# Double-Layered Hybrid Neural Network Approach for Solving Mixed Integer Quadratic Bilevel Problems

Shamshul Bahar Yaakob and Junzo Watada

**Abstract.** In this paper we build a double-layered hybrid neural network method to solve mixed integer quadratic bilevel programming problems. Bilevel programming problems arise when one optimization problem, the upper problem, is constrained by another optimization, the lower problem. In this paper, mixed integer quadratic bilevel programming problem is transformed into a double-layered hybrid neural network. We propose an efficient method for solving bilevel programming problems which employs a double-layered hybrid neural network. A two-layered neural network is formulate by comprising a Hopfield network, genetic algorithm, and a Boltzmann machine in order to effectively and efficiently select the limited number of units from those available. The Hopfield network and genetic algorithm are employed in the upper layer to select the limited number of units, and the Boltzmann machine is employed in the lower layer to decide the optimal solution/units from the limited number of units selected by the upper layer.The proposed method leads the mixed integer quadratic bilevel programming problem to a global optimal solution. To illustrate this approach, several numerical examples are solved and compared.

## 1 Introduction

Bilevel programming has increasingly been addressed in the literature, both from the theoretical and computational points of view [1, 2]. This bilevel programming model

Shamshul Bahar Yaakob
Graduate School of IPS, Waseda University, 2-7 Hibikino, Wakamatsu,
Kitakyushu 808-0135, Fukuoka, Japan
and
School of Electrical Systems Engineering, Universiti Malaysia Perlis,
02600 Jejawi,Perlis, Malaysia
e-mail: `shamshul@fuji.waseda.jp`

Junzo Watada
Graduate School of IPS, Waseda University, 2-7 Hibikino, Wakamatsu,
Kitakyushu 808-0135, Fukuoka, Japan
e-mail: `junzow@osb.att.ne.jp`

has been widely applied to decentralized planning problems involving a decision progress with a hierarchical organization. It is characterized by the existence of two optimization problems in which the constraint region of the upper-level problem is implicitly determined by another optimization problem. The bilevel programming problem is hard to solve. In fact, the problem has been proved to be NP-hard [3].

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 [4]. The properties of bilevel programming problems are summarized as follows:

i. an interactive decision-making unit exist within a predominantly hierarchical structure;

ii. the 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 decision maker's (DM's) problem can be reflected in both the objective function and the set of feasible decision space.

The basic concept of the bilevel programming 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 [5]. This decision-making process is extremely useful to such decentralized systems as agriculture, government policy, economic systems finance, power systems, transportation, and network designs, and is particularly suitable for conflict resolution [6, 7, 8].

A conventional solution approach to the bilevel programming problem is to transform the original two level problems into a single level one by replacing the lower level optimization problem with its Kuhn-Tucker optimization conditions. Branch-and-bound method [9, 10, 11], descent algorithms [12, 13], and evolutionary method [14, 15, 16] have been proposed for solving the bilevel programming problems based on this reformulation. Compared with classical optimization approaches, the prominent advantage of neural computing is that it can converge to the equilibrium point (optimal solutin) rapidly, and this advantage has been attracting researches to solve bilevel programming problem using neural network approach. Shih [17] and Lan [18] recently proposed neural network for solving the linear bilevel programming problem. But it deserves pointing out that there are no reports on solving mixed integer quadratic bilevel programming problem using neural network approach.

In this study, we apply structural learning to a Boltzmann machine (BM). The Hopfield network is an interconnected neural network originally proposed by J.J. Hopfield in 1982 [19]. Now the Hopfield neural network used to easily terminate at a local minimum of the describing energy function. The BM [20] is likewise an interconnected neural network, which improves Hopfield network performance by using probabilities to update both the state of a neuron and its energy function, such that the latter rarely falls into a local minimum.

We formulate a two-layered neural network comprising a Hopfield network, genetic algorithm, and a BM in order to effectively and efficiently select a limited number of units from those available. The Hopfield network and genetic algorithm are employed in the upper layer to select the limited number of units, and the BM is employed in the lower layer to decide the optimal solution/units from the limited number of units selected by the upper layer. The double-layered hybrid neural network, whose two layers connect corresponding units in the upper and lower machines, constitutes an effective problem solving method.

The remainder of the paper is organized as follows. Section 2 contains an introduction to bilevel programming problems. Sections 3 and 4 explain the double-layered BM for the solving mixed integer quadratic programming problem. Results of numerical examples are reported in Section 5. Finally, Section 6 concludes the paper.

