

Application of Discrete Hopfield-type Neural Network for Max-Cut Problems

Ling-Yun Wu Xiang-Sun Zhang Ju-Liang Zhang

Academy of Mathematics and System Sciences
 Chinese Academy of Sciences
 Beijing 100080, P.R.China
 wlyun@amath8.amt.ac.cn

Abstract *In this paper, we discuss the convergence property of the discrete Hopfield-type neural network (DHNN) running in asynchronous mode. Then a DHNN with negative diagonal weight matrix is designed to solve the Max-Cut problem, which can approach good solutions.*

Keywords discrete Hopfield-type neural network, Max-Cut, combinatory optimization.

1 Introduction

There have been more than ten years since artificial neural networks were applied in the field of optimization. As we known, the application of neural networks in the field of optimization was initiated by Hopfield and Tank in 1985. Hopfield and Tank's seminal paper [13] demonstrated that the Travelling Salesman Problem (TSP) could be solved by using a Hopfield neural network. Since then, Hopfield neural network have always been the majority neural network in solving optimization problems. A variety of feedback neural networks similar to Hopfield neural network have been proposed to solve linear programming problems (for example, [21, 22, 25]) and quadratic programming problems (for example, [3, 18, 25]) as well as combinatorial optimization problems (for example, [1, 8, 13, 14, 16, 23]) because of the potential of extremely rapid computation power and speed of neural networks, which can be obtained through hardware implementation. Cichocki et al. [6] and Zhang [24] are two comprehensive books about the neural networks and optimization, and a concise review of neural networks for combinatorial optimization was written by Smith [20].

The discrete Hopfield-type neural network (DHNN) denoted by $\mathcal{N} = (\mathbf{W}, \mathbf{T})$ can be described

by the following equation:

$$\mathbf{x}(t+1) = \mathbf{sgn}\{(\mathbf{W}\mathbf{x}(t) - \mathbf{T})\},$$

where $\mathbf{W} = (w_{ij})$ is an $n \times n$ transition matrix (or weight matrix), $\mathbf{T} = (t_1, \dots, t_n)^T$ is a threshold vector, and

$$\mathbf{sgn}\{\mathbf{x}\} = (\text{sgn}(x_1), \dots, \text{sgn}(x_n))^T$$

where the operator sgn is the signum function (or bipolar binary function) defined by

$$\text{sgn}(y) = \begin{cases} 1 & \text{if } y \geq 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes, a variation of the signum function is used in DHNN, called unipolar binary function:

$$\text{sgn}(y) = \begin{cases} 1 & \text{if } y \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

There are two categories of DHNN identified by their operating modes: synchronous mode (parallel mode) and asynchronous mode (serial mode). The general algorithms of the DHNN in the two modes are as follows,

Algorithm 1. (*Synchronous DHNN*)

1. given a stop criterion \mathcal{C}
2. given an initial vector $\mathbf{x}(0)$, $t = 0$
3. $x_i(t+1) = \text{sgn}(\mathbf{W}_i\mathbf{x}(t) - t_i) \quad i = 1, \dots, n$
4. if $\mathbf{x}(t+1)$ satisfies \mathcal{C} then stop
 else $t := t + 1$ and go to step 3

Algorithm 2. (*Asynchronous DHNN*)

1. given a stop criterion \mathcal{C} and a point-to-point map

$$M : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$$

2. given an initial vector $\mathbf{x}(0)$ and set $i(0) = 1, t = 0$
3. take $i(t) = M(i(t-1)) \in \{1, \dots, n\}$ if $t > 0$
4. $x_i(t+1) = \begin{cases} \text{sgn}(\mathbf{W}_i \mathbf{x}(t) - t_i) & i = i(t) \\ x_i(t) & i \neq i(t) \end{cases}$
5. if $\mathbf{x}(t+1)$ satisfies \mathcal{C} then stop
else $t := t+1$ and go to step 3

where $\mathbf{W}_i, i = 1, \dots, n$ are the rows of \mathbf{W} .

In this paper, we only consider the DHNN running in asynchronous mode.

