Tank and Hopfield, 1986) extended the concept to deal with LP problems. However, the model lacks the material technologies for further development (Hopfield, 1988). Most of the later works are simulated on digital computers, not on the hardware implementation. From a practical viewpoint, HNNs have been applied to many areas of MP problems, and we organize them in a systematic way. Interested readers can refer to Burke and Ignizio (1992), Cichocki and Unbehauen (1993), Klimasauskas (1989), Looi (1992), Sharda (1994), Smith (1999), Widrow et al. (1994), and Zhang (2000) for details.

2.1. Classifications of Hopfield networks

HNN is a recurrent NN, synaptic connection pattern, whereby there is a Lyapunov function for the activity dynamics (Hopfield, 1982, 1984). In the original HNN model the state of the system evolves from any initial state to a final state where it is a (local) minimum of the Lyapunov function. Based on the output functions, HNNs can be classified into two popular forms: discrete and continuous-time models.

2.1.1. Discrete Hopfield networks

The actions of the neurons in discrete/discrete-time HNN can be illustrated as $y_j(k+1) = \psi\left[\sum_{j=1}^{n} w_{ij}x_j(k) - \theta_i\right]$, $i = 1,\dots,n$, $j = 1,\dots,n$, as in Fig. 2, where the unit delay z^{-1} is $y_j(k+1) = \psi[v_j(k+1)]$, $v_j(k+1) = \sum_{j=1}^{n} w_{ij}x_j(k) - \theta_j$, θ_i is an externally applied threshold, x_j is input neurons, and $v_j(k) = v_j(kT_s)$ ($k = 0,1,\dots,k$ denote the discrete-time and T_s is the sampling period). In general, we always assume the sample period is normalized to unity ($T_s = 1$).

In the discrete HNN, the units use a bipolar output function where the states of the units or zero, i.e., the output of the units remain the same if the current state is equal to some threshold value: $x_i = 0$, if $\sum_j w_{ij}x_j < \theta_i$; otherwise, $x_i = 1$, if $\sum_j w_{ij}x_j > \theta_i$, and no change otherwise. Here, w_{ij} is the connection weight between units i and j, and θ_i is the threshold of unit i. It is obvious that the energy function is non-increasing with the updating of the neuron's state (Hopfield, 1982). In general, the property of a recurrent network can be described as an energy function, which is used to improve the stability of recurrent network.

Hopfield (1982) presented an energy function to demonstrate its stability as

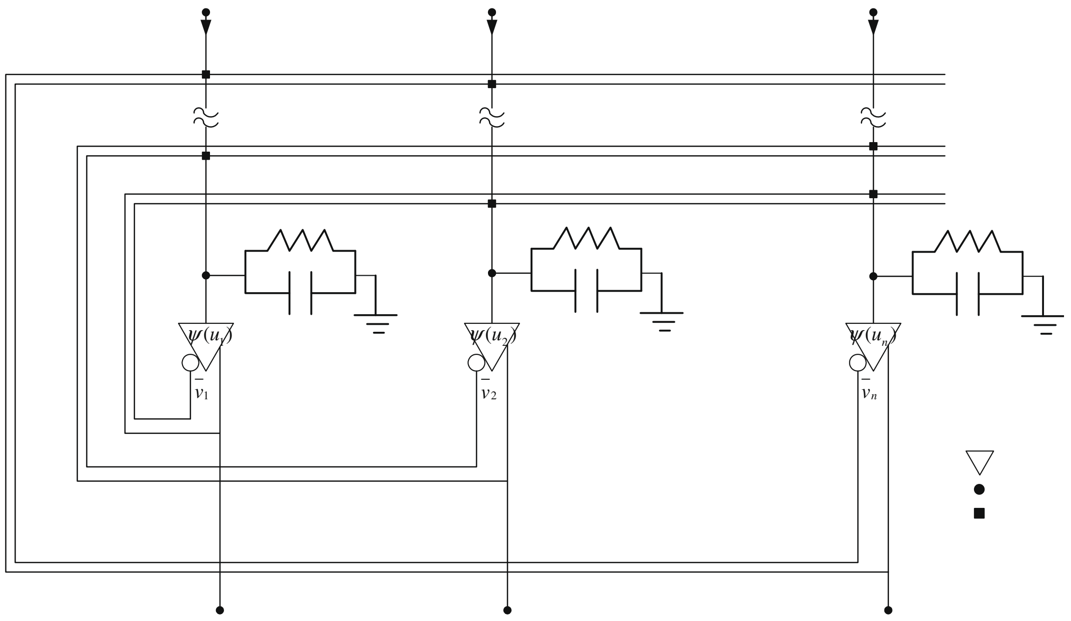

Fig. 1. Basic structure of HNNs.

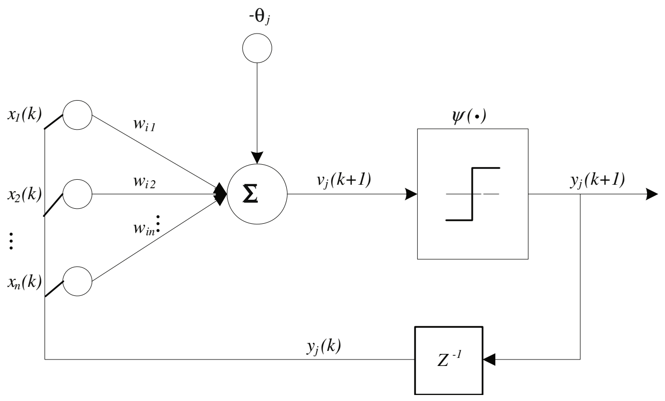

Fig. 2. Discrete-time Hopfield structure.

$$E_{\mathrm{DHNN}} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i w_{ij} x_j - \sum_{i=1}^{n} x_i I_i \tag{1}$$

and showed that the energy function is a Lyapunov function, which can lead the final state into a stable state, if the neurons are, repetitively, updated one at a time in any order. On the other hand, the local minimum of the energy function corresponds to the energy of the stored patterns. According to the work of Bruck and San (1988), the network can converge to a minimum from any initial state. Thus, the characteristic is helpful for applications. In addition, one famous NN for optimization, the Boltzmann machine, is an extension of discrete HNN. It only replaces the deterministic local search dynamics of the HNN by randomized local search dynamics (Kumar, 2005; Kurita and Funahashi, 1996), but it has only been applied to a small number of problems.

2.1.2. Continuous Hopfield networks

The continuous or continuous-time HNN is a generalization of the discrete case. The common output functions utilized in the networks are sigmoid and hyperbolic tangent functions. Fig. 3 shows the common circuit of the continuous-time HNN, where $T_{cj} = R_jC_j$, $j = 1,\ldots,n$, is the integration time constant of the jth neuron, and θ_j is an externally applied threshold. The integrator can be realized by an operational amplifier, capacitor C_j, and resistor R_j. Here, $\gamma_j > 0$ is called the forgetting factor of the integrator, which forces the internal signal v_j to zero for a zero input. To formulate the network, we establish a differential equation for the activation potential $v_j(t)$ as $T_{cj}(dv_j/dt) = -\gamma_j v_j + (\sum_{i=1}^{n} w_{ji}x_i - \theta_j)$. The output of the neuron is

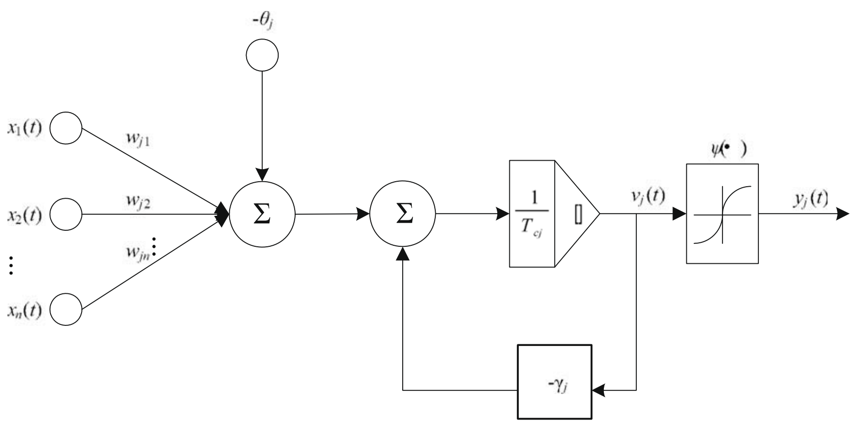

Fig. 3. Continuous-time Hopfield structure.

given by $y_j = \psi(\cdot)$, which is a continuous output function such as a sigmoid function.

Hopfield (1984) proposed an energy function for the continuous HNN as

$$E_{\text{CHNN}} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} x_i w_{ij} x_j - \sum_{i=1}^{m} x_i I_i + \sum_{i=1}^{n}\left(\frac{1}{R_i}\right)\int_0^{x_i} g_i^{-1}(x_i)dx, \tag{2}$$

where the function $g_i^{-1}(x_i)$ is a monotone increasing function. Hopfield and Tank (1985), Tank and Hopfield (1986) introduced the continuous HNN to solve the TSP and LP problems. Afterwards, many researchers implemented HNN to solve the optimization problem, especially in MP problems. Hence, the continuous model is our major concern. In general, the continuous model is superior to the discrete one in terms of the local minimum problem, because of its smoother energy surface. Hence, the continuous HNN has dominated the solving techniques for optimization problems, especially for combinatorial problems.

2.2. Methods of Hopfield networks

HNN manipulates MP problems through approximation (Hopfield and Tank, 1985). It emulates the behavior of neurons through an electronic circuit. After the circuit is established with many similar sets of components, a parallel computation can be realized for an on-line solution. These special characteristics are beneficial for real-time optimization and have been applied to solve optimization problems such as LP, NLP, quadratic programming, MILP, and multi-objective linear programming (Bouzerdoum and Pattison, 1993; Cichocki and Unbehauen, 1993; Ham and Kostanic, 2001; Hopfield and Tank, 1985; Kennedy and Chua, 1988; Looi, 1992; Rodríguez-Vázquez et al., 1988, 1990; Shih et al., 2004; Smith et al., 1998; Smith, 1999; Wang, 1992, 1996; Xia, 1996).

