Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Computers and Mathematics with Applications 48 (2004) 95–108

An International Journal
computers & mathematics
with applications

www.elsevier.com/locate/camwa

A Neural Network Approach to Multiobjective and Multilevel Programming Problems

HSU-SHIH SHIH*
Graduate Institute of Management Science
Tamkang University, Tamsui, Taipei 251, Taiwan, R.O.C.
hshih@mail.tku.edu.tw

UE-PYNG WEN
Department of Industrial Engineering & Engineering Management
National Tsing Hua University, Hsinchu 300, Taiwan, R.O.C.
upwen@ie.nthu.edu.tw

E. S. LEE
Department of Industrial & Manufacturing Systems Engineering
Kansas State University, Manhattan, KS 66506, U.S.A.
eslee@ksu.edu

KUEN-MING LAN
Dept. Industrial Engineering & Engineering Management
National Tsing Hua University, Hsinchu, Taiwan, R.O.C.
d917802@oz.nthu.edu.tw

HAN-CHYI HSIAO
Dept. of Industrial Engineering & Management
I-Shou University, Ta-Hsu, Kaohsiung 840, Taiwan, R.O.C.
chyi@mail2000.com.tw

(Received and accepted December 2003)

Abstract—This study aims at utilizing the dynamic behavior of artificial neural networks (ANNs) to solve multiobjective programming (MOP) and multilevel programming (MLP) problems. The traditional and nontraditional approaches to the MLP are first classified into five categories. Then, based on the approach proposed by Hopfield and Tank [1], the optimization problem is converted into a system of nonlinear differential equations through the use of an energy function and Lagrange multipliers. Finally, the procedure is extended to MOP and MLP problems. To solve the resulting differential equations, a steepest descent search technique is used. This proposed nontraditional algorithm is efficient for solving complex problems, and is especially useful for implementation on a large-scale VLSI, in which the MOP and MLP problems can be solved on a real time basis. To illustrate the approach, several numerical examples are solved and compared. © 2004 Elsevier Ltd. All rights reserved.

*Author to whom all correspondence should be addressed.

The authors really appreciate National Science Council, Taiwan, R.O.C., for the financial support under the grant numbers: NSC 89-2213-E-214-063 and NSC 90-2218-E-214-009. And special thanks shall be given to Professor Sy-Ming Guu at Yuan Ze University and Chih-Hwa Hsu at I-Shou University, Taiwan for their valuable suggestions.

Keywords—Neural network, Energy function, Multilevel programming, Multiobjective programming, Dynamic behavior.

1. INTRODUCTION

In a hierarchical organization with multiple decision makers (DMs), decentralized planning problems are tackled and hopefully solved by the use of the multilevel mathematical programming (MLP) approach. The organization explicitly assigns each agent a unique objective and a set of decision variables as well as a set of common constraints that affect all the agents [2]. There are four common characteristics of the multilevel organization:

(i) interactive decision-making units exist within a predominantly hierarchical structure;
(ii) execution of decisions is sequential, from top level to bottom level;
(iii) each unit independently maximizes its own net benefits, but is affected by actions of other units through externalities; and
(iv) the external effect on a DM's problem can be reflected in both his objective function and his set of feasible decision space.

The basic concept of the MLP method is that an upper-level DM sets his or her goal and/or decisions and then asks each subordinate level of the organization for their optima which are calculated in isolation. Lower-level DM's decisions are then submitted and modified by the upper-level DM with consideration of the overall benefits for the organization. The process is continued until a satisfactory solution is reached [3]. This decision-making process is extremely useful to such decentralized systems as agriculture, government policy, economic systems, finance, warfare, transportation, and network designs, and is particularly suitable for conflict resolution [4–7].

The simplest case of MLP problems is the bilevel programming (BLP) problem, where the top level DM has control over the vector $\mathbf{x}_1$ while the bottom level DM controls the vector $\mathbf{x}_2$. Letting the performance functions of f_1 and f_2 for the two planners are linear and bounded, then the BLP problem can be represented as

$$\max_{x1} f_1(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{c}_{11}^{\mathrm{T}}\mathbf{x}_1 + \mathbf{c}_{12}^{\mathrm{T}}\mathbf{x}_2 \qquad \text{(upper level)}, \tag{1}$$

where $\mathbf{x}_2$ solves

$$\begin{aligned} &\max_{x2} f_2(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{c}_{21}^{\mathrm{T}}\mathbf{x}_1 + \mathbf{c}_{22}^{\mathrm{T}}\mathbf{x}_2 \qquad \text{(lower level)},\\ &\text{s.t.}\quad (\mathbf{x}_1, \mathbf{x}_2) \in \mathbf{X} = \{(\mathbf{x}_1, \mathbf{x}_2) \mid \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 \le \mathbf{b},\ \text{and } \mathbf{x}_1, \mathbf{x}_2 \ge \mathbf{0}\}, \end{aligned}$$

where $\mathbf{c}_{11}$, $\mathbf{c}_{12}$, $\mathbf{c}_{21}$, $\mathbf{c}_{22}$, and $\mathbf{b}$ are vectors, $\mathbf{A}_1$, and $\mathbf{A}_2$ are matrices, and $\mathbf{X}$ represents the constraint region.

Notice that if there is no hierarchical control feature, equation (1) can be simplified to a MOP problem. In this concern, MOP and MLP are closely related.

The above-nested optimization model has been proven to be *NP*-hard by Ben-Ayed and Blair [8] and is difficult to solve. Various methods have been proposed to solve these MLP problems (see [9] for detail). We can roughly classify them into five categories (see Figure 1). The first three categories, i.e., extreme-point search, transformation approach, and descent and heuristic, can be referred to as the traditional approaches. And the last two, i.e., intelligent computation (instead of evolutionary approach in [9]) and interior point approach, are based on more recent developments. The basic concept of extreme-point search, Category I, is to seek a compromise vertex by simplex algorithm based on adjusting the control variables. The transformation approach, Category II, involves transforming the lower-level problems into constraints for the higher level by the use of various techniques such as Karush-Kuhn-Tucker (KKT) conditions, penalty functions, barrier functions, etc. Category III is developed for solving discrete or nonlinear MLP problems, and is based on existing search techniques or heuristic approaches such as gradient techniques,