2 Theoretical Results

Definition 1. A DHNN $\mathcal{N} = (\mathbf{W}, \mathbf{T})$ is called convergent from a given initial state $\mathbf{x}(0)$ if there exists an integer k such that $\mathbf{x}(k) = \text{sgn}\{\mathbf{W}\mathbf{x}(k) - \mathbf{T}\}$. $\mathbf{x}^* = \mathbf{x}(k)$ is referred to as a stable state. A network is called convergent if it converges to a stable state from any given initial state. A DHNN is called convergent to a stable cycle from a given initial state $\mathbf{x}(0)$ if there exists integers k and q such that $\mathbf{x}(k+q) = \text{sgn}\{\mathbf{W}\mathbf{x}(k) - \mathbf{T}\}$.

The following theorem is given by Hopfield in [12].

Theorem 1. ([12]) Let \mathbf{W} be a symmetric matrix with nonnegative diagonal. The DHNN $\mathcal{N} = (\mathbf{W}, \mathbf{T})$, which runs in the asynchronous mode, will always converge to a stable state.

The asynchronous DHNN can be used to solve such combinatorial optimization problem

$$(\mathcal{P}) \quad \begin{array}{ll} \min & -\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{T}^T \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in \{-1, 1\}^n. \end{array}$$

There were few applications of DHNN in combinatorial optimization before Zhuo et al. [26]. The main reason is that the weight matrix \mathbf{W} is not always satisfied the condition in Theorem 1, therefore the network does not guarantee to be convergent. Zhuo et al. [26] proposed a DHNN as a solver for Four-Coloring Map problem by rearranging the weight matrix to zero diagonal.

It is interesting to notice that rearranging w_{ii} will not affect the solution set of the problem \mathcal{P} . In fact,

$$\begin{aligned} & -\frac{1}{2} \mathbf{x}^T (\mathbf{W} + \text{Diag}(\Delta w_{ii})) \mathbf{x} + \mathbf{T}^T \mathbf{x} \\ = & -\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{T}^T \mathbf{x} + \sum_{i=1}^n \Delta w_{ii} x_i^2 \\ = & -\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x} + \mathbf{T}^T \mathbf{x} + \sum_{i=1}^n \Delta w_{ii} \end{aligned}$$

where $\sum_{i=1}^n \Delta w_{ii}$ is a constant which does not contribute to the solution set. But according to the above theorem, the diagonal elements of the matrix \mathbf{W} take very important role in the convergence analysis. So we can simply assign proper values to the diagonal elements of \mathbf{W} in order to make the network convergent.

On the other hand, the values of the diagonal elements are closely related to the number of stable states of the network. The following theorem tells us that smaller diagonal elements may result in less stable states of a network.

Theorem 2. ([15], also [24]) Let $\mathcal{N}^1 = (\mathbf{W}^1, \mathbf{T})$, $\mathcal{N}^2 = (\mathbf{W}^2, \mathbf{T})$ be two DHNN, where $w_{ij}^1 = w_{ij}^2$ for $i \neq j$, and $w_{ii}^1 \leq w_{ii}^2$ for $i = 1, \dots, n$. Then $\Omega_{\mathcal{N}^1} \subseteq \Omega_{\mathcal{N}^2}$ where $\Omega_{\mathcal{N}^i}$ is the set of all stable state of the network $\mathcal{N}^i (i = 1, 2)$ respectively.

Generally speaking, since there is a finite number of states for a network, the network will always converge to a stable state or a stable cycle. However, while a network is used to solve a optimization problem, we are concerned about whether the energy function is decreasing as the network is updating rather than whether the network converge to a stable state. Thus the following definition are proposed.

Definition 2. For a given energy function $E(t)$, a network \mathcal{N} is called E-convergent form a given initial state $\mathbf{x}(0)$ if it converges to a stable state or a stable cycle with $\Delta E(t) \leq 0$ for $t = 0, 1, \dots$. A network is called totally E-convergent if it is convergent from any given initial state.

By the above definition, we have the following theorem.