From the computational aspect, the operation of HNN for an optimization problem manages a dynamic system characterized by an energy function, which is the combination of the objective function and constraints of the original problem. Because there are some similarities between the function and the formulation of the NLP problem, many common techniques of NLP can thus be exploited. In such a way, three common techniques to handle NLP problems, i.e., penalty functions, Lagrange functions (or augmented Lagrange functions), and primal and dual functions, are adapted. Firstly, penalty functions use penalty parameters to combine the constraints and the objective function, and then they construct an energy function to be minimized. Secondly, Lagrange functions (or augmented Lagrange functions) take advantage of Lagrange multiples to construct an energy function to be operated. Thirdly, primal and dual functions are made up of a primal function and a dual function and try to reach the minimum gap between the functions. These three techniques are suitable for solving various MP problems such as LP, NLP, and MILP. The following sections review the literature of HNNs from a technical standpoint.

3. Penalty function methods

The penalty function method is a popular technique for optimization in which it is used to construct a single unconstrained problem or a sequence of unconstrained problems. A searching method can be adopted to search for the solution in the decision space, and the steepest descent approach is the popular technique for obtaining the searching direction (Cichocki and Unbehauen, 1993).

In our survey, many NN approaches utilize a rather basic penalty function to build the energy function and usually converge to a stable point. In the following contents, we illustrate some essential parts by the classes of LP, NLP, and MILP problems, respectively.

3.1. Linear programming problems

The common LP problems are illustrated as the primal LP form (LPF) and the dual LP form (DLPF), of which the formulations of both can be

$$\min f(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}, \quad \text{s.t. } \boldsymbol{Ax} \geqslant \boldsymbol{b}\ (= g_j(\boldsymbol{x}) \geqslant b_j,\ j = 1,2,\ldots,m)$$
$$\text{and} \quad \boldsymbol{x} \geqslant 0, \tag{LPF}$$

where $\boldsymbol{x}, \boldsymbol{c} \in \Re^{n\times 1}$, $\boldsymbol{A} \in \Re^{m\times n}$, and $\mathbf{b} \in \Re^{m\times 1}$;

$$\max g(\boldsymbol{y}) = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}, \quad \text{s.t. } \boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} \leqslant \boldsymbol{c}\ (= f_i(\boldsymbol{y}) \leqslant c_i,\ i = 1,2,\ldots,n)$$
$$\text{and} \quad \boldsymbol{y} \geqslant 0, \tag{DLPF}$$

where $\boldsymbol{c} \in \Re^{n\times 1}$, $\boldsymbol{A} \in \Re^{m\times n}$, $\mathbf{b} \in \Re^{m\times 1}$, and $\boldsymbol{y} \in \Re^{m\times 1}$ is a vector of dual independent variables.

Tank and Hopfield (1986) first proposed the NN structure to solve the LP problem, and its energy function can be defined as

$$E_{\text{TH}}(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{j=1}^{m}(g_j^{+}(\boldsymbol{x}))^2 + \sum_{i=1}^{n} x_i^2/2sR_{ii}, \tag{3}$$

where R is an $n \times n$ diagonal matrix, $s > 0$ is a penalty parameter, and $g^{+} = [g_1^{+}, g_2^{+}, \ldots, g_m^{+}]^{\mathrm{T}}$ is a penalty vector when $g_j(\boldsymbol{x}) < b_j$. Here, the penalty parameter s must be sufficiently large to lead the last term to be neglected. However, this assumption might make their model unreliable for solving the LP problem (Kennedy and Chua, 1988).

To improve Tank and Hopfield's NN, Kennedy and Chua (1988) proposed a kind of NN structure with an inexact penalty function, where the energy function is

$$E_{\text{KC}}(\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{s}{2}\sum_{j=1}^{m}(g_j^{+}(\boldsymbol{x}))^2. \tag{4}$$

Here, $s > 0$ is a penalty parameter, and for an inexact penalty rule, the solution converges to LP as $s \to \infty$. However, there is difficulty in choosing a huge number of parameters (Cichocki and Unbehauen, 1993). Rodríguez-Vázquez et al. (1988) directly use another penalty method to transform the LP problem into an unconstraint optimization problem. Their energy function is illustrated as

$$E_{\text{RV}}(\boldsymbol{x}, \alpha) = f(\boldsymbol{x}) + \alpha\left|\sum_{j=1}^{m} \min\{0, g_j(\boldsymbol{x}) - b_j\}\right|, \tag{5}$$

where $\alpha > 0$. A non-negative function will satisfy it. In addition, for each discrete-time step k, we have $x_i(k) = \max\{x_i(k),0\}$. Once the feasible region is reached, the trajectory moves toward the minimal direction of the objective function. Although Rodríguez-Vázquez et al. (1990) pointed out that their network has no equilibrium point in the classical sense, however, according to our investigation (Lan et al., 2007), their network can converge to an optimal solution of the corresponding convex programming problem from any initial state.

Maa and Shanblatt (1992) used a two-phase NN structure for solving the problem. In the first phase, $t < t_1$, t_1 is randomly selected, and the structure is the same as in Kennedy and Chua. The stability of the network is dependent on how to choose the penalty parameter s and time parameter t_1, but it is not easy to choose t_1. If the initial solution does not fall into a feasible region, then the solution does not converge to the stable state in the final. In addition, Chong et al. (1999) analyzed a class of NN models for solving LP problems by dynamic gradient approaches based on exact non-differentiable penalty functions. They developed an analytical tool helping the systems converge to a solution within a finite time.

3.2. Non-linear programming problems

Since non-linearity commonly exists everywhere, NLP problems have huge applications in the real world and have been drawn much attention in both theoretical and practical aspects. Although NLP problems are complex in computation, NN approaches with a penalty function offer a faster convergence (Hagan et al., 1996). A general NLP form is shown as (Bazaraa et al., 2006):

$$\begin{aligned} \min \quad & f(\boldsymbol{x}), \\ \text{s.t.} \quad & h_i(\boldsymbol{x}) = 0 \quad \text{for } i = 1,2,\ldots,l, \\ & g_j(\boldsymbol{x}) \leqslant 0 \quad \text{for } j = 1,2,\ldots,m, \\ & \boldsymbol{x} \in X, \end{aligned} \tag{NLP}$$

where $f, g_1,\ldots,g_m, h_1,\ldots,h_l$ are functions defined on E_n, X is a subset of, and $\boldsymbol{x}$ is a vector of n components $x_1,\ldots,x_n$. In general, the penalty function $f_p(\boldsymbol{x})$ is represented as $f_p(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{j=1}^{m} K_j\zeta_j[g_j(\boldsymbol{x})] + \sum_{i=1}^{l} K_i\zeta_i[h_i(\boldsymbol{x})]$, where K_j and K_i are commonly referred to as penalty parameters, and ζ_j and ζ_i are the terms of penalty function terms. For instance, a typical NN for solving the NLP problem can be written as $f_p(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{j=1}^{m} K_j \max\{0, g_j(\boldsymbol{x})\}^{\beta} + \sum_{i=1}^{l} K_i(|h_i(\boldsymbol{x})|^{\alpha}/\alpha)$, where $\alpha, \beta \geqslant 0$, and the penalty parameters are $K_j, K_i \geqslant 0$.

For the NLP problems, how to choose the penalty parameter is rather critical. In our survey, the penalty function methods are usually achieved in either one of the two following ways:

(i) Penalty parameters K_i and K_j are increased for network training.
(ii) Penalty parameters K_i and K_j are selected with a sufficiently large positive number that makes the unconstrained problem close to the approximation problem of NLP.

Many techniques use the steepest descent approach in searching for solutions by the NN dynamic system with the updating equations $\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \mu(\partial f_p(\boldsymbol{x})/\partial \boldsymbol{x})$, where the learning rate is $\mu > 0$. To prevent the solution from falling into a local minimum, a noise term to escape it will reach the global optimum. Hence, this dynamic update equation can be derived as $\boldsymbol{x}(k+1) = \boldsymbol{x}(k) - \mu(\partial f_p(\boldsymbol{x})/\partial \boldsymbol{x}) + \sigma^2\boldsymbol{n}$, where $\boldsymbol{n}$ is a noise vector that is randomly generated with a standard normal distribution z (zero mean and unity variance) and σ^2 is a parameter related to convergence (Cichocki and Unbehauen, 1993).

Silva et al. (2005) introduced another recurrent NN to handle optimization in which there are constrained terms in different stages without any hindrance from each other. At the same time, Effati and Baymain (2005) presented a new recurrent NN model which is simpler and more intuitive for solving convex NLP problems. They used a constraint set instead of penalty parameters to overcome the difficulty of selection. Their work is valuable for tackling NLP problems.

3.3. Mixed-integer linear programming problem

MILP problems have a number of applications in the real world. The most common description is with 0–1 variables which can be represented as (Watta and Hassoun, 1996):

$$\begin{aligned} \min \quad & f(\boldsymbol{x}, \boldsymbol{v}), \\ \text{s.t.} \quad & h_i(\boldsymbol{x}, \boldsymbol{v}) = 0, \quad i = 1,2,\ldots,q, \\ & g_j(\boldsymbol{x}, \boldsymbol{v}) \leqslant 0, \quad j = 1,2,\ldots,p, \\ & m_l \leqslant x_l \leqslant M_l, \quad l = 1,2,\ldots,n, \\ & v_k \in \{0,1\}, \quad k = 1,2,\ldots,r, \end{aligned} \tag{MILP}$$

where $f, g_1, \ldots, g_p, h_1,\ldots,h_q$ are functions defined on $E_{n\times r}$, $\boldsymbol{x}$ is a vector of n components $x_1,\ldots,x_n$ which satisfy boundaries between m_l and M_l, and $\boldsymbol{v}$ is the 0–1 integer vector of components, $v_1,\ldots,v_r$.