## 2 Bilevel Programming Problems

Bilevel programming is a case of multilevel mathematical programming that is defined to solve decentralized planning problems with multiple decision makers in a multilevel or hierarchical organization [14]. Among different levels, decision makers play a game called the Stackelberg game [15], in which the follower responds to any decision made by the leader but is not controlled directly by the leader. Thus the leader is able to adjust the performance of the overall multilevel system indirectly by his decisions. Bilevel programming involves two optimization problems where the constraint region of the first-level problem is implicitly determined by the other second-level optimization problem. Bilevel programming problem, where a top level DM has control over the vector $x_1$ while a bottom level DM controls the vector $x_2$. Letting the performance functions of $F(x_1,x_2)$ and $f(x_1,x_2)$ for the two planners be linear and bounded, then the bilevel programming problem can be represented as

$$\max_{x_1} F(x_1,x_2) = c_{11}x_1 + c_{12}x_2 \quad \text{(upper level)} \tag{1}$$

where $x_2$ solves

$$\max_{x_2} f(x_1,x_2) = c_{21}x_1 + c_{22}x_2 \quad \text{(lower level)}$$

subject to $(x_1,x_2) \in X = \{(x_1,x_2) \mid A_1x_1 + A_2x_2 \leq b, x_1, x_2 \geq 0\}$, where $c_{11}, c_{12}, c_{21}$, $c_{22}$ and $b$ are vectors, $A_1$ and $A_2$ are matrices, and $X$ represents the two-dimensional constraint region.

Compared with multi-objective programming, bilevel programming is able to overcome the aforementioned limitations. A bilevel model provides an interactive platform for both the upper-and-lower level problems. The objectives of both levels can also reach their utmost simultaneously. However, bilevel programming problems are generally difficult to solve because the lower level actually serves as a nonlinear constraint and the whole problem is intrinsically a non-convex programming problem. Bilevel programming is workable only if efficient algorithm are available for large actual cases. A number of attempts have been made to develop efficient algorithms for the bilevel programming problem, such as the iterative optimization assignment algorithm by Asakura and Sasaki [20] and the sensitivity-analysis-based (SAB) algorithm by Yanf et. al [21]. In this study, a neural network (NN)-based method is employed to resolve the bilevel programming problems.

## 3 Hopfield and Boltzmann Machine

The Hopfield network is a fully connected, recurrent neural network, which uses a form of the generalized Hebb rule to store Boolean vectors in its memory. Each unit(neuron)-$n$ has a state value denoted by $s_n$, In any situation, combining the state of all units leads to a global state for the network. For example, let us consider a network comprising three units $s_1, s_2$ and $s_3$. The global state at time step t is denoted by a vector s, whose elements are $s_1, s_2$ and $s_3$. When a user presents the network with an input, the network will retrieve the item in its memory which most closely resembles that particular input.

In general, the Hopfield network operates by taking an input, evaluating the output (in other words the global status $s$). This global state is the input, providing it works correctly, together with other prototypes, which are stored in the weight matrix by Hebb's postulate, formulated as

$$w_{ij} = \frac{1}{N} \sum_p X_{i_p} X_{j_p} \tag{2}$$

where, $p = 1 \cdots N, w_{ij}$ is the weight of the connection from neuron $j$ to neuron $i, N$ is the dimension of the vector, $p$ the number of training patterns, and $X_{i_p}$ the $p$th input for the neuron $i$. In other words, using Hebb's postulate, we create the weight matrix, which stores the entire prototype that we want the network to remember. Because of these features, it is sometimes referred to as an ”Auto-associative Memory”. However, it is worth noting that the maximum number of prototypes that a Hopfield network can store is only 0.15 times the total number of units in the network [19].

One application of the Hopfield network is to use it as an energy minimizer. This application comes to life because of the ability of Hopfield networks to minimize an energy function during its operation. The simplest form of energy function is given by the following:

$$E = \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ij} s_i s_j \tag{3}$$

Here $w_{ij}$ denotes the strength of the influence of neuron $j$ on neuron $i$. The $w_{ij}$ are created using Hebb's postulate as mentioned above, and they belong to a symmetric matrix with the main diagonal line containing only zeroes (which means there are no self-feedback connections). Because of this useful property, the Hopfield network can also be used to solve combinatorial optimization problems. However, Hopfield networks suffer from a major disadvantage in that they sometimes converge to a local rather than to the global minima, which usually happens when dealing with noisy inputs. In order to straigten out this problem, a modification was made to the BM.

The BM is an interconnected neural network, and is a modification of the Hopfield network which helps it to escape from local minima. The main idea is to employ simulated annealing, a technique derived from the metallurgy industry. It works by first relaxing all the unites (in other words, causing them to freely move by applying sufficient ”heat”). After that, the temperature is gradually decreased. During this process, the unites will move at lower and lower speed until they become fixed and form a new structure as the temperature decreases.