Theorem 3. Let $\mathcal{N} = (\mathbf{W}, \mathbf{T})$ be a discrete Hopfield-type neural network running in asynchronous mode with symmetric matrix \mathbf{W} . Let

$$\begin{aligned} K_i &= \min\{|\sum_{j \neq i} w_{ij} x_j - t_i| : \\ & \sum_{j \neq i} w_{ij} x_j - t_i \neq 0, \\ & \mathbf{x} = (x_1, \dots, x_n)^T \in \{-1, 1\}^n\}. \end{aligned}$$

If

$$w_{ii} > -K_i, i = 1, \dots, n$$

then the network is totally E-convergent with the energy function

$$E(t) = -\frac{1}{2} \mathbf{x}(t)^T \mathbf{W} \mathbf{x}(t) + \mathbf{T}^T \mathbf{x}(t).$$

Proof. Let $\mathbf{x}^t = \mathbf{x}(t)$ and $\mathbf{x}^{t+1} = \mathbf{x}(t+1)$ be network states in step t and $t+1$ respectively and $\mathbf{x}^{t+1} \neq \mathbf{x}^t$. Note that the network is running in the asynchronous mode, we can suppose that $\mathbf{x}^{t+1} = \mathbf{x}^t + \Delta\mathbf{x}$ with $\Delta\mathbf{x}_k = -2x_k^t$ and $\Delta\mathbf{x}_i = 0, i \neq k$.

Consider the energy function

$$E(t) = -\frac{1}{2}\mathbf{x}(t)^T \mathbf{W} \mathbf{x}(t) + \mathbf{T}^T \mathbf{x}(t),$$

we have

$$\begin{aligned} \Delta E(t) &= E(\mathbf{x}^{t+1}) - E(\mathbf{x}^t) \\ &= \left[-\frac{1}{2}(\mathbf{x}^t + \Delta\mathbf{x})^T \mathbf{W} (\mathbf{x}^t + \Delta\mathbf{x})\right. \\ &\quad \left.+ \mathbf{T}^T (\mathbf{x}^t + \Delta\mathbf{x})\right] \\ &\quad - \left[-\frac{1}{2}(\mathbf{x}^t)^T \mathbf{W} \mathbf{x}^t + \mathbf{T}^T \mathbf{x}^t\right] \\ &= -\Delta\mathbf{x}^T \mathbf{W} \mathbf{x}^t - \frac{1}{2}\Delta\mathbf{x}^T \mathbf{W} \Delta\mathbf{x} + \mathbf{T}^T \Delta\mathbf{x} \\ &= 2x_k^t \left(\sum_{j=1}^n w_{kj} x_j^t\right) - 2w_{kk}(x_k^t)^2 - 2x_k^t t_k \\ &= 2x_k^t \left(\sum_{j \neq k} w_{kj} x_j^t - t_k\right). \end{aligned}$$

If we can show that $\Delta E(t) \leq 0$, the theorem is proved.

For the case that $x_k^t = 1, x_k^{t+1} = -1$, we have

$$\text{sgn}\left(\sum_{j=1}^n w_{kj} x_j^t - t_k\right) = -1,$$

i.e.,

$$\sum_{j=1}^n w_{kj} x_j^t - t_k < 0,$$

thus

$$\sum_{j \neq k} w_{kj} x_j^t - t_k < -w_{kk}.$$

Since $w_{kk} > -K_k$, then

$$\sum_{j \neq k} w_{kj} x_j^t - t_k < K_k$$

which implies that

$$\sum_{j \neq k} w_{kj} x_j^t - t_k \leq 0.$$

So,

$$\Delta E(t) \leq 0.$$

For the case of $x_k^t = -1, x_k^{t+1} = 1$, it can be proved similarly. \square

Since we are interested in the global solution of a given optimization problem, it is not favorable if there is no stable state corresponding to a global solution. The following theorem give a guarantee.

Theorem 4. (*[26], also [24]*) *If a DHNN $\mathcal{N} = (\mathbf{W}, \mathbf{T})$ is E-convergent with energy function*

$$E(t) = -\frac{1}{2}\mathbf{x}(t)^T \mathbf{W} \mathbf{x}(t) + \mathbf{T}^T \mathbf{x}$$

then there is at least one stable state (or states in a stable cycle) which corresponds to a global solution of the problem \mathcal{P} .