Since, the problem with integer variables is difficult to solve by an approximate technique, its development is rather slow. Watta and Hassoun (1996) used a coupled gradient network approach for solving a MILP problem. Their energy function $E_{\mathrm{WH}}(\boldsymbol{x},\boldsymbol{v})$ is constructed by summing the objective function $f(\boldsymbol{x},\boldsymbol{v})$ and the penalty function $P(\boldsymbol{x},\boldsymbol{v})$ to the constraints. The dynamic update is carried out by the steepest gradient descent with a fixed increment.

We can in general separate the constraints into inequality equations, equality equations, and bounded variables constraints as $P(\boldsymbol{x},\boldsymbol{v}) = G(\boldsymbol{x},\boldsymbol{v}) + H(\boldsymbol{x},\boldsymbol{v}) + V(\boldsymbol{v})$. The first penalty term of the right-hand side is $G(\boldsymbol{x}, \boldsymbol{v}) = \sum_{j=1}^{p} \vartheta[g_j(\boldsymbol{x}, \boldsymbol{v})]$, where $\vartheta(\xi) = \xi^2$ for all $\xi > 0$, and $\vartheta(\xi) = 0$ for all $\xi \leqslant 0$. If the solution is unsatisfactory with the inequality constraints, then the penalty will be effective. The second term $H(\boldsymbol{x}, \boldsymbol{v}) = \sum_{i=1}^{q} h_i^2(\boldsymbol{x}, \boldsymbol{v})$ enforces the equality constraint. The last term $V(\boldsymbol{v}) = \sum_{k=1}^{r} v_k(1 - v_k)$ is to keep the 0–1 variables feasible. Note that the larger value of penalty functions is 0.5 and falls to zero as it saturates at $v_i = 0$ or $v_i = 1$. They thus recommend an energy function as

$$E_{\mathrm{WH}}(\boldsymbol{x}, \boldsymbol{v}) = A * f(\boldsymbol{x}, \boldsymbol{v}) + B^* \sum_{j=1}^{p} \vartheta[g_j(\boldsymbol{x}, \boldsymbol{v})] + C^* \sum_{i=1}^{q} h_i^2(\boldsymbol{x}, \boldsymbol{v}) + D^* \sum_{k=1}^{r} v_k(1 - v_k). \tag{6}$$

Table 1
Penalty function methods for optimization problems.

Proposed method	Problem type	Activation function	Penalty	Learning rate	Initial state
Tank and Hopfield (1986)	LP	Sigmoid function	s must be sufficiently large	C is a positive diagonal matrix	N/A
Kennedy and Chua (1988)	LP and NLP	Sigmoid function	$s > 0$ and will be convergent as $s \to \infty$	C is a positive diagonal matrix	N/A
Rodrıguez-Vázquez et al. (1988, 1990)	LP and NLP	$x_i(k+1) = \max\{x_i(k), 0\}$	$\alpha > 0$ and $S_i = \begin{cases} 1 & \text{if } g_i(x) > 0 \\ 0 & \text{if } g_i(x) \leqslant 0 \end{cases}$	$0 < \mu_i < 1$	Any initial state
Maa and Shanblatt (1992)	LP and NLP	Sigmoid function	ε is a small positive constant and $s > 0$	N/A	Feasible variables
Watta and Hassoun (1996)	MILP	Sigmoid function	$A, B, C, D > 0$	$0 < \eta_x, \eta_v < 1$	A random initial condition near the center of phase space
Aourid and Kaminska (1996)	MILP	Sigmoid function	$a > 0$, $\mu > a/2\lambda_{\max}$, λ = the eigenvalues of constrained vector	N/A	Feasible variables
Chong et al. (1999)	LP	Sigmoid function	Parameters chosen from a selection rule	N/A	Feasible variables
Silva et al. (2005)	NLP	Sigmoid function	N/A	N/A	Feasible variables
Effati and Baymain (2005)	NLP	N/A	Parameters chosen from the constrained differential equations	N/A	Feasible variables

Note: N/A – not available.

Eq. (6) uses the two activation functions to fit the constraints $x_l = \rho_l(h_l^x) = m_l + (M_l - m_l/1 + e^{-\lambda_l h_l^x})$ and $v_k = \sigma_k(h_k^v) = (1/1 + e^{-\lambda_k h_k^v})$, where λ_l and λ_k are activation function slopes. Based on the function, it is crucial to select an initial point. If the origin is chosen, then the solution is easily restricted by the boundaries of the decision variables with a non-optimal solution. In addition, the computational load is much heavier due to the presence of many parameters, i.e., A, B, C, D, ρ_l $(l = 1,2,\dots,n)$, σ_k $(k = 1,2,\dots,r)$, η_x, η_v, λ_x, and λ_v.

Aourid and Kaminska (1996) proposed another penalty function for solving a 0–1 LP problem with inequality constraints. They transformed the general 0–1 LP problem to a concave minimization problem and then took advantage of a penalty function to transform the concave problem into an unconstrained problem. The model considers two kinds of penalty parameters for their energy function and provides a selection rule to choose the penalty parameters for the concave function.

3.4. Summary

The category of penalty function methods for HNNs is the first category to deal with MP problems. It is easy to formulate an approximating problem from the original problem with the energy function, but the penalty parameters are difficult to pick and hard to implement in the hardware. To overcome the disadvantages, several other approaches have been applied to escape the local optimal solution, including noise vector and increasing the training numbers (El-Bouri et al., 2005; Qu et al., 2006). Although these approaches are imperfect, they still open up a new viewpoint for optimization. Furthermore, we organize the developments on penalty function methods in terms of problem type, activation function, penalty, learning rate, and initial state as shown in Table 1, in order to understand how these parts are defined and utilized.

4. Lagrange multiplier related methods

Similar to the penalty function methods, Lagrange multiplier and augmented Lagrange multiplier methods merge an objective function and constraints into an energy function of the target networks. This category uses two kinds of dynamic structures to handle real variables and Lagrange multiplier (dual variables). We will review these methods for solving the LP, NLP and MINLP problems, respectively. Afterwards, we organize some essential parts of the category in Table 2 for the benefit of future research.

4.1. Linear programming problems

Aside from some earlier works, Zhang and Constantinides (1992) proposed a Lagrange method and an augmented Lagrange method for solving LP problems through HNNs. Their energy functions by Lagrange and augment Lagrange multipliers are listed as

$$E_L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{j=1}^{m} \boldsymbol{\lambda}(g_j(\boldsymbol{x})) \tag{7}$$

and

$$E_{aL}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{K}) = f(\boldsymbol{x}) + \sum_{j=1}^{m} \boldsymbol{\lambda}(g_j(\boldsymbol{x})) + \frac{1}{2}\sum_{j=1}^{m} \boldsymbol{K}|g_j(x)|^2, \tag{8}$$

where constraints $g_j(\boldsymbol{x})$ should be modified as $g_j(\boldsymbol{x}) = 0$, and $\boldsymbol{\lambda}$ and $\boldsymbol{K}$ are Lagrange multiplier vectors and positive penalty parameters, respectively. Both energy functions utilize two types of terms to fit the real and dual variables, resulting in a longer computational time. It can be proven that the addition of the quadratic penalty term in Eq. (8) increases the positive definiteness of Lagrange's Hessian matrix (Gill et al., 1981). Furthermore, if the coefficients in $\boldsymbol{K}$ are sufficiently large, then Hessian matrix of the Lagrange can force to all eigenvalues of its elements to be greater than zero. From the standpoint of implementation, this characteristic is of great importance since the solution can be sought in iterations.

Gill et al. (1981) mentioned that if a function is convex, then gradient-based searching methods are guaranteed to converge to a local minimum. In addition, Zhang and Constantinides (1992) also suggested that the penalty parameter $\boldsymbol{K}$ is no more than 5 for convergence. According to our experience (Shih et al., 2004), $\boldsymbol{K}$ should be a very small number, e.g., 0.01, in order to obtain an optimal solution.

Shih et al. (2004) introduced an augmented Lagrange multiplier method to solve multi-objective and multi-level problems based on the LP approach. Its energy function $E_S(\boldsymbol{x}, \boldsymbol{\lambda})$ is

$$E_S(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{Ax} - \boldsymbol{b}) - \frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{\lambda} + \frac{\boldsymbol{K}}{2}(\boldsymbol{Ax} - \boldsymbol{b})^T(\boldsymbol{Ax} - \boldsymbol{b}). \tag{9}$$

The function includes penalty parameters, a Lagrange multiplier and a regularization term. Each iteration requires the two steepest gradients with $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ for the direction of searching for the stable point, thus obtaining a rather close solution.

4.2. Non-linear programming problems

The Lagrange multiplier methods are commonly exploited in solving NLP problems. The NN approach uses of Lagrange multipliers to obtain its energy function with a Lagrange multiplier vector $\boldsymbol{\lambda}$ as $f_L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{l} \boldsymbol{\lambda} h_i(\boldsymbol{x})$ where $\boldsymbol{x} \in \Re^{n\times 1}$ and $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \dots, \lambda_l]^T \in \Re^{l\times 1}$. The solving procedure then utilizes the dynamic system of equations through the steepest descent approach (Ham and Kostanic, 2001). If its objective function is a non-convex function, then the procedure is easily trapped in a local minimum region. In addi-

Table 2
Lagrange multiplier related methods for optimization problems.

Proposed method	Problem type	Activation function	Penalty	Learning rate	Initial state
Zhang and Constantinides (1992)	LP and NLP	N/A	$0 < c < 5$	N/A	Any initial state
Gong et al. (1997)	NLP	N/A	N/A	$\rho(t) = \frac{\rho_0}{1+\alpha\cdot t}$ or $\rho(t) = \rho_0 \cdot \exp^{-\alpha\cdot t}$ ρ_0 is large, and α is a positive for Lagrange multiplier vector; $\mu > 0$ for variables	It usually uses the origin point for initial state
Wu and Tam (1999)	NLP	Sigmoid function	$k > 0$	N/A	Any initial state
Walsh et al. (1998)	MINLP	A hard-limit function for discrete variables, and a sigmoid function for continuous variables	$\beta_j > 0$ and σ_i is a scaling factor known as the slope in the continuous neuron i	$0 < \mu_c, \mu_\lambda < 1$	N/A

Note: N/A– not available.

tion, if it does not guarantee that the Lagrange term is a convex function, then the trajectory may exhibit oscillations near the local minimum. Hence, Ham and Kostanic (2001) described that a noise factor α could reduce the oscillation as $\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \mu_\lambda[\partial f_L(\boldsymbol{x}(k), \boldsymbol{\lambda}(k))/\partial\boldsymbol{\lambda} - \alpha\boldsymbol{\lambda}(k)]$, where $0 \leqslant \alpha \leqslant 1$.