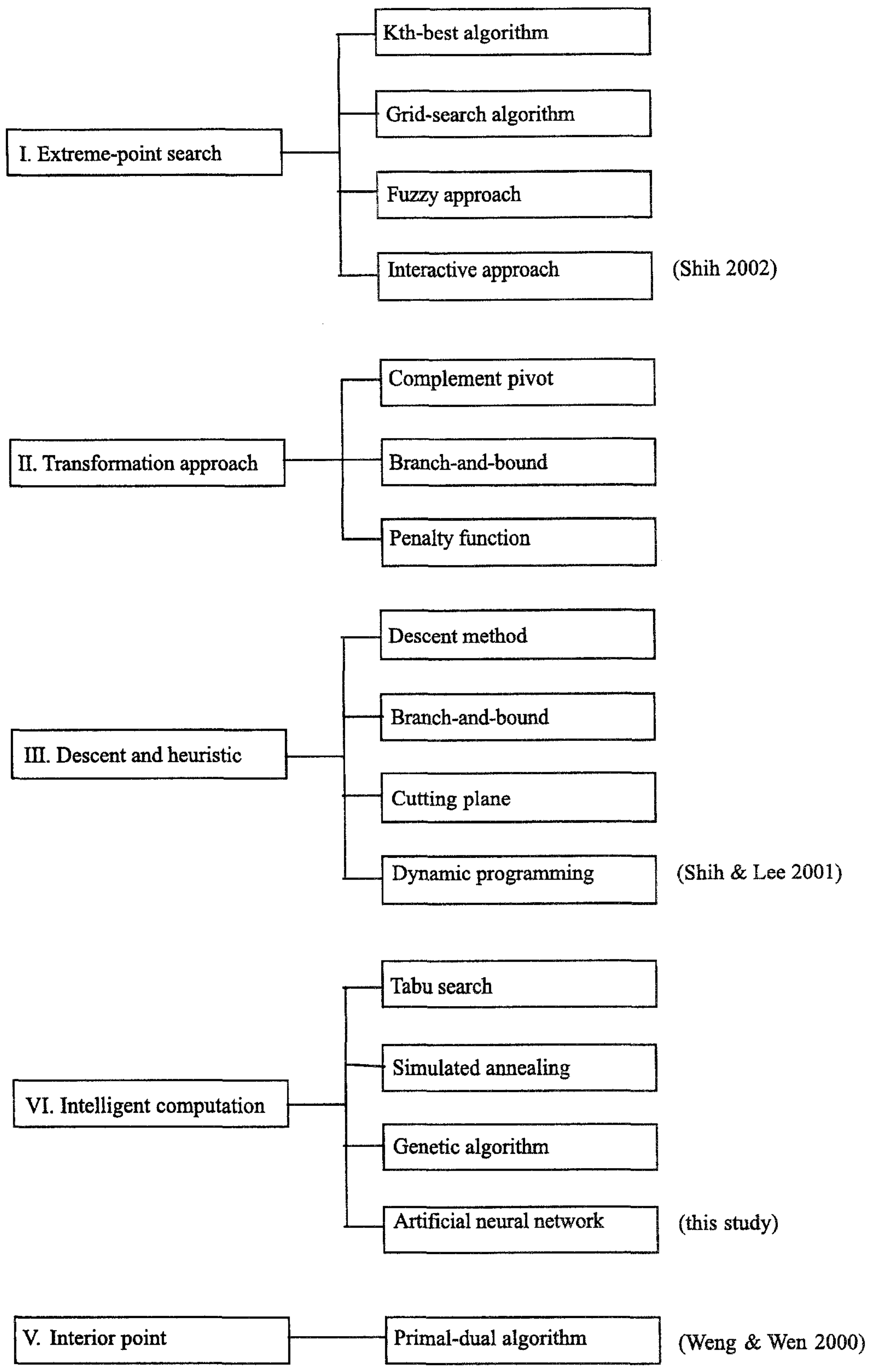

Figure 1. A taxonomy of methods for multilevel programming problems (extension of [9]).

cutting-plane algorithm, and branch-and-bound heuristics. The newly developed interior point method [10] is less sensitive to problem size and would be suitable for solving MLP problems. We append this method to Lee and Shih's original classification [9] as Category V. Category VI is based on the recent developments in intelligent computations, which are especially suited for solving NP-hard problems such as the multiploy problems. This category is relatively new and includes Tabu search, simulated annealing, and genetic algorithm. Artificial neural networks (ANNs) can also be included in this category and this paper will investigate the feasibility of using ANN to solve the MLP problems.

In real-world applications, the dynamic nature of the MLP makes the problem very difficult to solve. Although various approaches have been suggested to lessen the burden of computation such as the use of fuzzy logic (Category I) [11], dynamic programming (Category III) [12], and interactive approach (Category I) [13], further research is still needed to obtain better approaches. ANN, with its dynamic representation, appears to be a promising technique for solving this difficult problem on a real-time basis. This paper explores this nontraditional approach, ANN, to solve the MLP and MOP problems. Furthermore, real-time solution of these problems is also explored. Notice that we temporarily arrange the interactive approach as Category I. However, interactive approach can also be developed based on other methods in other categories.

Various investigators have applied ANN to solve optimization problems. Smith [14] summarized and reviewed the various approaches and divided the techniques into two categories:

(1) Hopfield neural network approach and
(2) self-organizing neural network approach.

Although other networks, such as the Boltzmann machine, have also been applied to solve optimization problems, this machine approach can be considered as an extension of the Hopfield network [15].

Burke and Ignizio [15] claimed that there are two types of networks for emulating neural information processing of operations research problems: adaptive networks and nonadaptive networks. Hopfield network belongs to the latter case, in which information is processed through node activations and computation with no learning from examples. Its weights are derived from an energy function, or serving as a Lyapunov function, and the nodes stay transit according to the impact of local information (i.e., feedback) until a steady state is reached. Note that Hopfield and Tank's network is recurrent [16].

2. ANN FOR OPTIMIZATION

ANN is a promising technique to solve optimization problems because it can emulate the operations of the brain and uses parallel processing to save computational time [17]. The method dates back to six decades. McCulloch and Pyne [18,19] utilized logical calculus to emulate nervous activities. Since then, various types of analogue neural networks have been proposed for computation. In particular, Hopfield and Tank [1] made a momentous contribution for neural computation by solving a traveling salesman problem. Many investigators followed their work. Although Shirazi and Yih [20] present some weakness of the approach, many followers make further improvements and verifications (see [21]). Since then ANN approaches have been widely accepted as a competitive approach to traditional optimization techniques.

In addition to the use of ANN to solve the traveling salesman problem, ANN was also used for solving linear programming (LP) problems. For example, Zak *et al.* [22] compared three ANNs in terms of their model complexity, complexity of individual neurons, and accuracy of solutions. Xia [23] compares three ANNs and proposes a network for solving the general LP problems through using simple hardware. At the same time, Cichocki *et al.* [17] recommend a new network to manage LP problems on-line by a simplified low-cost analog network. These works give us a trend for making further applications.