3 Application to The Max-Cut Problem

Consider an undirected, simple graph (i.e., a graph with no loops or parallel edges) $G = (V, E)$, where $V = \{1, \dots, n\}$ is the vertex set of G and E the edge set of G . An edge $ij \in E$ connects vertices i and j . Let w_{ij} be the weight of edge ij , and $\mathbf{W} = (w_{ij}) \in \mathbb{R}^{n \times n}$ which is called weight matrix.

For $S \subseteq V$, the set $\delta(S) = \{ij \in E : i \in S, j \in V \setminus S\}$ is called the cut determined by S . The Max-Cut problem on G is to find $S \subseteq V$ such that

$$w(\delta(S)) \triangleq \sum_{ij \in \delta(S)} w_{ij} \quad (1)$$

is maximized. We refer to $w(\delta(S))$ as the weight of the cut $\delta(S)$.

It is well known that the Max-Cut problem is NP-complete. In fact it was one of the six basic problems which appeared in the list of Karp [17]. The problem is very well studied in the literature and numerous results on lower bounds and expected size for different classes of graphs are known. A large number of applications of the Max-Cut problem can be found in the literature of Physics and VLSI-design. We refer the interested reader to [7, 19] and the references therein for more details about the Max-Cut problem and its applications.

Since the Max-Cut problem is NP-complete for which the exact solution is difficult to obtain, different heuristic or approximation algorithms have been proposed, for example, algorithms based on the SDP relaxation in Goemans et al. [10], Helmberg et al. [11], Benson et al. [2] and Choi et al. [5], and heuristic algorithm in Burer et al. [4].

Funabiki et al. [9] proposed a binary neural network for the Max-Cut problem. The network they used is a continuous time neural network while our model in this paper is a discrete time network. The

energy function they used is quite different from that of us. Since we can not get their test problems, we never compare our network with their model in this paper.

The Max-Cut problem

$$\max_{S \subseteq V} \sum_{ij \in \delta(S)} w_{ij}.$$

can be formulated as an integer quadratic program by introducing cut vector. A cut vector $\mathbf{x} \in \{-1, 1\}^n$ is defined as:

$$x_i = \begin{cases} 1 & i \in S \\ -1 & i \in V \setminus S. \end{cases}$$

Then the Max-Cut problem is equivalent to the following problem,

$$\max_{x \in \{-1, 1\}^n} \sum_{i < j} w_{ij} \frac{1 - x_i x_j}{2}.$$

By exploiting the symmetry of $\mathbf{W} = (w_{ij})$ and $x_i x_i = 1$, we have

$$\begin{aligned} & \sum_{i < j} w_{ij} \frac{1 - x_i x_j}{2} \\ &= \frac{1}{4} \sum_{i, j} w_{ij} (1 - x_i x_j) \\ &= \frac{1}{4} \sum_{i=1}^n \left(\sum_{j=1}^n w_{ij} x_i x_i - \sum_{j=1}^n w_{ij} x_i x_j \right) \\ &= \frac{1}{4} \mathbf{x}^T \mathbf{W}' \mathbf{x} \end{aligned}$$

where $\mathbf{W}' = (w'_{ij})$, and

$$\begin{aligned} w'_{ij} &= -w_{ij}, i \neq j \\ w'_{ii} &= \sum_{j=1}^n w_{ij} - w_{ii} = \sum_{j \neq i} w_{ij}. \end{aligned}$$

For $\hat{\mathbf{W}} = \frac{1}{2} \mathbf{W}'$, the Max-Cut problem can be rewritten in the following more general form,

$$\max_{x \in \{-1, 1\}^n} \frac{1}{2} \mathbf{x}^T \hat{\mathbf{W}} \mathbf{x},$$

and therefore equivalent to

$$\min_{x \in \{-1, 1\}^n} -\frac{1}{2} \mathbf{x}^T \hat{\mathbf{W}} \mathbf{x}$$

which can be solved by a DHNN $\mathcal{N} = (\hat{\mathbf{W}}, \mathbf{0})$ running in asynchronous mode.