Gong et al. (1997) provided a modified Lagrange method for solving the convex optimization problem. The NNs' dynamics demonstrate that the equilibrium point satisfies the Karush–Kuhn–Tucker (KKT) conditions of the original problem. They also provide a rule for selecting the learning rate as $\rho(t) = \rho_0/1 + \alpha\boldsymbol{t}$ or $\rho(t) = \rho_0 \cdot \exp^{-\alpha t}$, where ρ_0 is a constant and $\boldsymbol{\alpha}$ is a positive controlling parameter for the learning rate. Wu and Tam (1999) provided a novel NN model for solving the quadratic programming problem through a Lagrange multiplier.

The other relevant form is the augmented Lagrange multiplier method with extra penalty terms. The form is $f_{aL}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{K}) = f(\boldsymbol{x}) + \sum_{i=1}^{l}\boldsymbol{\lambda}h_i(\boldsymbol{x}) + \sum_{i=1}^{l}\boldsymbol{K}(h_i(\boldsymbol{x}))^2$, where $\boldsymbol{\lambda}$ and $\boldsymbol{K}$ are the vectors of Lagrange multipliers and penalty parameters, respectively (Ham and Kostanic, 2001). The algorithm forms a different optimization problem at every iterative step and it is closely related to the so-called QP-based projected Lagrange method (Gill et al., 1981). For simplicity, the augmented Lagrange multiplier can be extended to NLP problems with inequality constraints of the form:

$$f_{aL}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{K}) = f(\boldsymbol{x}) + \sum_{j=1}^{m}\boldsymbol{\lambda}\max\{0, g_j(\boldsymbol{x})\} + \sum_{j=1}^{m}\frac{\boldsymbol{K}}{2}\max\{0, g_j(\boldsymbol{x})\}^2, \tag{10}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{K}$ are the vectors of Lagrange multipliers and the penalty parameters with $\in \Re^{m\times 1}$, respectively. A compact form can be expressed as (Ham and Kostanic, 2001):

$$f_{aL}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{K}) = f(\boldsymbol{x}) + \sum_{j=1}^{m} S_j[\boldsymbol{\lambda}g_j(\boldsymbol{x}) + \boldsymbol{K}/2(g_j(\boldsymbol{x}))^2], \tag{11}$$

where $S_j = 1$ if $g_j(\boldsymbol{x}) > 0$, otherwise $S_j = 0$.

Eq. (11) is close to the structure proposed by Rodrıguez-Vázquez et al. (1988) and can find an optimal solution with $\boldsymbol{x}$ and $\boldsymbol{\lambda}$. However, this solution is sensitive to the values of Lagrange multipliers and it must take a considerable number of iterations to be convergent (Gill et al., 1981).

4.3. Mixed-integer non-linear programming problems

Walsh et al. (1998) introduced an augmented Lagrange multiplier function method to solve mixed-integer non-linear programming (MINLP) problems. Their form is similar to that in Watta and Hassoun (1996). They solved large temporal unit commitment problems in the power system. The MINLP problem is illustrated as

$$\begin{aligned} \min \quad & f(\boldsymbol{x}, \boldsymbol{v}) \\ \text{s.t.} \quad & h_i(\boldsymbol{x}, \boldsymbol{v}) = 0, \quad i = 1, 2, \ldots, q, \\ & x_l \geqslant 0, \quad l = 1, 2, \ldots, n, \\ & v_k \in \{0, 1\}, \quad k = 1, 2, \ldots, r, \end{aligned} \tag{MIN LP}$$

where f, $h_1,\ldots,h_q$ are functions defined on $E_{n\times r}$, $\boldsymbol{x}$ is a vector of n components $x_1,\ldots,x_n$, and $\boldsymbol{v}$ is a 0–1 integer vector of components $v_1,\ldots,v_r$. They used an augmented HNN with two sets of neurons: the continuous neurons x_l, $l = 1,2,\ldots,n$, and the binary neurons v_k, $k = 1,2,\ldots,r$.

The activation function g_c of the continuous neuron is a sigmoid function $V_{cij} = \psi_c(U_{cij}) = 1/2(1 + \tanh(\lambda U_{cij}))$, where λ is a scaling factor for its slope in the neuron cij, and V_{cij} and U_{cij} are the respective output and input data. Additionally, the activation function g_d for the discrete neuron dij is $V_{dij} = \psi_d(U_{dij}) = 1$, if $U_{dij} > 0$; $V_{dij} = \psi_d(U_{dij}) = 0$, otherwise. Here, V_{dij} and U_{dij} are the output and input, respectively. All neurons have a bias input I_{cij} for continuous neuron cij and I_{dij} for discrete neuron dij, respectively. The matrix T is given a standard interconnection among all neurons, continuous and discrete. Here, $T_{dij\to ckm}$ is the connection from the discrete neuron dij to the continuous neuron ckm. However, a new form of interconnection, matrix W, between neuron pairs is introduced. For example, $W_{ij\to km}$ is the connection from neuron pair cij and dij to neuron pair ckm and dkm. Their energy function of the augmented HNN is then proposed as

$$\begin{aligned} E_{\text{walsh}} = & -\frac{1}{2}\left(\sum_{i,j}\sum_{k,m}T_{ckm\to cij}V_{cij}V_{ckm}\right) - \sum_{i,j}\sum_{k,m}T_{dkm\to cij}V_{cij}V_{dkm} \\ & -\frac{1}{2}\left(\sum_{i,j}\sum_{k,m}T_{dkm\to dij}V_{dij}V_{dkm}\right) - \sum_{i,j}I_{cij}V_{cij} \\ & -\sum_{i,j}I_{dij}V_{dij} - \frac{1}{2}\left(\sum_{i,j}\sum_{k,m}W_{km\to ij}V_{dij}V_{cij}V_{dkm}V_{ckm}\right). \end{aligned} \tag{12}$$

It is proven that Eq. (12) should be (local) minimized for both the continuous variables output V_{cij} and the discrete variables output V_{dij}. Since the activation function of the discrete neuron is a hard-limit function, it will easily drop in a local optimum. If the discrete variables are selected to be zero, then the binary variables are not modified for later iterations. Moreover, the continuous activation function takes advantage of a general sigmoid function tanh (output $= (e^{input} - e^{-input})/(e^{input} + e^{-input})$), whose output is between 1 and -1. Thus, the continuous variables do not stably converge to the real variables which are greater than 1 or less than -1.

Dillon and O'Malley (2002) proposed another augmented HNN to solve MINLP problems. Their structure requires many variables and parameters, and it will adds much computational burden. Hence, the proposed network may be inappropriate for applications.

4.4. Summary

After surveying the above developments, we find that the parameter setting for Lagrange multipliers and penalties as well as the learning rates have major influences on the convergence and stability of HNNs. Lagrange multiplier and augmented Lagrange multiplier methods require more variables and parameters than do the penalty function methods. This increases the computational complexity and is difficult to control, but its structure can provide a precise solution. Furthermore, we also collect the characteristics of this category of approach for NNs in terms of problem type, activation function, penalty, learning rate, and initial state as shown in Table 2, so as to understand in what way the key parts are involved.

5. Primal and dual methods

The primal and dual methods provide another means to deal with MP problems. According to the Theorem of Duality, the primal solution is equal to the dual solution when reaching an optimal solution in LP (Bazaraa et al., 2006). However, some researchers rely on the primal and dual rules to construct the energy function. However, the function is complicated if one considers both primal and dual variables. Thus, the literature only discusses small-sized problems (Wang, 1997, 1998; Xia and Wang, 1995; Xia, 1996). This section presents the network for solving LP, networks flows, and NLP problems. In addition, we also aggregate some essential parts of HNNs in this category for the benefit of future studies.

5.1. Linear programming problems

Xia and Wang (1995) first used bounded variables to construct a new NN approach to solve the LP problem with no penalty parameters. They suggested that the equilibrium point is the same as the exact solution when simultaneously solving the primal problem and its dual problem. Hence, they defined an energy function for solving LP problems as $E_{xw}(\boldsymbol{x},\boldsymbol{y})$:

$$E_{\mathrm{XW}}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}(c^T\boldsymbol{x} - b^T\boldsymbol{y})^2 + \frac{1}{2}x^T(\boldsymbol{x} - |\boldsymbol{x}|) + \frac{1}{2}\boldsymbol{y}^T(\boldsymbol{y} - |\boldsymbol{y}|) + \frac{1}{2}\|A^T\boldsymbol{x} - b\|_2^2 + \frac{1}{2}[A^T\boldsymbol{y} - c][(A^T\boldsymbol{y} - c) - |A^T\boldsymbol{y} - c|]. \tag{13}$$

Their NN approach has three important advantages: (i) avoiding a significant amount of parameters, (ii) escaping from an infeasible solution, and (iii) being able to estimate the duality gap indirectly. Afterwards, Xia and Wang (1998) then showed many NNs structures which include penalty and primal–dual forms for solving optimization problems. They proved that the approach improved the Maa and Shanblatt's parameters selection (Maa and Shanblatt, 1992), and demonstrated that it can be used in general MP problems.

Recently, Malek and Tari (2005) represented two new methods to solve the primal LP problem and found their optimal solution for both primal and dual LP problems. They claimed that their methods give better solutions for dual variables and better optimal solutions. In addition, they realized that the new network can solve LP problems with efficient convergence within a finite time.