Simulated annealing is an optimization technique. In Hopfield networks, local minima are used in a positive way, but in optimization problems, local minima get in the way; one must have a way of escaping from them. When optimizing a very large and complex system (i.e., a system with many degrees-of-freedom), instead of ”always” going downhill, we try to go downhill ”most of the time”. Initially, the probability of not going downhill should be relatively high (”high temperature”), but as time (iterations) go on, this probability should decrease (with the temperature decreasing according to an annealing schedule).

Now the convergence time of a BM is usually extremely long. According to the ”annealing schedule”, if $T_0$ is very large, then a strategy is pursued whereby neurons are flipping on and off at random, totally ignoring incoming information. If $T_0$ is close to zero, the network behaves ”deterministically”, i.e. like a network of McCulloch-Pitts neurons.

Although the way in which a BM works is similar to a Hopfield network, we cannot use Hebb’s postulate to create the weight matrix representing the correlations between units. Instead, we have to use a training (learning) algorithm - one based on the Metropolis algorithm.

The BM can be seen as a stochastic, generative counterpart of the Hopfield network. In the BM, probability rules are employed to update the state of neurons and the energy function as follows:

If $V_i(t+1)$ is the output of neuron $i$, in the subsequent time iteration $t+1$, $V_i(t+1)$ is 1 with probability $P$, and $V_i(t+1)$ is 0 with probability $1-P$, where

$$P[V_i(t+1)] = f\left(\frac{u_(t)}{T}\right). \tag{4}$$

Here, $f(\cdot)$ is a sigmoid function, T is a network temperature, and $u_i(t)$ is the total input to neuron $i$ shown in equation (4), which is given by

$$u_i(t) = \sum_{j=1} w_{ij} V_i(t) + \theta_i \tag{5}$$

where, $w_{ij}$ is the weight between neurons $i$ and $j$, $\theta_i$ is the threshold of neuron $i$, and $V_i$ is the state of unit $i$. The energy function, $E$, proposed by Hopfield, is written as:

$$E(t) = \frac{1}{2}\sum_{i,j=1} w_{ij}V_i(t)V_j(t) - \sum_{i=1}^{n} \theta_i V_i(t) \tag{6}$$

Hopfield has shown that this energy function simply decreases with learning [19]. There is the possibility that this energy function converges to a local minimum. However, in the case of the BM, the energy function can increase with minute probability. Therefore, the energy function will be unlikely to fall into a local minimum. Thus, the combination of Hopfield network and BM offers a solution to overcome the problem of finding the optimal number of units in the neural network. Accordingly, this study proposes a double-layered BM which we discuss in detail in the Section 4.

## 4 Double-Layered Hybrid Neural Network

Conventionally, the number of units is decided on the basis of expert experience. In order to solve this problem, we formulate a double-layered neural network consisting of both Hopfield and Boltzmann neural networks and hybrid with genetic algorithm at the upper layer. This double-layered model can be employed to select are units from those available. The double-layered model has two layers - referred to as the upper and lower layers, respectively. The functions of the layers are as follows:

1. Upper layer (Hopfield neural network and genetic algorithm) is used to select are units from the total. This hybrid layer is called a “supervising layer”.
2. Lower layer (BM) is used to decide the optimal units from the selected units in the upper layer. This Boltzmann layer is called an “executing layer”.

This double-layered hybrid neural network is a new type of neural network model which deletes units (neurons) in the lower layer that are not selected in the upper layer during execution. The lower layer is then restructured using the selected units. Because of this feature, the double-layered hybrid neural network converges more efficiently than a conventional BM. This is an efficient method for solving a selection problem by transforming its objective function into the energy function, since the Hopfield and Boltzmann networks converge at the minimum point of the energy function.