The ANN approach to optimization is to deal with a dynamic system, in which the energy function characterizes the behavior of the network and represents the problem to be solved. An ANN architecture can be realized by the use of an electronic circuit for on-line solution with parallel-distributed process. This is known as programming without computation [24]. These special characteristics are beneficial for real-time optimization, and can be applied to many areas such as vehicle routing, pattern recognition, scheduling, manufacturing, and numerous business applications [16,25–27].

For a LP problem with inequality constraints, we have

$$\begin{aligned} &\min f(\mathbf{x}) = \mathbf{c}^{\mathrm{T}}\mathbf{x}, \\ &\text{s.t.}\ \ \mathbf{A}\mathbf{x} - \mathbf{b} \le \mathbf{0}, \\ &\qquad \mathbf{x} \ge \mathbf{0}, \end{aligned} \tag{2}$$

where $\mathbf{x}, \mathbf{c} \in R^{n\times 1}$, $\mathbf{A} \in R^{m\times n}$, and $\mathbf{b} \in R^{m\times 1}$.

The energy function $E(\mathbf{x})$ can be established using the Lagrange multiplier method as follows:

$$E(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{c}^{\mathrm{T}}\mathbf{x} + \boldsymbol{\lambda}^{\mathrm{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}) - \frac{\alpha}{2\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\lambda}} + \frac{K}{2(\mathbf{A}\mathbf{x} - \mathbf{b})^{\mathrm{T}}(\mathbf{A}\mathbf{x} - \mathbf{b})}, \tag{3}$$

where $K, \boldsymbol{\lambda} \in R^{m\times 1}$ and $\mathbf{x} \ge \mathbf{0}$. In addition, the regularization term $\alpha\boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\lambda}$ is for improving the stability of the method [28], and α is a small positive number, e.g., 0.001.

Taking partial derivate of equation (3) with respect to $\mathbf{x}$ and $\boldsymbol{\lambda}$, we obtain the following differential equations:

$$\frac{\partial E(\mathbf{x}, \boldsymbol{\lambda})}{\partial \mathbf{x}} = \mathbf{c} + K\mathbf{A}^{\mathrm{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}) + \mathbf{A}^{\mathrm{T}}\boldsymbol{\lambda}, \tag{4}$$

$$\frac{\partial E(\mathbf{x}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}} = \mathbf{A}\mathbf{x} - \mathbf{b} - \alpha\boldsymbol{\lambda}. \tag{5}$$

After applying the steepest descent method, equations (4) and (5) can be converted into a system of difference equations in discrete form

$$\begin{aligned} x_j(k+1) &= \begin{cases} x_j(k) - \mu_j(k)\left\{c_j + \sum\limits_{i=1}^{m} a_{ij}\left[Kr_i(k) + \lambda_i(k)\right]\right\}, & \text{if } x_j(k+1) > 0, \\ 0, & \text{if } x_j(k+1) \le 0. \end{cases} \\ \lambda_i(k+1) &= \lambda_i(k) + \nu_i(k)\left[r_i(k) - \alpha\lambda_i(k)\right], \end{aligned} \tag{6}$$

where $r_i(k) = \sum_{j=1}^{n} a_{ij}x_j - b_i$, $K \ge 0$, $\alpha \ge 0$, and $\mu_j(k)$ and $\nu_i(k)$ are positive learning rate parameters.

Note that the update equations for the independent vectors in the above expressions guarantee that all components remain nonnegative. The ANN architecture realization of this process can be implemented by an electronic circuit, and will be simulated through MATLAB software [29] in this paper.

A drawback in the penalty function approach is that the penalty parameter K has to be large enough in order to get a solution with reasonable accuracy. However, from a practical implementation standpoint, a very large parameter value is not convenient [21]. We tested a model, which always obtains a solution with accuracy in the range between 0.1 and 1.0. Also, in order to avoid oscillation of the system, the discrete-time learning rate parameters μ and ν in the descent algorithm cannot be too large. The interested readers can follow the procedure of Ku and Lee [30] to choose the appropriate value. We tested the model with a small positive number, e.g., 0.005 to 0.01, which is generally regarded as reasonable.

3. ANN FOR MULTIPLE OBJECTIVE PROGRAMMING

Using ANN to solve the MOP is fairly new, and only a limited number of investigations have been carried out. Sun *et al.* [32,33] focused on the elicitation, representation, and utilization of the preference information of a DM in a feed-forward ANN framework. Gen *et al.* [34] proposed a new neural network technique for solving fuzzy multiobjective LP problems based on the two-phase approach of Lee and Li [35]. Recently, Balakrishnan *et al.* [36] utilized a compromise factor α $(0 \leq \alpha \leq 1)$ to reflect the preference given to the two objectives of emission and economic dispatch. And one objective function was established and solved by using the ANN technique. The first one only tries to obtain the decision information for the interactive procedure through ANN, and the third one is simply a weighting method [31] with two objectives. However, the second one indeed directly utilizes ANN to solve the MOP problem. This work verifies the feasibility of the ANN approach for solving the MOP problems in spite of the fact that some details are omitted, and enlightens a path for our development.

To deal with the problem of MOP, we utilize the method of global criterion [37] with a normalized process [38]. The essence of this approach includes the following:

(a) reference points, i.e., the concept of an ideal system,
(b) distance, i.e., location of alternatives away from reference points, and
(c) normalization, i.e., the process to eliminate nonmeasurability among objectives.

Note that, based on two reference points in each objective, Hwang *et al.* [38] proposed a normalized process to obtain a better estimation.

Reference points of the technique are positive ideal solution (PIS) $f_k^+(\mathrm{x})$ and negative ideal solution (NIS) $f_k^-(\mathrm{x})$ [35]. A PIS is defined, for each objective, as the maximum (the best) solution of the single-objective maximization problem, and the minimum (the best) solution of the single-objective minimization problem. NIS is defined as the opposite of PIS. In other words, for each objective, the maximum (or worst) solution of the minimization problem, and the minimum (the worst) solution of the maximization problem. A total $2k$ numbers of PISs and NISs are generated for a problem with k objectives. In addition, the distance measured is related to Minkowski's L_P metric with some modifications (see [35,39]). The L_P metric defines distance between two points, i.e., one objective f_k and its ideal solution $f_k(\mathrm{x}^+)$, in k-dimensional space

$$d_p = \left[\sum_k \left(f_k^+(\mathrm{x}) - f_k(\mathrm{x})\right)^p\right]^{1/p}, \qquad \text{for } p \geq 1.$$

The parameter, p, for distance setting depends on the preference of the DM or the ease of manipulation. There are several popular settings:

(a) $p = 1$, Manhattan distance;
(b) $p = 2$, Euclidean distance;
(c) $p = \infty$, Tchebycheff distance.

For the convenience of processing, it is quite common to use an extreme case, i.e., $p = 1$ or $p = \infty$, so that the above representation can be kept in linear form.