5.2. Network flows

Wang (1997, 1998) proposed two primal and dual methods for solving assignment problems (AP) and shortest path routing problems (SPP). The primal AP and dual AP can be formulated, respectively, as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}, \quad \text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1, \quad \forall j = 1,2,\ldots,n; \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad \forall i = 1,2,\ldots,n; \quad x_{ij} \in \{0,1\} \quad \forall i,j = 1,2,\ldots,n. \end{aligned} \tag{AP}$$

$$\max \quad \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j, \quad \text{s.t. } u_i + v_j \leqslant c_{ij} \quad \forall i,j = 1,2,\ldots n. \tag{DualAP}$$

Here, c_{ij} and x_{ij} are the respective cost and decision variables, associated with assigning entity i to position j. Here, $x_{ij} = 1$ if entity i is assigned to position j, and u_i and v_i are defined as the dual decision variables. The energy function of the primal assignment problem is

$$E_{w97}(t,x(t)) = \frac{\omega}{2}\left\{\sum_{i=1}^{n}\left[\sum_{j=1}^{n} x_{ij}(t) - 1\right]^2 + \sum_{j=1}^{n}\left[\sum_{i=1}^{n} x_{ij}(t) - 1\right]^2\right\} + \alpha\exp\left(-\frac{t}{\tau}\right)\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}(t), \tag{14}$$

where ω, α, and τ are positive constants, and $\alpha\exp(-t/\tau)$ is a temperature parameter whose role is to balance the effect of cost minimization and constraint satisfaction (Wang, 1993). On the other hand, the form of the dual AP energy function is formulated as

$$E_{w97d}(t,u(t),v(t)) = \frac{\omega}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\{g[u_i(t) + v_j(t) - c_{ij}]\}^2 - \beta\exp(-t/\tau)\sum_{i=1}^{n}[u_i(t) + v_i(t)], \tag{15}$$

where $\beta > 0$, $g(\cdot)$ is a non-negative and non-decreasing activation function given by $\psi(s) = 0$, if $s \leqslant 0$ and $\psi(s) > 0$, if $s > 0$. Similar to Eq. (14), $\beta\exp(-t/\tau)$ is a temperature parameter as well. He claimed that its computational complexity is reduced and can be implemented for a large-scale AP in a real-time basis.

Wang (1998) presented another similar primal dual method for solving SPP. Based on the edge path representation (Bazaraa et al., 2005), the SPP can be formulated as an integer LP problem for solving the minimal-cost SPP. The mathematical formulation is

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n} c_{ij}x_{ij} \\ \text{s.t.} \quad & \sum_{k=1,k\neq i}^{n} x_{ik} - \sum_{l=1,l\neq i}^{n} x_{li} = \begin{cases} 1, & \text{if } i = 1 \\ 0, & \text{if } i\neq 1 \ \&\ i\neq n \\ -1, & \text{if } i = n; \end{cases} \\ & x_{ij} \in \{0,1\}; \quad i\neq j; \quad i,j = 1,2,\ldots,n. \end{aligned} \tag{SPP}$$

The dual SPP can be formulated as

$$\begin{aligned} \max \quad & y_n - y_1, \\ \text{s.t.} \quad & y_j - y_i \leqslant c_{ij}, \quad i\neq j; \quad \forall i,j = 1,2,\ldots,n. \end{aligned} \tag{Dual SPP1}$$

The term y_i denotes the dual decision variables associated with vertex i. Since the objective function and constraints in the dual problem involve differences in variables only, an equivalent dual SPP with $n - 1$ variables can be formulated by defining $z_i = y_i - y_1$, for $i = 1, 2,\ldots, n$. Thus, the dual SPP formulated is

$$\begin{aligned} \max \quad & z_n, \\ \text{s.t.} \quad & z_j - z_i \leqslant c_{ij}, \quad i\neq j; \quad \forall i,j = 1,2,\ldots,n. \end{aligned} \tag{Dual SPP2}$$

Here, $z_1 \equiv 0$. The value of the objective function at its maximum is still the total cost of the shortest path.

The energy function for the primal SPP can be defined as

$$E_{w98}[t,x(t)] = \frac{\omega}{2}\sum_{i=1}^{n}\left[\sum_{k\neq i}[x_{ik}(t) - x_{ki}(t)] - \delta_{i1} + \delta_{in}\right]^2 + \alpha\sum_{i=1}^{n-1}\sum_{j=2,j\neq i} c_{ij}\exp(-t/\tau)x_{ij}(t), \tag{16}$$

where ω, α, and τ are positive constants, s and δ_{pq} is the Kronecker delta function defined as $\delta_{pq} = 1$ if $p = q$; otherwise $\delta_{pq} = 0$. The role of $\alpha\exp(-t/\tau)$ is explained in his previous research (Wang, 1993). Similar to the primal SPP, the dual SPP energy function is formulated as

$$E_{w98d}[t,z(t)] = \frac{\omega}{2}\sum_{i=1}^{n}\sum_{j\neq i}[g(z_j(t) - z_i(t) - c_{ij})]^2 - \beta\exp(-t/\tau)z_n(t), \tag{17}$$

where ω, β, and τ are positive constants, $\psi(\cdot)$ is a non-negative and non-decreasing activation function, and $\beta\exp(-t/\tau)$ is the temperature parameter. He proposed the rules for choosing the parameter, in which ω must be large in order to expedite the convergence of the dual routing network, and the role of β is usually set to be $\beta \approx \omega c_{\max}$ to balance the effect of constraint satisfaction and objective maximization.

5.3. Non-linear programming problems

Wu et al. (1996) presented a primal–dual method to solve LP and NLP problems and demonstrated that the solution is quickly convergent with high accuracy. They defined the primal and the dual methods for the quadratic programming (QP) problems as

$$\min \quad f_q(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{a}\boldsymbol{x}; \quad \text{s.t.} \quad \boldsymbol{D}\boldsymbol{x} = \boldsymbol{b} \quad \text{and} \quad \boldsymbol{x} \geqslant 0, \tag{QP}$$

$$\max \quad f_q(\boldsymbol{y}) = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y} - \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}; \quad \text{s.t.} \quad \boldsymbol{D}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{a} \leqslant 0, \tag{DQP}$$

where $\boldsymbol{A}$ is an $m \times m$ symmetric positive semi-definite matrix, $\boldsymbol{D}$ is an $n \times m$ matrix and $\text{rank}(\boldsymbol{D}) = m, \boldsymbol{y}, \mathbf{b} \in \Re^n$ and $\boldsymbol{x}, \boldsymbol{a} \in \Re^m$.

According to the Theorem of Duality, $\boldsymbol{x}$ and $\boldsymbol{y}$ are optimal solutions to (QP) and (DQP), respectively, if and only if $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy the following constraints:

$$\boldsymbol{D}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{D}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{a} \leqslant 0, \quad \boldsymbol{x}(\boldsymbol{D}^{\mathrm{T}}\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{a}) = 0. \tag{QP-DOP}$$

These models need much computational effort in the NN dynamic structure and takes up much memory space in each step. Hasam and Nezam (2006) presented another primal–dual NN structure for solving LP and NLP problems. The network can obtain a robust result from any initial solution. Its major advantages include no parameter setting and low costs for implementation, which are deemed to be better.

5.4. Summary

This category belongs to a discrete HNN structure so that the network can be easily implemented in hardware. However, due to its complexity, how to build a primal and dual network structure is the major concern. In addition, we see that most networks dealing with network flow problems are classified into this category. For better understanding, we organize the characteristics of the methods for NN in terms of the problem type, activation functions, penalty, learning rate, and initial state as shown in Table 3.

6. Remarks on hopfield networks

This section first defines the traveling salesman problem (TSP) solved by HNNs. Some drawbacks for the approach are then discussed. Later, remarks can be drawn on the use of HNNs.

Because Hopfield and Tank (1985) introduced a network model to solve a TSP, HNNs have dominated the NN approach for optimization. For a TSP, they gave the following definition. A set of n cities, $A, B, C, \ldots$, have pairwise distances of separation, $d_{AB}, d_{AC}, \ldots, d_{BC}, \ldots$. The problem is to find a closed tour which visits each city once, returns to the starting city, and has a minimum total travel length. Here any individual city is indicated by the output states of a set of n neurons, and can be in any one of the n positions in the tour list. For n cities, a total of n independent sets of n neurons are needed to describe a complete tour. Hence, this is a total of $N = n^2$ neurons which are displayed as an $n \times n$ square array for the TSP network. Since the representation of neural outputs of the network in terms of n rows of n neurons, the N symbols of outputs will be represented by double indices V_{Xj}. The row subscript has the explanation of a city name, and the column subscript the position of that city in a tour. To permit the N neurons in the TSP network to compute a solution to the problem, the network must be explained by an energy function in which the lowest energy state corresponds to the best path. The space over which the energy function is minimized in this limit is the 2^n corners of the N-dimensional hypercube. Consider those corners of this space which are the local minima of the energy function (Hopfield and Tank, 1985)

$$\begin{aligned} E_{\text{CHNN\_TSP}} = {} & A/2\sum_X\sum_i\sum_{j\neq i} V_{Xi}V_{Xj} + B/2\sum_i\sum_X\sum_{X\neq Y} V_{Xi}V_{Yi} \\ & + C/2\left(\sum_X\sum_i V_{Xi} - n\right)^2 + D/2\sum_X\sum_{Y\neq X} \\ & \times \sum_i d_{XY}V_{Xi}(V_{Y,i+1} + V_{Y,i-1}). \end{aligned} \tag{18}$$

Here, A, B, C, and D are positive. The four terms in Eq. (18) are defined as: (i) term 1 is zero if and only if each city row X contains no more than one 1, i.e., the rest of the entries being zero; (ii) term 2 is zero if and only if each "position in tour" column contains no more than one 1, i.e., the rest of the entries being zero; (iii) term 3 is zero if and only if there are n entries of one in the entire matrix; and (iv) term 4 describes lower values corresponding to shorter tours. The first three terms describe the feasibility requirements and result in a valid tour by zero (Hopfield and Tank, 1985). The last term represents the objective function of TSP. In addition, the four parameters A, B, C, and D must be chosen by trial and error for a particular problem. Various results illustrate different strategies for the actual implementation of HNNs, such as a sigmoid function with the value in the range $(0, 1)$. Eq. (18) can be used to derive the following expression for the strength of connection between pairs of neurons Xi and Yj. Since the time for the system to converge is an arbitrary unit, the value of the time constant of the amplifiers in the system, τ, can be set to 1.0, and gain u_0 is a scalar factor.