The double-layered hybrid neural network just described converts the objective function into energy functions of two components - namely the upper layer (Hopfield network and genetic algorithm) $E_u$ and the lower layer (BM) $E_l$. The double-layered BM is tuned such that the upper layer influences the lower layer with probability 0.9, and the lower layer influences the upper layer with probability 0.1. The main reason for selecting these probabilities is based on trial and error, it is found that the selected probabilities provide the best solution to our proposed method. Thus the double-layered hybrid neural network is iterated with

$$Y_i = 0.9y_i + 0.1x_i$$

for the upper layer, and

$$X_i = x_i(0.9y_i + 0.1)$$

for the lower layer. Here $Y_i$ in the upper layer is a value transferred to the corresponding nodes in the upper layer, $X_i$ in the lower layer is a value transferred to corresponding nodes in the lower layer, $y_i$ is the value of the present state at node $i$ in the upper layer, and $x_i$ is the value of the present state at node $i$ in the lower layer, respectively. $X_i$ means that the value is influenced to the tune of 90% from the value of node $i$ in the upper layer. When $Y_i$ is 1, $X_i = x_i$; otherwise, when $y_i$ is 0, 10% of the value of $x_i$ is transferred to the other nodes. On the other hand, $Y_i$ has a 10% influence on the lower layer. Therefore, even if the upper layer converges to a local minimum, the disturbance from the lower layer makes the upper layer escape from this local minimum. When the local minima possess a large barrier, dynamic behavior may be used (by changing 0.9 and 0.1 dynamically) - this phenomenon is similar to simulated annealing. The proposed algorithm of the double-layered hybrid neural network is as follow, and is shown in Figure 1 where (a) represents the processing stage of the algorithm and (b) illustrates the final stage of the proposed method:

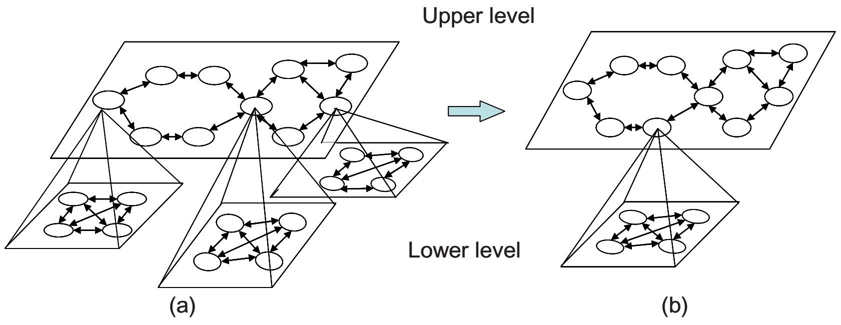

**Fig. 1** Double-layered hybrid neural network

## 5 Numerical Examples

In this section we will present two examples provided in [22] to illustrate the validity of the double-layered BM approach for the mixed integer quadratic bilevel programming.

(Proposed Algorithm)

Step 1. Set each parameter to its initial value.
Step 2. Input $K_u$ (weight for upper layer) and $K_l$ (weight for lower layer) .
Step 3. Execute the upper layer.
Step 4. If the output value of a unit in the upper layer is 1, add some amount of this value to the corresponding unit in the lower layer. Execute the lower layer.
Step 5. After executing the lower layer at a constant frequency, decrease the temperature.
Step 6. If the output value is sufficiently large, add a certain amount of the value to the corresponding unit in the upper layer.
Step 7. Iterate from Step 3 to Step 6 until the temperature reaches the restructuring temperature.
Step 8. Restructure the lower layer using selected units in the upper layer
Step 9. Execute the lower layer until reaching the termination condition.

*Example 5.1.*

$$\begin{aligned}
\min \quad & (x_1-30)^2+(x_2-20)^2-20y_1+20y_2 \\
\text{subject to } & x_1+2x_2 \le 30 \\
& x_1+x_2 \ge 30 \\
& 0 \le x_1 \le 15 \\
& 0 \le x_2 \le 15 \\
& \min \quad (x_1-y_1)^2+(x_2-y_2)^2 \\
& \text{subject to} \quad 0 \le y_1 \le 15 \\
& \qquad\qquad\quad\; 0 \le y_2 \le 15.
\end{aligned}$$

*Example 5.2.*

$$\begin{aligned}
\min \quad & y_1^2+y_2^2-x^2-4x \\
\text{subject to } & 0 \le x \le 2 \\
& \min \quad y_1^2+0.5y_2^2+y_1y_2+(1-3x)y_1+(1+x)y_2 \\
& \text{subject to} \quad 2y_1+y_2 \le 1 \\
& \qquad\qquad\quad\; y_1 \ge 0 \\
& \qquad\qquad\quad\; y_2 \ge 0.
\end{aligned}$$

Table 1 shows the comparison results by using the proposed method in this paper and the results in the references. $F$ and $f$ are the objective function value of the upper-level and lower-level programming problem, respectively.