The method of global criterion is to minimize the distance function of the multiple objectives as

$$\begin{aligned} &\min \left\{\sum_k \left[\frac{\left(f_k^+(\mathrm{x}) - f_k(\mathrm{x})\right)}{\left(f_k^+(\mathrm{x}) - f_k^-(\mathrm{x})\right)}\right]^p\right\}^{1/p}, \qquad k = 1, \dots, K, \\ &\text{s.t. } p \geq 1, \quad x \in \mathbf{X}, \end{aligned} \tag{7}$$

where $\mathbf{X}$ is the constraint set of the above MODM expression.

Based on equation (7), equations (3)–(6) can be obtained, and the problem can be solved by the ANN approach. In the present paper, MATLAB software [28] is used to carry out the solution process.

To illustrate the approach, the following example is solved.

EXAMPLE 1. A two-objective programming problem [9].

$$\begin{aligned}
&\max f_1 = 2x_1 - x_2 && \text{(objective 1)},\\
&\max f_2 = x_1 + 2x_2 && \text{(objective 2)},\\
\text{s.t.}\quad & 3x_1 - 5x_2 \le 15, &&\\
& 3x_1 - x_2 \le 21, &&\\
& 3x_1 + x_2 \le 27, &&\\
& 3x_1 + 4x_2 \le 45, &&\\
& x_1 + 3x_2 \le 30, &&\\
& x_1, x_2 \ge 0, &&
\end{aligned}$$

$\mathbf{X}$ will be used to represent the constraint set.

After obtaining the PIS $\mathbf{f}^{+}(\mathbf{x}) = (13.5, 21)$ and NIS $\mathbf{f}^{-}(\mathbf{x}) = (-10, 0)$, we can set the new auxiliary problem through the method of global criterion, with $p = 1$, as

$$\begin{aligned}
\min d_p &= \left\{\left[\frac{(13.5 - f_1)}{(13.5 + 10)}\right]^p\right\}^{1/p} + \left\{\left[\frac{(21 - f_1)}{(21 - 0)}\right]^p\right\}^{1/p},\\
\text{s.t.}\quad & \mathbf{x} \in \mathbf{X},
\end{aligned}$$

where $k = 2$.

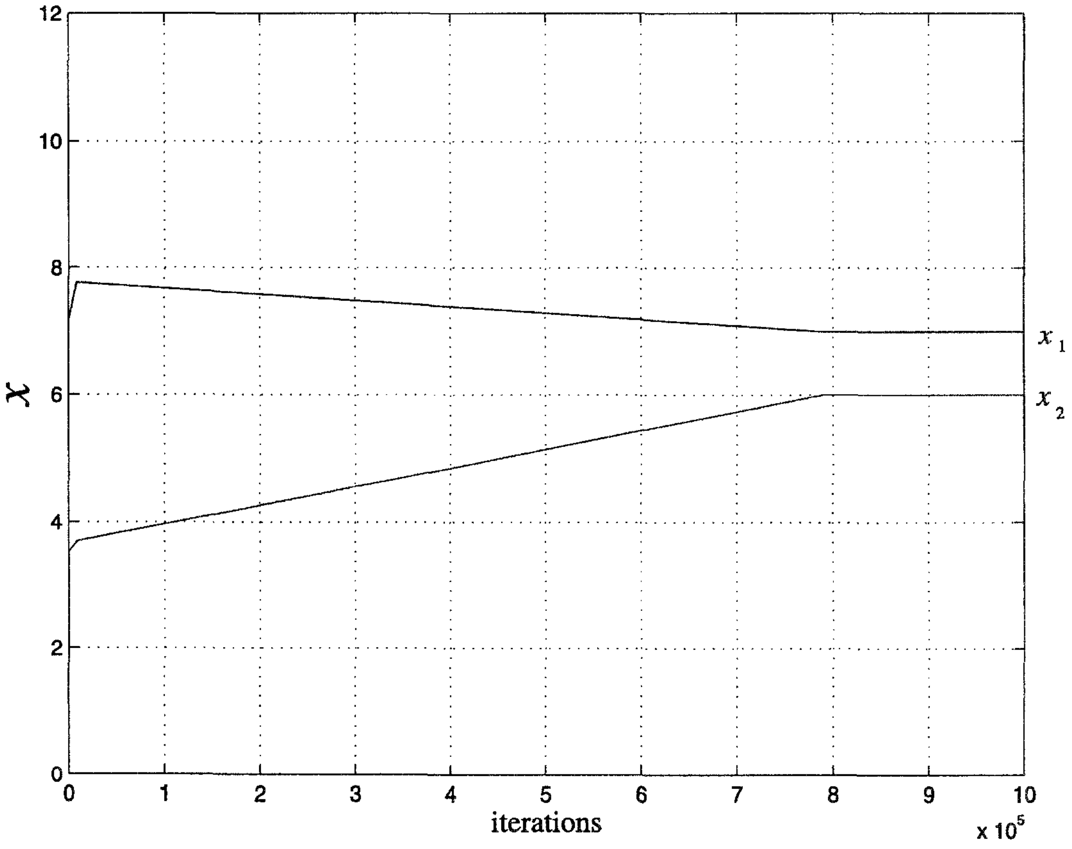

Figure 2. Solution trajectories, Example 1 ($p = 1$).