Although HNN is the most popular model of NNs for optimization, it suffers from some drawbacks: (i) the minimal solution does not guarantee global optimization; (ii) the artificial variables A, B, C and D are not easily implemented; and (iii) the energy function may pose many unnecessary local solutions which are difficult to avoid (Takahashi, 1998). Until now, many researchers have tried to improve the drawbacks of HNNs, and we briefly show offer attention so that future studies could follow.

Wilson and Pawley (1988) first focused on the effectiveness of computation for the HNN dealing with TSPs. They examined the original model of Hopfield and Tank (1985) through six aspects. First, the parameters, i.e., penalty parameters, city sizes, and gain u_0, are rather sensitive to the operating points, as changing by as little as 10% of their original values can prevent the system from converging. Second, the input current is at best in randomization. Third, the distance setting (for the cities) must not be the same to make reorganization easily. Fourth, the fixed city positions in a tour can reduce computational complexity, and the quality of the paths obtained is improved significantly. Fifth the algorithm of Durbin and Willshaw (1987) provides global information on its initialization. Their idea introduces a bias into the initial values to reflect the cities located on opposite sides of the unit square that are likely to be on opposite sides of a tour. Finally, the weights of an inter-neural connection must be within a range of values, instead of the same value initially, in order to avoid degeneracy in the hardware implementation. After the modifications, the HNNs can reliably solve small-size TSPs (Wilson and Pawley, 1988).

According to our survey, continuous HNNs have four deficiencies that need to be improved for solving large-size problems (Takahashi, 1998). In the following four sections, we examine these issues and try to provide some directions for future research.

6.1. Parameter setting and the initial state

How to set the parameters and choose an initial state are the two challenges in utilizing HNNs for beginners. We first list these two parts in our remarks.

6.1.1. How to set parameters

HNN could lead to an infeasible solution finally if the parameters are inadequately set for computation. In fact, its energy function often causes infeasible solutions in solving TSPs (Kennedy and Chua, 1988). The investigation on the inter-relationship among the parameters shows that TSPs do not have a scaling property and only a small range of parameter combinations result in the stable solution (Kennedy and Chua, 1988). In addition, HNNs commonly take advantages of gradient and Lagrange multiplier methods for

Table 3
Primal and dual methods for optimization problems.

Proposed method	Problem type	Activation function	Penalty	Learning rate	Initial state
Xia and Wang (1995)	LP	N/A	N/A	N/A	Feasible variables
Wu et al. (1996)	LP and NLP	N/A	N/A	N/A	Feasible variables
Wang (1997)	Assignment problem (network flows)	$f[u_{ij}(t)] = x_{\max}/\{1 + \exp[-\zeta u_{ij}(t)]\}$ or $f[u_{ij}(t)] = \zeta u_{ij}(t)]$	ω is a large number, $\alpha \geqslant \omega/c_{\max}$($c_{\max}$ is maximum cost with the edge in the graph) $\beta \approx \omega c_{\max}$ τ is must sufficient large	N/A	Feasible variables
Wang (1998)	Shortest path problem (network flows)	$x_{ij}(t) = f[u_{ij}(t)]$ – primal $x_{ij}(t) = f[z_j(t) - z_i(t) - c_{ij}]$ – dual	ω is a large number, $\alpha \geqslant \omega/c_{\max}$($c_{\max}$ is maximum cost with the edge in the graph) $\beta \approx \omega c_{\max}$ τ is must sufficient large τ is must sufficient large	N/A	Feasible variables
Malek and Tari (2005)	LP	N/A	N/A	$\eta, \eta_1, \eta_2 > 0$	Feasible variables
Hasam and Nezam (2006)	LP and NLP	N/A	N/A	N/A	Any initial state

Note: N/A – not available.

solving problems. These methods always need many parameters to build an approximate function. As the size of the problem grows, its complexity increases significantly. However, most current techniques control these parameters through a trial and error procedure. It would be a great advance if there is a guideline for setting parameters in the future.

Aiyer et al. (1990) analyzed the behavior of the continuous HNN based on the matrix expression and derived the parameter settings for the TSP with a better solution. Talaván and Yáñez (2002) proposed some analytical conditions of continuous HNNs with the problem size of up to 1000 cities. These parameters depend on the number of cities and the magnitude of their distances. Their parameter setting can be applied to other NN algorithms for improving their convergence, such as a chaotic simulated annealing algorithm (Chen and Aihara, 1995), a local minima escape algorithm (Peng et al., 1996), and a simulated annealing algorithm (Papageorgiou et al., 1998). Moreover, the setting procedure can be implemented on other energy functions (Abe, 1993; Chen and Aihara, 1995; Kamgar-Parsi et al., 1990). In the meaning time, Talaván and Yáñez (2002) also suggested another setting approach to solve a generalized quadratic knapsack problem. Tan et al. (2005) recommended an enhanced parametric formulation that maps TSPs onto a continuous HNN and illustrated its result by simulation. Unfortunately, the parameter setting in the NNs for combinatorial optimization has not been studied. However, we believe that further developments on the parameter setting of HNNs can derive to more reliable cases.

6.1.2. Choosing the initial state

The choice of an initial state affects the convergence in searching for optimal solutions (Cichocki and Unbehauen, 1993; Ham and Kostanic, 2001; Peng et al., 1996). We have collected the chosen initial states and listed them in the last column of Tables 1–3. Aside from arbitrarily choosing the states, some of them suggest that the initial state must be chosen in a feasible region to keep the final solution from not dissipating (Cichocki and Unbehauen, 1993). On the other hand, since it is uneasy to choose the state, some researches even point out that the initial state must be close to the optimal solution (Maa and Shanblatt, 1992; Zak et al., 1995). No matter how it is done, there should be a systematic way to choose the initial state for networks in the future.

6.2. Infeasible solutions

As discussed before, an infeasible solution could be obtained if one inadequately sets parameters or arbitrarily chooses initial states. Takahashi (1998) queried about the original HNNs as: (i) a solution can converge on non-optimal locally minimum solutions; (ii) a solution may converge on infeasible solutions; and (iii) solutions are very sensitive by their parameters. Thus, he provided an extended energy function to escape the infeasible solution through carefully tuning the parameters for TSPs, after which the feasible solution can be guaranteed. Furthermore, he modified a series of theories to promote a feasible solution for the extended HNN so that the solution eventually turns out to be TSP-feasible. We see that his structure is more complex than the original one, and its computational burden increases as the problem size does. It seems that Takahashi's study provides an optimistic direction for the future development, least to say, a feasible solution will be obtained in the final stage from his contribution.

6.3. Local versus global optimal solutions

It is common that the solution of HNNs is easily trapped at a local optimum (Kennedy and Chua, 1988). Various approaches are contributed to avoid local optimal solutions (Martín-Valdivia et al., 2000; Peng et al., 1996; Qu et al., 2006), but there has been no perfect approach until now. Peng et al. (1996) improved the HNNs with a local minimal searching algorithm, e.g., gradient methods or Runge–Kutta methods, and its solution is stable at a local minimum state from any initial state. In fact, their network uses the HNN for a local search and an auxiliary network for choosing the suitable initial state. Although their development is difficult to be theoretically analyzed (Qu et al., 2006), it still offers a guide for solving large-scale optimization problems.

Yalcinoz and Short (1997) proposed another approach with a defined rule to solve a large-scale economic dispatching problem, and its solution can converge to a valid region. Martín-Valdivia et al. (2000) suggested a local minima escape algorithm (Peng et al., 1996) with an augmented Lagrange multiplier method to solve the TSPs with 51 cities and 101 cities. Following this streamline, many researches studies have tries to modify HNNs or to combine with other heuristic algorithms so as to avoid the local optima

(Sexton et al., 1998, 1999). This direction can help HNNs to solve large-scale problems in the future.

6.4. Computational efficiency

A noteworthy feature of the HNNs is their ability for massive parallel computation (da Silva et al., 2006; Hopfield and Tank, 1985; Li, 1996; Lin et al., 2000; Pavlik, 1992; Wang, 1993). However, their complex structure for parallel computation may lead to difficulties in neuron updating (Looi, 1992; Smith, 1999). Therefore, how to design an effective parallel process is an important issue in the future.

Along with the trend of simulation procedure for solving problems on digital computers, Méida-Casermeiro et al. (2001) proposed a new concept for updating units in a discrete HNN for solving TSPs. Two neuron updating rules, the one-path updating rule and the two-path updating rule, which are similar to two-optimal and three-optimal tour improvement heuristics are respectively supplied (Croes, 1958). The computational results with city numbers from 51 to 532 are demonstrated, and a direction to improve the computational efficiency is also discussed. Munehisa et al. (2001) presented another HNN structure for updating the weights for a number of neurons. Four test problems from the TSP library are examined by the proposed and show a rather good performance (Reinelt, 1991). However, larger problems take up significant computational time by soft computing.

Hardware computing, on the other hand, is much faster than soft computing and is the noteworthy advantage of HNNs. It needs more elements to be implemented in the hardware for large-sized problems. Moreover, the development of VLSI circuits has improved so tremendously that implementation will not be a problem later if the material problem can be overcome.