4 Computational Results

As the discussion in the previous section, the diagonal elements of the matrix $\hat{\mathbf{W}}$ must be rearranged in order to ensure that the network is E-convergent. Zhuo et al. [26] used zero diagonal weight matrix network. By our new result, negative diagonal can be used instead of zero diagonal. According to the Theorem 2, smaller diagonal elements may result in less stable states of a network, then reduce the possibility to be trapped in a state corresponding to local minimum or poor quality solution and may approach to a better solution.

The set of test problems we used are the same as that Helmberg et al. [11], Benson et al. [2], Choi et al. [5], Burer et al. [4] used to test their methods of solving the Max-Cut problem. All graphs are generated with a platform independent graph generator, **rudy**, which was written by Giovanni Rinaldi. In these test problems, the weights of edges are all integers. Therefore, we have

$$K_i \geq 0.5, \quad i = 1, \dots, n$$

Then we can reassign the diagonal elements of matrix $\hat{\mathbf{W}}$ to a negative value d satisfied $-0.5 < d < 0$.

The results are reported in Table 1 and Table 2 for using different diagonal values d . The sizes of the graphs are given as $(|V|, |E|)$, $|V|$ is the number of the vertex and $|E|$ is the number of the non-zero weights. Table 1 show the best results in 10 tests with random initial points, and Table 2 are the average values of all results in 10 tests. It is obviously that using negative diagonal improve the solution quality dramatically in both best and average values.

Graph	Size	$d = 0$	$d = -0.4$
G1	(800,19176)	11420	11497
G11	(800,1600)	470	556
G22	(2000,19990)	12963	13109
G32	(2000,4000)	1150	1372

Table 1: the best value in 10 tests

Graph	Size	$d = 0$	$d = -0.4$
G1	(800,19176)	11382.7	11444
G11	(800,1600)	457.2	549.4
G22	(2000,19990)	12901.2	13058.5
G32	(2000,4000)	1134.2	1360.4

Table 2: the average value in 10 tests

In Table 3, the results are compared with the results of DSDP method [5], one of the latest algorithms

by solving the Max-Cut SDP relaxation, and CirCut method [4], a new heuristic algorithm for the Max-Cut problem. To our knowledge, DSDP and CirCut are two of the most efficient algorithms for the Max-Cut problem at present. The results in Table 3 are the best values obtained in 100 tests with random initial points. The quality of solutions given by DHNN is almost better than those of DSDP except on problems G48, G49 and G50. On the problem G49, all three algorithms obtained the global optimal value. Moreover, on problems G11, G13, G33, G55 and G62, the solution found by DHNN are best. Note that our code is simple without any further improving technique and CirCut is a well developed heuristic code, the quality of the solutions obtained by DHNN is competitive.

Graph	DSDP	CirCut	DHNN
G11	542	554	560
G12	540	552	548
G13	564	572	574
G14	2922	3053	3024
G15	2938	3039	3013
G20	838	939	895
G21	841	921	881
G22	12960	13331	13167
G23	13006	13269	13157
G24	12933	13287	13140
G30	3038	3377	3200
G31	2851	3255	3111
G32	1338	1380	1378
G33	1330	1352	1354
G34	1334	1358	1356
G48	6000	6000	5992
G49	6000	6000	6000
G50	5880	5856	5846
G55	9960	10240	11639
G56	3634	3943	3700
G57	3320	3412	3394
G60	13610	14081	13718
G61	5252	5690	5316
G62	4612	4740	4894
G64	7624	8575	8055

Table 3: Comparison with DSDP and CirCut

5 Conclusion

In this paper, we give new convergence conditions for discrete Hopfield-type neural network. According to this condition, we construct a discrete Hopfield-type neural network (DHNN) with negative diago-

nal weight matrix to solve Max-Cut problem. Elementary numerical experiments show that the performance of such DHNN is exciting. Comparing with other methods solving Max-Cut, for example SDP method and CirCut method which are two of the most efficient solving Max-Cut problem at present, the solution quality of DHNN is competitive or even better than those of them. Meanwhile, we note that although we simply change the diagonal elements to negative, the quality are improved dramatically.