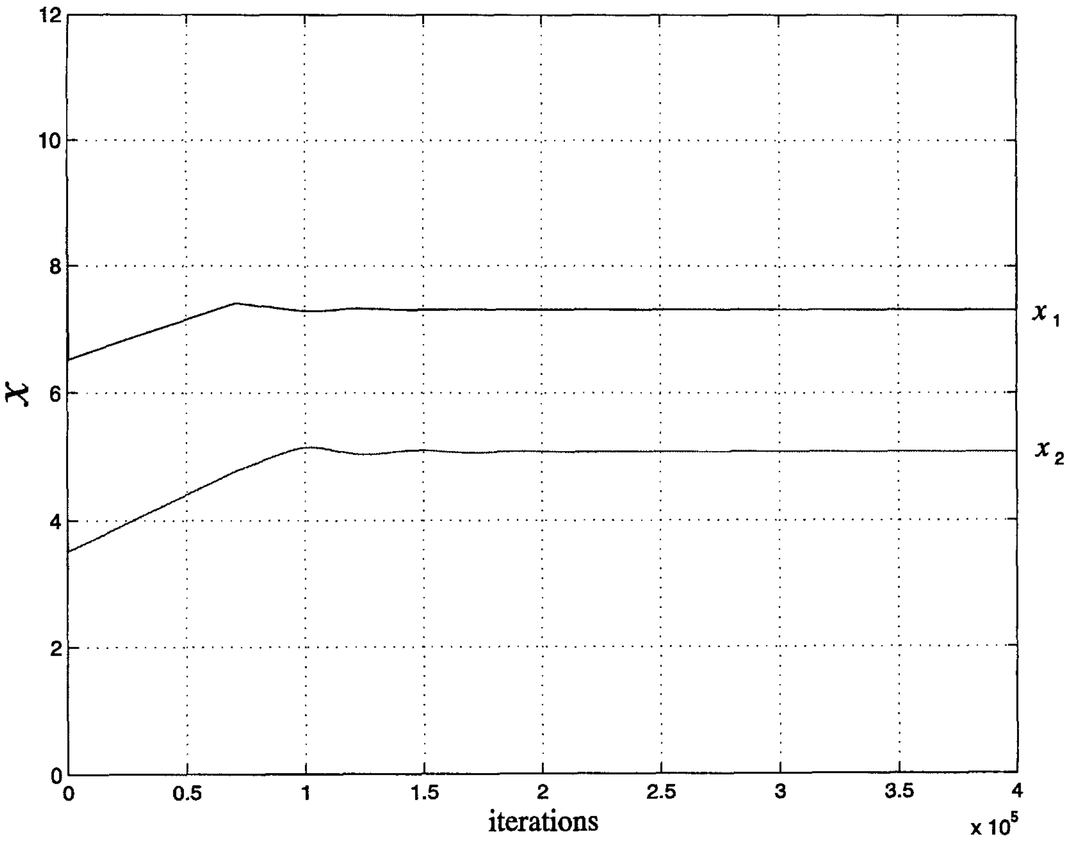

Figure 3. Solution trajectories, Example 1 ($p = \infty$).

The vectors $\mathbf{b}_{7\times 1}$, $\mathbf{c}_{1\times 9}$, and the matrix $\mathbf{A}_{7\times 9}$ are known and the problem is solved with $p = 1$. The minimal Manhattan distance is $d_p = 0.32952$ with the decision vector $\mathbf{x}^* = (x_1^*, x_2^*) = (7.000, 6.000)$ and the objective vector $\mathbf{f}^* = (f_1^*, f_2^*) = (8.000, 19.000)$ at $K = 0.5$. We can find that the same result can be obtained by using extreme-point search approach (Category I) [9].

For $p = \infty$, the formulation can be expressed as

$$\begin{aligned} &\min d_p, \\ &\text{s.t.}\ \ \mathbf{x} \in \mathbf{X}, \\ &\qquad \frac{(13.5 - f_1)}{(13.5 + 10)} \le d_p, \\ &\qquad \frac{(21 - f_1)}{(21 - 0)} \le d_p. \end{aligned}$$

By setting up the vectors $\mathbf{b}_{11\times 1}$, $\mathbf{c}_{1\times 8}$, and the matrix $\mathbf{A}_{8\times 11}$, the problem was solved. The minimal Tchebycheff distance $d_p = 0.168526$ with $\mathbf{x}^* = (x_1^*, x_2^*) = (7.3073, 5.0782)$ and the objective vector $\mathbf{f}^* = (f_1^*, f_2^*) = (9.5364, 17.4529)$ at $K = 0.5$. We can also find that the same solution can be obtained by using Lingo code [40].

The solution trajectories with $\mathbf{x}$ versus the number of iterations are shown in Figures 2 and 3 for $p = 1$ and $p = \infty$, respectively. The influences of the K values on the same solution range between 0.1 and 3.0 for $p = 1$ as well as $p = \infty$, with learning rate $\mu = 0.01$.

4. ANN FOR MULTIPLE LEVEL PROGRAMMING

The most popular approach to solve the nested MLP optimization problems is using the KKT conditions and transforms the original problem to its first level auxiliary problem [7]. In this way, the problem is reduced to a regular mathematical programming problem. However, the transformed problem or the auxiliary problem is difficult to solve due to nonlinearity, which was introduced through the complementary slackness conditions (CSCs). Many researchers have devised approaches to solve this nonlinear problem [9]. We adopt the mixed-integer programming approach by introducing a binary variable, $\eta \in \{0,1\}$. Thus, the product constraint, $wq = 0$, is replaced by the following two simpler inequalities: $q \leq M\eta$ and $w \leq (1-\eta)M$ [41]. The auxiliary formulation now becomes

$$\begin{aligned} \max f_1(\mathbf{x}_1, \mathbf{x}_2) &= \mathbf{c}_{11}^{\top}\mathbf{x}_1 + \mathbf{c}_{12}^{\top}\mathbf{x}_2, \\ \text{s.t.}\quad & \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 \leq \mathbf{b}, \\ & \mathbf{w} \leq (1-\eta)M, \\ & \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 - \mathbf{b} \leq M\eta, \\ & \mathbf{w}^{\top}\mathbf{A}_2 = \mathbf{c}_{22}, \\ & \eta \in \{0,1\}, \\ & \mathbf{x}_1, \mathbf{x}_2, \mathbf{w} \geq \mathbf{0}, \end{aligned} \tag{8}$$

where M is a large positive constant.

The proposed ANN approach is applied to the above expression. Following the same procedure as that used in Section 3, the MLP problem can be solved. Furthermore, the KKT conditions can be applied to the much more general case of bilevel decentralized programming problems.

To demonstrate the approach, two examples were solved. One is a bilevel programming problem and the other is a bilevel decentralized programming problem.

EXAMPLE 2. A bilevel programming problem [7].

$$\max_{x1} f_1 = -2x_1 + 11x_2 \qquad \text{(upper level)}$$

where x_2 solves

$$\begin{aligned} \max_{x2} f_2 &= -x_1 - 3x_2 \qquad \text{(lower level)}, \\ \text{s.t.}\quad & x_1 - 2x_2 \leq 4, \\ & 2x_1 - x_2 \leq 24, \\ & 3x_1 + 4x_2 \leq 96, \\ & x_1 + 7x_2 \leq 126, \\ & -4x_1 + 5x_2 \leq 65, \\ & x_1 + 4x_2 \geq 8, \\ & x_1, x_2 \geq 0, \end{aligned}$$

The above constraint set will be represented by $\mathbf{Y}$.