The HNN structure causes a logistic difficulty (solutions with several local minima) in computing (Gee and Prager, 1995). For this topic, Takahashi (1999) provided two structures of the specific problems which are related to the formulation. One is how to map a real world problem onto a discrete HNN problem (Takahashi, 1999), and the other is how to organize a mathematical formulation to a continuous HNN problem (Takahashi, 1999). The first problem gives hope for future applications of HNNs if the transformation can be interpreted adequately. Although many researchers have been successful in illustrating their HNNs to solve real problems such as assignment problems, scheduling problems, shortest path problems, TSPs, and vehicle routing problems (Araújo et al., 2001; Nygard et al., 1990; Gee and Prager, 1995; Smith et al., 1996; Wang, 1996), more and more applications are expected in the future. The second problem is difficult to be conquered since it still lacks a general network, guaranteeing an optimal solution for a general MP problem. Our review collects and organizes a series of methods for solving the special cases of MP problems. Some connections among these cases and discussed drawbacks might be helpful for blossoming more general cases in the future.

We have posted four issues in dealing with the use of HNNs for optimization. These contents are practical when utilizing the networks, but we only discuss how to formulate the problems and then solve the problems through mathematical software on a digital computer. In this paper we have not implemented HNNs by hardware. This might be another story for later research.

7. Conclusions and future works

After reviewing the HNNs for optimization in the area of operations research, we can see that the networks solve various MP problems. After respectively classifying the techniques of NNs, penalty functions, Lagrange multipliers, and primal–dual methods, the various programming problems can be placed and solved by the proposed techniques. Moreover, some essential parts of NN approaches are also organized in the three tables for ease of use by beginners.

Based on the review, we have obtained some directions for future studies. First, the computational efficiency of the HNN approaches does not perform consistently well. Therefore, it is valuable to develop a more efficient NN for solving a generalized problem. Second, the HNN provides another aspect for solving a large-sized problem with a parallel process on a real-time basis. However, material advancement for establishing the necessary VLSI circuit is needed in the future. Third, although there are some drawbacks in utilizing HNNs, we suggest that a hybrid algorithm, combined with a special characteristic of other techniques, could enhance the networks for advanced applications in many areas (Lan et al., 2007). Fourth, we also believe that a creative thinking on the network structure might lead to another tremendous advance in this area. It is hoped that the review can arouse more interest on HNNs in the future.

Acknowledgements

The authors are really appreciate National Science Council, Taiwan, ROC, for the financial support under the grant number: NSC 95-2221-E-032 -030-MY3.

Appendix A. The definition of the Lyapunov function (Haykin, 1999)

Functions which make use of a continuous scalar function of the state vector are called a Lyapunov function. The Lyapunov function proves the stability of a certain fixed point in a dynamical system or autonomous differential equation. One must be aware that the basic Lyapunov Theorems for autonomous systems are a sufficient, but not necessary tool to prove the stability of an equilibrium system. Finding a Lyapunov Function in a certain equilibrium situation might be a matter of luck. Trial and error is the common method to apply when testing Lyapunov functions on some equilibrium systems.

Haykin (1999) provided a serious of theorems to define and supports us general formulation to illustrate the Lyapunov function as follows.

Theorem 1. *The equilibrium state $\bar{\mathbf{x}}$ is stable if in a small neighborhood of $\bar{\mathbf{x}}$ there exists a positive definite function V(x) such that its derivative with respect to time is negative semi-definite in that region.*

Theorem 2. *The equilibrium state $\bar{\mathbf{x}}$ is asymptotically stable if in a small neighborhood of $\bar{\mathbf{x}}$ there exists a positive definite function V(x) such that its derivative with respect to time is negative definite in that region. A scalar function V(x) that satisfies these requirements is called a Lyapunov function for the equilibrium state $\bar{\mathbf{x}}$.*

The function $V(x)$ is positive definite in the state space S, and for all x in S, it satisfies the following requirements:

(i) The function $V(x)$ has continuous partial derivatives with respect to the elements of the state vector x.
(ii) $V(\bar{\mathbf{x}}) = 0$.
(iii) $V(x) > 0$ if $x \neq \bar{\mathbf{x}}$.

Let $\bar{\mathbf{x}} = \mathbf{0}$ be an equilibrium state of the autonomous system $\dot{x} = f(x)$ and let $\dot{V}(x) = (\partial V/dx)(dx/dt) = \nabla f(x)$ be the time derivative of the Lyapunov function V. There exist three types of stable equilibrium, such as

(1) Stable equilibrium:If the Lyapunov function V is locally positive definite and the time derivative of the Lyapunov function is locally negative semi-definite, where $\dot{V}(x) \leqslant 0$, $\forall x \in S$ for some neighborhood space S, then the equilibrium is proven to be stable.
(2) Locally asymptotically stable equilibrium:If the Lyapunov function V is locally positive definite and the time derivative of the Lyapunov function is locally negative definite, where $\dot{V}(x) \leqslant 0, \forall x \in S \setminus \{0\}$ for some neighborhood space S, then the equilibrium is proven to be locally asymptotically stable.
(3) Globally asymptotically stable equilibrium:If the Lyapunov function V is globally positive definite and the time derivative of the Lyapunov function is globally negative definite, where $\dot{V}(x) \leqslant 0 \ \forall x \in \Re^n \setminus \{0\}$, then the equilibrium is proven to be globally asymptotically stable.

In many problems of the HNN, the energy function can be considered as a Lyapunov function. However, the existence of a Lyapunov function is sufficient but not necessary for stability. The Lyapunov function $V(x)$ provides the mathematical basis for the global stability analysis of the non-linear dynamical system. The global stability analysis is much more powerful in its conclusion than local stability analysis – that is, every globally stable system is also locally stable, but not vice versa.

References

Abe, S., 1993. Global convergence and suppression of spurious states of the Hopfield neural networks. IEEE Transactions on Circuits Systems 40, 246–257.

Ackley, D.H., Hinton, G.E., Sejnowski, T.J., 1985. A learning algorithm for Boltzmann machines. Cognitive Science 9, 147–169.

Aiyer, S.V.B., Niranjan, M., Fallside, F., 1990. A theoretical investigation into the performance of the Hopfield model. IEEE Transactions on Neural Networks 1, 204–215.

Aourid, M., Kaminska, B., 1996. Minimization of the 0–1 linear programming problem under linear constraints by using neural networks: Synthesis and analysis. IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications 43, 421–425.

Araújo, F., Rebeiro, B., Rodrigues, L., 2001. A neural network for shortest path computation. IEEE Transactions on Neural Network 12, 1067–1073.

Basheer, I.A., Hajmeer, M., 2000. Artificial neural networks: Fundamentals, computing, design, and application. Journal of Microbiological Methods 43, 3–31.

Bazaraa, M.S., Jarvis, J.J., Sherali, H.D., 2005. Programming and Network Flows, third ed. John-Wiley, New York.

Bazaraa, M.S., Sherali, H.D., Shetty, C.M., 2006. Nonlinear Programming Theory and Algorithms, third ed. John-Wiley, New York.

Bouzerdoum, A., Pattison, T.R., 1993. Neural network for quadratic optimization with bound constraints. IEEE Transactions on Neural Networks 4, 293–304.

Bruck, J., San, J., 1988. A study on neural networks. International Journal of Intelligent Systems 3, 59–75.

Burke, L.I., Ignizio, J.P., 1992. Neural network and operations research: An overview. Computers and Operations Research 19, 179–189.

Chen, L., Aihara, K., 1995. Chaotic simulated annealing by a neural network model with transient chaos. Neural Networks 8, 915–930.

Chong, E.K.P., Hui, S., Żak, S.H., 1999. An analysis of a class of neural networks for solving linear programming problems. IEEE Transactions on Automatic Control 44, 1995–2006.

Cichocki, A., Unbehauen, R., 1993. Neural Networks for Optimization and Signal Processing. John Wiley, New York.

Croes, G.A., 1958. A method for solving traveling salesman problems. Operations Research 6, 791–812.

da Silva, I.N., Amaral, W.C., Arruda, L.V.R., 2006. Neural approach for solving several types of optimization problems. Journal of Optimization Theory and Applications 128, 563–580.

Dillon, J.D., O'Malley, M.J., 2002. A Lagrangian augmented Hopfield network for mixed integer non-linear programming problems. Neurocomputing 42, 323–330.

Du, K.L., Swamy, M.N.S., 2006. Neural Networks in a Softcomputing Framework. Springer, London, UK.

Durbin, R., Willshaw, D., 1987. An analogue approach to the traveling salesman problem using an elastic net method. Nature 326, 689–691.

Effati, S., Baymain, M., 2005. A new nonlinear neural network for solving convex nonlinear programming problems. Applied Mathematics and Computation 168, 1370–1379.

El-Bouri, A., Balakrishnan, S., Popplewell, N., 2005. A neural network to enhance local search in the permutation flowshop. Computers and Industrial Engineering 49, 182–196.

Gee, A.H., Prager, R.W., 1995. Limitations of neural networks for solving travelings salesman problems. IEEE Transactions on Neural Networks 6, 280–282.

Gill, P.E., Murray, W., Wright, M.H., 1981. Practical Optimization. Academic, London.

Gong, D., Gen, M., Yamazaki, G., Xu, W., 1997. Lagrangian ANN for convex programming with linear constraints. Computers and Industrial Engineering 32, 429–443.

Hagan, M.T., Demuth, H.B., Beale, M., 1996. Neural Network Design. PWS Publishing Co, Massachusetts.

Ham, F.M., Kostanic, I., 2001. Principles of Neurocomputing for Science and Engineering. McGraw-Hill, New York.

Hasam, G.-O., Nezam, M.-A., 2006. An efficient simplified neural network for solving linear and quadratic programming problems. Applied Mathematics and Computation 175, 452–464.

Haykin, S., 1999. Neural Networks: A Comprehensive Foundation, second ed. Prentice-Hall, New Jersey.

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences 79, 2554–2558.

Hopfield, J.J., 1984. Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the National Academy of Sciences 81, 3088–3092.

Hopfield, J.J., 1988. Artificial neural networks. IEEE Circuits and Devices Magazine September, 3–10.

Hopfield, J.J., Tank, D.W., 1985. Neural computation of decisions in optimization problems. Biological Cybernetics 52, 141–152.

Kamgar-Parsi, Behzad, Kamgar-Parsi, Behrooz, 1990. On problem solving with Hopfield neural networks. Biological Cybernet 62, 415–423.