**Table 1** Comparison of optimal solutions

| Example No. | Proposed Method | Reference |
|---|---|---|
| 1 | $(x_1, x_2, y_1, y_2) = (15, 7.511, 10, 7.514)$ | $(15, 7.501, 10, 7.501)$ |
| | $F = 330.821$ | 331.262 |
| | $f = 24$ | 25 |
| 2 | $(x, y_1, y_2) = (0.8394, 0.7521, 0)$ | $(0.8438, 0.7657, 0)$ |
| | $F = -2.1091$ | $-2.0769$ |
| | $f = -0.5742$ | $-0.5863$ |

## 6 Conclusions

A new proposal was presented to solve a mixed integer quadratic bilevel programming problems. We proposed the double-layered hybrid neural network that included upper layer which employs the genetic algorithm and Hopfield network and lower layer which employs BM. The resultant numerical outcomes of the method proposed in this paper were also compared to the results obtained in the references [22]. The proposed double-layered hybrid neural network proved to be very efficient from the computational point of view and quality of solutions. By using the same token, we could conclude that proposed method is capable to solve bilevel programming problem. Furthermore, the approach could also be applied to solve a complicated mixed integer quadratic bilevel programming problems.

## Acknowledgment

The first author would like to thank Universiti Malaysia Perlis and the Ministry of Higher Education Malaysia for a study leave in Waseda University under the SLAI-KPT scholarship.

## References

1. Bard, J.: Practical bilevel optimization: Algorithm and applications. Kluwer Academic Publishers, Dordrecht (1998)
2. Dempe, S.: Foundation of bilevel programming. Kluwer Academic Publishers, London (2002)
3. Ben-Ayed, O., Blair, O.: Computational difficulty of bilevel linear programming. Operations Research 38(3), 556–560 (1990)
4. Anandalingam, G., Friesz, T.L.: Hierarchical optimization: An introduction. Annals of Operations Research 34(1), 1–11 (1992)
5. Bard, J.F.: Coordination of a multidivisional organization through two levels of management. Omega 11(5), 457–468 (1983)
6. Carrion., M., Arroyo, J.M., Conejo, A.J.: A bilevel stochastic programming approach for retailer futures market trading. IEEE Transactions on Power Systems 24(3), 1446–1456 (2009)

7. Shimizu, K., Ishizuka, Y., Bard, J.F.: Nondifferentiable and two-level mathematical programming. Kluwer Academic, Boston (1997)
8. Colson, B., Marcotte, P., Savard, G.: Bilevel programming: A survey. International Journal of Operations Research 3(2), 87–107 (2005)
9. Bard, J.F., Moore, J.T.: A branch-and-bound algorithm for the bilevel programming problem. SIAM Journal on Scientific and Statistical Computing 11(2), 281–292 (1990)
10. Al-Khayyal, F., Horst, R., Paradalos, P.: Global optimization on concave functions subject to quadratic constraints: an application in nonlinear bilevel programming. Annals of Operations Research 34(1), 125–147 (1992)
11. Edmunds, T., Bard, J.F.: An algorithm for the mixed-integer nonlinear bilevel programming problem. Annals of Operations Research 34(1), 149–162 (1992)
12. Savard, G., Gauvin, J.: The steepest descent direction for the nonlinear bilevel programming problem. Operations Research Letters 15(5), 265–272 (1994)
13. Vicente, L., Savarg, G., Judice, J.: Descent approaches for quadratic bilevel programming. Journal of Optimization Theory and Applications 81(2), 379–399 (1994)
14. Hejazi, S.R., Memariani, A., Jahanshanloo, G., Sepehri, M.M.: Bilevel programming solution by genetic algorithms. In: Proceeding of the First National Industrial Engineering Conference (2001)
15. Wu, C.P.: Hybrid technique for global optimization of hierarchical systems. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernatics, vol. 3, pp. 1706–1711 (1996)
16. Yin, Y.: Genetic algorithms based approach for bilevel programming models. Journal of Transportion Engineering 126(2), 115–120 (2000)
17. Shih, H.S., Wen, U.P.: A neural network approach for multi-objective and multilevel programming problems. Journal of Computer and Mathematics with Applications 48(1-2), 95–108 (2004)
18. Lan, K.M., Wen, U.P.: A hybrid neural network approach to bilevel programming problems. Applied Mathematics Letters 20(8), 880–884 (2007)
19. Markowitz, H.: Mean-variance analysis in portfolio choice and capital markets. Blackwell, Malden (1987)
20. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Bolzmann machine. Cognitive Science 9(1), 147–169 (1985)
21. Watada, J., Oda, K.: Formulation of a two-layered Boltzmann machine for portfolio selection. International Journal Fuzzy Systems 2(1), 39–44 (2000)
22. Muu, L.D., Quy, N.V.: A global optimization method for solving convex quadratic bilevel programming problems. Journal of Global Optimization 26(2), 199–219 (2003)