Applying the KKT transformation, the above original problem becomes

$$\begin{aligned} \max f_1 &= -2x_1 + 11x_2, \\ \text{s.t.}\quad & \mathbf{x} \in \mathbf{Y}, \\ & x_1 - 2x_2 - 4 \leq M\eta_1, \quad w_1 \leq (1-\eta_1)M, \\ & 2x_1 - x_2 - 24 \leq M\eta_2, \quad w_2 \leq (1-\eta_2)M, \\ & 3x_1 + 4x_2 - 96 \leq M\eta_3, \quad w_3 \leq (1-\eta_3)M, \\ & x_1 + 7x_2 - 126 \leq M\eta_4, \quad w_4 \leq (1-\eta_4)M, \\ & -4x_1 + 5x_2 - 65 \leq M\eta_5, \quad w_5 \leq (1-\eta_5)M, \\ & -x_1 - 4x_2 + 8 \leq M\eta_6, \quad w_6 \leq (1-\eta_6)M, \end{aligned}$$

$$
\begin{aligned}
&-2w_1 - w_2 + 4w_3 + 7w_4 + 5w_5 + 4w_6 = -3,\\
&\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6 \in \{0,1\},\\
&w_1, w_2, w_3, w_4, w_5, w_6 \geq 0.
\end{aligned}
$$

Making use of the procedure discussed earlier, the above expression can be reformulated as an unconstrained optimization problem with branch-and-bound tree for 0-1 variables η_n, $n = 1, 2, \ldots, 6$. Using the MATLAB software [28] and with $M = 1000$, $K = 0.5$, and $\mu = 0.005$, the problem was solved and the results are $(x_1^*, x_2^*) = (17.5000, 10.9000)$ with $(f_1^*, f_2^*) = (85.0909, -50.2000)$. Compared to the traditional approach [9] with Lingo [40], the solution is $(x_1^*, x_2^*) = (17.45455, 10.90909)$ with $(f_1^*, f_2^*) = (85.09091, -50.18182)$.

The solution trajectory with x versus the number of iterations is shown in Figure 4 and the influences of the values of K $(0.1 \leq K \leq 1.0)$ respect to the reasonable solutions are tabulated in Table 1.

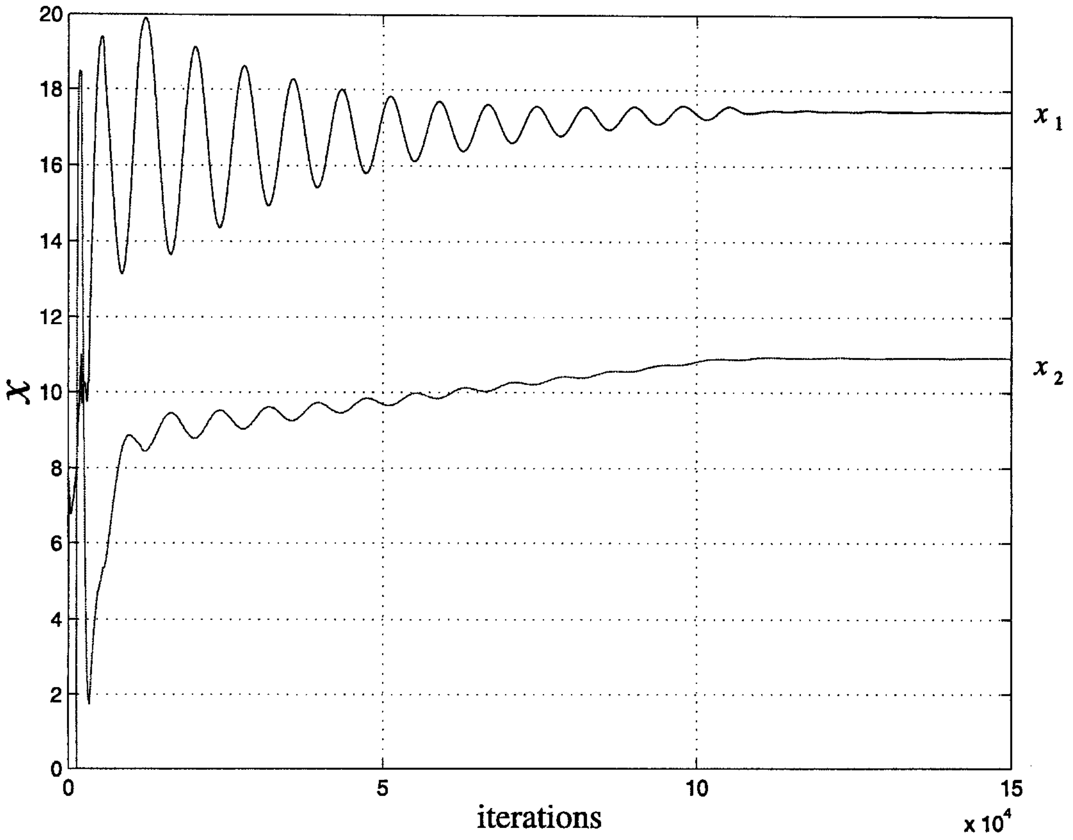

Figure 4. Solution trajectories, Example 2.

Table 1. The solutions with different values of parameter K (Example 2).

	$K = 0.1$	$K = 0.5$	$K = 1.0$
x_1	17.4546	17.5000	17.5000
x_2	10.9089	10.9000	10.9000
f_1	85.0889	85.0909	85.0909
f_2	−50.1813	−50.2000	−50.2000

Note: Learning rate parameter $\mu = 0.005$ here.

EXAMPLE 3. A bilevel decentralized system with a center at the upper level and three independent divisions at the lower level [4].

$$\max_{x1} f_1 = x_1 + x_2 + 2x_3 + x_4,$$

where x_2, x_3, and x_4 solve

$$\begin{aligned}
&\max f_{21} = x_1 + 3x2 + x_3 + x_4; \\
&\max f_{22} = x_1 + x_2 + 3x_3 + x_4; \\
&\max f_{23} = x_1 + x_2 + x_3 + 3x_4; \\
&\text{s.t.}\quad 3x_1 + 3x_2 \le 30, \quad 2x_1 + x_2 \le 20, \\
&\qquad x_3 \le 10, \quad x_2 + x_4 \le 15, \\
&\qquad x_4 \le 10, \quad x_1 + 2x_2 + 2x_3 + x_4 \le 40, \\
&\qquad x_1, x_2, x_3, x_4 \ge 0.
\end{aligned}$$

The above constraint will be expressed as $\mathbf{Z}$.