Kennedy, M.P., Chua, L.O., 1988. Neural networks for nonlinear programming. IEEE Transportation Circuits System 35, 554–562.

Klimasauskas, C.C., 1989. Neural networks: A new technology for information processing. Data Base 20, 21–23.

Kohonen, T., 1982. Self-organized formation of topologically correct feature maps. Biological Cybernetics 43, 59–69.

Kumar, S., 2005. Neural Networks: A Classroom Approach. International Edition, McGraw-Hill, Singapore.

Kurita, N., Funahashi, K., 1996. On the Hopfield neural networks and mean field theory. Neural Networks 9, 1531–1540.

Lan, K.M., Wen, U.P., Shih, H.S., Lee, E.S., 2007. A hybrid neural network approach to bilevel programming. Applied Mathematics Letters 20 (8), 880–884.

Li, S.Z., 1996. Improving convergence and solution quality of Hopfield-type neural networks with augmented Lagrange multipliers. IEEE Transactions on Neural Networks 7, 1507–1516.

Lin, C.L., Lai, C.C., Huang, T.H., 2000. A neural network for linear matrix inequality problems. IEEE Transactions on Neural Networks 11, 1078–1092.

Looi, C.K., 1992. Neural network methods in combinatorial optimization. Computers and Operations Research 19 (3/4), 191–208.

Maa, C.Y., Shanblatt, M.A., 1992. Linear and quadratic programming neural network analysis. IEEE Transaction on Neural Networks 3, 380–394.

Malek, A., Tari, A., 2005. Primal–dual solution for the linear programming problems using neural networks. Applied Mathematics and Computation 167, 198–211.

Martín-Valdivia, M., Ruiz-Sepúlveda, A., Triguero-Ruiz, F., 2000. Improving local minima of Hopfield networks with augmented Lagrange multipliers for large scale TSPs. Neural Networks 13, 283–285.

Méida-Casermeiro, E., Galán-Marín, G., Muñoz-Peréz, J., 2001. An efficient multivalued Hopfield network for the traveling salesman problem. Neural Processing Letters 14, 203–216.

Munehisa, T., Kobayashi, M., Yamazaki, H., 2001. Cooperative updating in the Hopfield model. IEEE Transactions on Neural Networks 12, 1243–1251.

Nygard, K.E., Jueli, P., Kadaba, N., 1990. Neural networks for selecting vehicle routing heuristic. ORSA Journal of Computing 2, 353–364.

Papageorgiou, G., Likas, A., Stafylopatis, A., 1998. Improved exploration in Hopfield network state-space through parameter perturbation driven by simulated annealing. European Journal of Operations Research 108, 283–292.

Pavlik, P., 1992. Parallel Hopfield machine. Neurocomputing 4, 89–91.

Peng, M.K., Gupta, N.K., Armitage, A.F., 1996. An investigation into the improvement of local minima of the Hopfield network. Neural Networks 9, 1241–1253.

Qu, H., Yi, Z., Tang, H., 2006. Improving local minima of columnar competitive model for TSPs. IEEE Transactions on Circuit and Systems I 53, 1353–1362.

Reinelt, G., 1991. TSPLIB – A traveling salesman problem library. ORSA Journal on Computing 3, 376–384.

Ricanek, K., Lebby, G.L., Haywood, K., 1999. Hopfield like networks for pattern recognition with application to face recognition. In: International Joint Conference on Neural Network, vol. 5, pp. 3265–3269.

Rodríguez-Vázquez, A., Domínguez-Castro, R., Rueda, A., Huertas, J.L., Sánchez-Sinencio, E., 1988. Switched-capacitor neural networks for linear programming. Electronics Letters 24, 496–498.

Rodríguez-Vázquez, A., Domínguez-Castro, R., Rueda, A., Huertas, J.L., Sánchez-Sinencio, E., 1990. Nonlinear switched-capacitor 'neural networks' for optimization problems. IEEE Transactions on Circuits Systems 37, 384–397.

Sexton, R.S., Aldaee, B., Dorsey, R.E., Johnson, J.D., 1998. Global optimization for artificial neural networks: A tabu search application. European Journal of Operational Research 106, 570–584.

Sexton, R.S., Dorsey, R.E., Johnson, J.D., 1999. Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing. European Journal of Operational Research 114, 589–601.

Sharda, R., 1994. Neural networks for the MS/OR analyst: An application bibliography. Interfaces 24, 116–130.

Shih, H.S., Wen, U.P., Lee, E.S., Lan, K.M., Hsiao, H.C., 2004. A neural network approach to multiobjective and multilevel programming problems. Computers and Mathematics with Applications 48, 95–108.

Shirazi, B., Yih, S., 1989. Critical analysis of applying Hopfield neural net model to optimization problems. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 1, 14–17 November, pp. 210–215.

Silva, I.N., Amaral, W.C., Arruda, L.V., 2005. Design and analysis of an efficient neural network model for solving nonlinear optimization problems. International Journal of Systems Science 36, 833–843.

Sima, J., Orponen, P., 2001. A Computational Taxonomy and Survey of Neural Network Models. Available from: <http://citeseer.ist.psu.edu/sima01computational.html>.

Smith, K.A., 1999. Neural networks for combinatorial optimization: A review on more than a decade of research. INFORMS Journal on Computing 11, 15–34.

Smith, K.A., Gupta, J.N.D., 2000. Neural networks in business: Techniques and applications for the operations research. Computers and Operations Research 27, 1023–1044.

Smith, K., Palaniswami, M., Krishnamoorthy, M., 1996. Traditional heuristic versus Hopfield neural network approaches to a car sequencing problem. European Journal of Operational Research 93, 300–316.

Smith, K., Palaniswami, M., Krishnamoorthy, M., 1998. Neural techniques for combinatorial optimization with applications. IEEE Transactions on Neural Networks 9, 1301–1318.

Takahashi, Y., 1998. Mathematical improvement of the Hopfield model for TSP feasible solutions by synapse dynamic systems. IEEE Transactions on Systems, Man, and Cybernetics, Part B 28, 906–919.

Takahashi, Y., 1999. A neural network theory for constrained optimization. Neurocomputing 24, 117–161.

Talaván, P.M., Yáñez, J., 2002. Parameter setting of the Hopfield network applied to TSP. Neural Networks 15, 363–373.

Tan, K.C., Tang, H., Ge, S.S., 2005. On parameter settings of Hopfield networks applied to traveling salesman problems. IEEE Transactions on Circuits and Systems – I 52, 994–1002.

Tank, D.W., Hopfield, J.J., 1986. Simple 'neural' optimization networks: An A/D converter, signal decision network and a linear programming circuit. IEEE Transactions on Circuits Systems 33, 533–541.

Walsh, M.P., Flynn, M.E., O'Malley, M.J., 1998. Augmented Hopfield network for mixed integer programming. IEEE Transactions on Neural Networks 10, 456–458.

Wang, J., 1992. Analogue neural network for solving the assignment problem. Electronics Letters 28, 1047–1050.

Wang, J., 1993. Analysis and design of a recurrent neural network for linear programming. IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications 40, 613–618.

Wang, J., 1996. A recurrent neural network for solving the shortest path problem. IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications 43, 482–486.

Wang, J., 1997. Primal and dual assignment networks. IEEE Transactions on Neural Networks 8, 784–790.

Wang, J., 1998. Primal and dual neural networks for shortest-path routing. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans 28, 864–869.

Watta, P.B., Hassoun, M.H., 1996. A coupled gradient network approach for static and temporal mixed-integer optimization. IEEE Transactions on Neural Networks 7, 578–593.

Widrow, B., Rumelhart, D.E., Lehr, M.A., 1994. Neural networks: Applications in industry, business and science. Communication of the ACM 37, 93–105.

Wilson, G.V., Pawley, G.S., 1988. On the stability of the traveling salesman problem algorithm of Hopfield and Tank. Biological Cybernetics 58, 63–70.

Wilson, R.L., Sharda, R., 1992. Neural networks. OR/MS Today 19, 36–42.

Wong, B.K., Lai, V.S., Lam, J., 2000. A bibliography of neural networks in business: Techniques and applications research: 1994–1998. Computers and Operations Research 27, 1045–1076.

Wu, A.I., Tam, P.K.S., 1999. A neural network methodology and strategy of quadratic optimization. Neural Computing and Applications 8, 283–289.

Wu, X.Y., Xia, Y.S., Li, J., Chen, W.K., 1996. A high-performance neural network for solving linear and quadratic programming problems. IEEE Transactions on Neural Network 7, 643–651.

Xia, Y., 1996. A new neural network for solving linear programming and its application. IEEE Transactions on Neural Networks 7, 525–529.

Xia, Y., Wang, J., 1995. Neural network for solving linear programming problem with bound variables. IEEE Transactions on Neural Networks 6, 515–519.

Xia, Y., Wang, J., 1998. A general methodology for designing globally convergent optimization neural networks. IEEE Transactions on Neural Networks 9, 1331–1334.

Xia, Y., Feng, G., Wang, J., 2005. A primal–dual neural network for online resolving constrained kinematic redundancy in robot motion control. IEEE Transactions on Systems Man and Cybernetics Part B – Cybernetics 35, 54–64.

Yalcinoz, T., Short, M.J., 1997. Large-scale economic dispatch using an improved Hopfield neural network. IEE Proceedings Generation, Transmission and Distribution 144, 181–185.

Zak, S.H., Upatising, V., Hui, S., 1995. Solving linear programming problems with neural networks: A comparative study. IEEE Transactions on Neural Networks 6, 94–104. The MatheWorks: <http://www.mathworks.com/>.

Zhang, X.S., 2000. Neural Networks in Optimization. Kluwer Academic, London.

Zhang, S., Constantinides, A.G., 1992. Lagrange programming neural networks. IEEE Transactions on Circuits Systems 39, 441–452.

Zhang, H.C., Huang, S.H., 1995. Applications of neural networks in manufacturing: A state-of-the-art survey. International Journal of Production Research 33, 705–728.

Zurada, J.M., 1992. Introduction to Artificial Neural Systems. West, New York.