After applying the KKT conditions to the original problem, a new auxiliary problem with only one level will be obtained. By exploiting Fortuny-Amat and McCarl's suggestion [41], we obtain a simple expression as

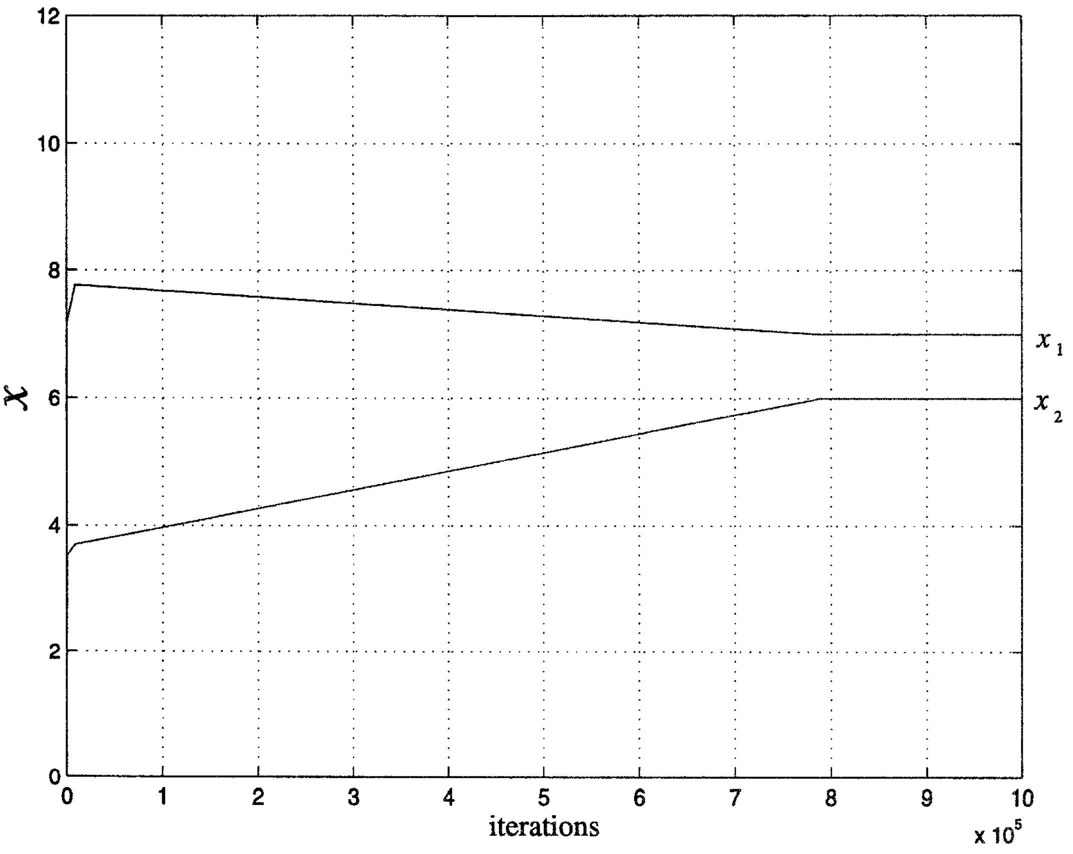

Figure 5. Solution trajectories, Example 3.

Table 2. The solutions with different values of parameter K (Example 3).

	$K = 0.1$	$K = 0.5$	$K = 1.0$	$K = 2.0$
x_1	9	7.7	6.3	6.1
x_2	1	2.3	3.7	3.9
x_3	10	10	10	10
x_4	5	5	5	5
f_1	35	35	35	35
f_{21}	27	29.6	32.4	32.8
f_{22}	45	45	45	45
f_{23}	35	35	35	35

Note: (1) Learning rate parameter $\mu = 0.01$ here.
(2) Alternative optimal solutions exist in this example.

$$\max f_1 = x_1 + x_2 + 2x_3 + x4,$$
$$\text{s.t.} \quad \mathbf{x} \in \mathbf{Z}.$$

$$\begin{aligned}
w_1 &\le (1-\eta_1)M, & 3x_1 + 3x_2 - 30 &\le M\eta_1,\\
w_2 &\le (1-\eta_2)M, & 2x_1 + x_2 - 20 &\le M\eta_2,\\
w_3 &\le (1-\eta_3)M, & (x_3 - 10) &\le M\eta_3,\\
w_4 &\le (1-\eta_4)M, & (x_2 + x_4 - 15) &\le M\eta_4,\\
w_5 &\le (1-\eta_5)M, & (x_4 - 10) &\le M\eta_5,\\
w_6 &\le (1-\eta_6)M, & (x_1 + 2x_2 + 2x_3 + x_4 - 40) &\le M\eta_6,
\end{aligned}$$

$$3w_1 + w_2 + 2w_6 = 3, \qquad w_3 + w_4 + 2w_6 = 3,$$
$$w_4 + w_5 + w_6 = 3,$$
$$\eta_1, \eta_2, \eta_3, \eta_4, \eta_5, \eta_6 \in \{0, 1\},$$
$$w_1, w_2, w_3, w_4, w_5, w_6 \ge 0.$$

Following the same procedure of Example 2, the above expression can be first reformulated as an unconstrained optimization problem, and then solved by using the MATLAB [29] software. With $M = 1000$, $K = 0.5$, and $\mu = 0.01$, the solution obtained is $(x_1^*, x_2^*, x_3^*, x_4^*) = (7.7, 2.3, 10, 5)$ with objective $(f_1^*, f_{21}^*, f_{22}^*, f_{23}^*) = (35, 29.6, 45, 35)$. Using the traditional approach, the optimal solutions obtained are $(x_1^*, x_2^*, x_3^*, x_4^*) = (5, 5, 10, 5)$ and $(10, 0, 10, 5)$ with objectives $(f_1^*, f_{21}^*, f_{22}^*, f_{23}^*) = (35, 35, 45, 35)$ and $(35, 25, 55, 35)$, respectively. However, we can find that alternative optimal solutions exist, with respect to different K values, in this example.

The solution trajectory with $\mathbf{x}$ versus the number of iterations is shown in Figure 5, and the reasonable solutions with respect to different K values $(0.1 \le K \le 2.0)$ are listed in Table 2.

5. DISCUSSIONS

We have investigated MOP and MLP problems through the dynamic behavior of ANNs. In contrast to traditional approaches concentrating on soft algorithm, the ANN approach has the advantage of being implemented on an electronic circuit for real-time problem-solving. In dealing with large-size problems, the circuit designed can be realized on a VLSI which houses a friendly environment for the hardware to manage the problems more efficiently. Furthermore, the ANN approach functions the same way as the interior point method in obtaining an approximate solution, which is different from the extreme-point search approach.

Due to a highly nonlinear energy function from the transformation process of the KKT conditions, the ANN approach to MLP problems would be difficult to handle provided that the problem has more than three levels. In addition, we can see that the range of K values for the MLP problems are smaller than those for the MOP problems. In this particular regard, MLP problems are difficult to solve by using this approach.

One problem exists while applying the ANN approach, which is the estimation of the values of the parameters. Although several pioneers suggested various guidelines to estimate the desirable parameters, more rigorous investigations are expected.

Since the simulation of our approach is conducted via employing the MATLAB software on a digital computer, we did not implement the network through hardware circuit. In such a way, the time needed to get whatever solutions depends solely on the platform's speed. Hence, the exploitation of massively parallel technology using ANN to solve MOP and MLP problems for computational savings will form one of our future researches.

REFERENCES

1. J.J. Hopfield and T.W. Tank, "Neural" computation of decisions in optimization problems, *Biological Cybernetics* **52**, 141–152, (1985).
2. G. Anandalingam and T.L. Friesz, Hierarchical optimization: An introduction, *Annals of Operations Research* **34**, 1–11, (1992).
3. J.F. Bard, Coordination of a multidivisional organization through two levels of management, *Omega* **11**, 457–468, (1983).
4. G. Anandalingam, A mathematical programming model of decentralized multi-level systems, *J. of Operational Research Society* **39**, 1021–1033, (1988).
5. O. Ben-Ayed, Bilevel linear programming, *Computers and Operations Research* **20**, 485–501, (1993).
6. K. Shimizu, Y. Ishizuka and J.F. Bard, *Nondifferentiable and Two-Level Mathematical Programming*, Kluwer Academic, Boston, (1997).
7. U.P. Wen and S.T. Hsu, Linear bi-level programming problems—A review, *J. of Operational Research Society* **42**, 125–133, (1991).
8. O. Ben-Ayed and C.E. Blair, Computational difficulties of bilevel linear programming, *Operations Research* **38**, 556–560, (1990).
9. E.S. Lee and H.S. Shih, *Fuzzy and Multi-Level Decision Making: An Interactive Computational Approach*, Springer-Verlag, London, (2001).
10. W.T. Weng and U.P. Wen, A primal-dual interior point algorithm for solving bilevel programming problems, *Asia-Pacific J. of Operational Research* **17** (2), 213–231, (2000).
11. H.S. Shih, Y.J. Lai and E.S. Lee, Fuzzy approach for multi-level programming problems, *Computers and Operations Research* **23** (1), 73–91, (1996).
12. H.S. Shih and E.S. Lee, Discrete multi-level programming in a dynamic environment, In *Dynamic Aspects in Fuzzy Decision Making*, Vol. 73, Studies in Fuzziness and Soft Computing, (Edited by Y. Yoshida), pp. 79–98, Physica-Verlag, Heidelberg, (2001).
13. H.S. Shih, An interactive approach for integrated multi-level systems in a fuzzy environment, *Mathl. Comput. Modelling* **36** (4/5), 569–585, (2002).
14. K.A. Smith, Neural networks for combinatorial optimization: A review on more than a decade of research, *INFORMS Journal on Computing* **11** (1), 15–34, (1999).
15. L.I. Burke and J.P. Ignizio, Neural network and operations research: An overview, *Computers and Operations Research* **19** (3/4), 179–189, (1992).
16. K.A. Smith and J.N.D. Gupta, Neural networks in business: Techniques and applications for the operations research, *Computers and Operations Research* **27** (11/12), 1023–1044, (2000).
17. A. Cichocki, R. Unbehauen, K. Weinzierl and R. Holzel, A new neural network for solving linear programming problems, *European J. of Operational Research* **93**, 244–256, (1996).
18. W.S. McCulloch and W.A. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematics and Biophysics* **5**, 115–133, (1943).
19. I.B. Pyne, Linear programming on an electronic analog computer, *Transactions of the American Institute Electrical Engineers* **75**, 139–143, (1956).
20. B. Shirazi and S. Yih, Critical analysis of applying Hopfield neural net model to optimization problems, In *IEEE International Conference on Systems, Man and Cybernetics*, 14–17 Nov. 1989, Vol. 1, pp. 210–215, (1989).
21. A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, John Wiley, New York, (1993).
22. S.H. Zak, V. Upatising and S. Hui, Solving linear programming problems with neural networks: A comparative study, *IEEE Transactions on Neural Networks* **6** (1), 94–104, (1995).

23. Y. Xia, A new neural network for solving linear programming problems and its applications, *IEEE Transactions on Neural Networks* **7** (2), 525–529, (1996).
24. L.O. Chua and G. Lin, Nonlinear programming without computation, *IEEE Transactions on Circuits and Systems* **CAS-31** (2), 182–188, (1984).
25. C.K. Looi, Neural network methods in combinatorial optimization, *Computers and Operations Research* **19** (3/4), 191–208, (1992).
26. R. Sharda, Neural networks for the MS/OR analyst: An application bibliography, *Interfaces* **24** (2), 116–130, (1994).
27. H.C. Zhang and S.H. Huang, Applications of neural networks in manufacturing: A state-of-the-art survey, *International Journal of Production Research* **33** (3), 705–728, (1995).
28. F.M. Ham and I. Kostanic, *Principles of Neurocomputing for Science and Engineering*, McGraw-Hill, New York, (2001).
29. The MathWorks, Inc., *MATLAB (Release 11)*, Natick, MA, (2000).
30. C.C. Ku and K.Y. Lee, Diagonal recurrent neural networks for dynamic system control, *IEEE Transactions on Neural Networks* **6** (1), 144–156, (1994).
31. A. Goicoechea, D.R. Hanson and L. Duckstein, *Multiobjective Decision Analysis with Engineering and Business Application*, John Wiley, New York, (1982).
32. M. Sun, A. Stam and R.E. Steuer, Solving multiple objective programming problems using feed-forward artificial neural networks: The interactive FFANN procedure, *Management Science* **42** (6), 835–849, (1996).
33. M. Sun, A. Stam and R.E. Steuer, Interactive multiple objective programming using Tchebycheff programs and artificial neural networks, *Computers and Operations Research* **27**, 601–620, (2000).
34. M. Gen, K. Ida and R. Kobuchi, Neural network technique for fuzzy multiobjective linear programming, *Computers and Industrial Engineering* **35** (3–4), 543–546, (1998).
35. E.S. Lee and R.J. Li, Fuzzy multiple objective programming and compromise programming with Pareto optimum, *Fuzzy Sets and Systems* **53**, 275–288, (1993).
36. S. Balakrishnan, P.S. Kannan, C. Aravindan and P. Subathra, On-line emission and economic load dispatch using adaptive Hopfield neural network, *Applied Soft Computing* **2**, 297–305, (2003).
37. C.L. Hwang and A.S.M. Masud, *Multiple Objectives Decision Making: Methods and Applications*, Springer-Verlag, Berlin, (1979).
38. C.L. Hwang, S.R. Paidy, K. Yoon and A.S.M. Masud, Mathematical programming with multiple objectives: A tutorial, *Computers and Operations Research* **7**, 5–31, (1980).
39. Y.J. Lai, TOPSIS for MODM, *European J. of Operational Research* **76**, 486–500, (1994).
40. Lindo Systems, Inc., *Lingo (Extended 6.0), Premier Modeling Tool*, Chicago, (2000).
41. J. Fortuny-Amat and B. McCarl, A representation and economic interpretation of a two-level programming problem, *J. of Operational Research Society* **32**, 783–792, (1981).