Unlike the complicate energy function used to solve TSP (see [1, 13, 16, 23] etc.), the energy function for the Max-Cut problem is simpler, and suitable for neural network. It is possibly another reason that DHNN can produce good solutions for the Max-Cut problem. There are many other combinatorial optimization problems can be formulated to simple integer programming problems. Another further work is to extend the idea in this paper to such combinatorial optimization problems.

However, like traditional Hopfield-type neural network, the neural network proposed in this paper converge to the first stable state they encounter, which will decrease the solution quality. There are many techniques to prevent network to be trapped in stable state corresponding poor quality solution, for example “hill-climbing”. If we combine these techniques with DHNN, we expect that the quality of the solution given by network is better. We shall study this problem further.

References

- [1] S. V. B. Aiyer, M. Niranjan, and F. Fallside. A theoretical investigation into the performance of the Hopfield model. *IEEE Trans. on Neural Networks*, Vol. 1, No. 2, pp. 204-215, 1990.
- [2] S. Benson, Y. Ye, and X. Zhang. Solving large-scale sparse semidefinite programs for combinatorial optimization. *SIAM J. Optim.*, Vol. 10, No. 2, pp. 443-461, 2000.
- [3] A. Bouzerdoum and T. R. Pattison. Neural network for quadratic optimization with bound constraints. *IEEE Trans. on Neural Networks*, Vol. 4, pp. 293-304, 1993.
- [4] S. Burer, R. D. C. Monteiro, and Y. Zhang. Rand-two relaxation heuristics for max-cut and other binary quadratic programs. Technical Report TR00-33, Department of Computational and Applied Mathematics, Rice University, Texas, 2000.

- [5] C. Choi and Y. Ye. Solving sparse semidefinite programs using the dual scaling algorithm with an interactive solver. Working paper, Department of Management Science, University of Iowa, Iowa, 2000.
- [6] A. Cichochi and R. Unbehauen. *Neural networks for optimization and signal processing*, John Wiley & Sons, New York, 1993.
- [7] M. Deza and M. Laurent. *Geometry of cuts and metrics*, Volume 15 of Algorithms and Combinatorics, Springer, 1997.
- [8] N. Funabiki, Y. Takefuji, and K. C. Lee. A neural network model for finding a near-maximum clique. *J. of Parallel and Distributed Computing*, Vol. 14, pp. 340-344, 1992.
- [9] N. Funabiki, S. Nishiawa, and S. Tajima. A binary neural network approach for max cut problems. *ICONIP'96 - HongKong*, pp. 631-635, 1996.
- [10] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of ACM*, Vol. 42, pp. 1115-1145, 1995.
- [11] C. Helmberg and F. Rendl. A spectral bundle method for semidefinite programming. *SIAM J. Optim.*, Vol. 10, pp. 673-696, 2000.
- [12] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, Vol. 79, pp. 2554-2558, 1982.
- [13] J. J. Hopfield and D.W. Tank. Neural computation of decisions in optimization problems. *Biological Cybernetics*, Vol. 52, pp. 141-152, 1985.
- [14] A. Jagota. Approximating maximum clique with a Hopfield network. *IEEE Trans. on Neural Networks*, Vol. 6, pp. 724-735, 1995.
- [15] L. C. Jiao. *System theory of neural networks*, Xian Electronic Scientific University Publishing House, Xian, China, 1990. (in Chinese)
- [16] A. Joppe, H. R. A. Cardon, and J. C. Bioch. A neural network for solving the Traveling Salesman Problem on the basis of city adjacency in the tour. *IJCNN*, Vol. 3, pp. 961-964, 1990.
- [17] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computation*, pp. 85-103. Plenum Press, New York, 1972.
- [18] M. P. Kennedy and L. O. Chua. Neural networks for nonlinear programming. *IEEE Trans. on Circuits and Systems*, Vol. 35, No. 5, pp. 554-562, 1988.
- [19] S. Poljak and Z. Tuza. Maximum cuts and large bipartite subgraphs. *DIMACS series in Discrete Mathematics and Theoretical Computer Science*, Vol. 20, 1995.
- [20] K. A. Simth. Neural networks for combinatorial optimization: a review of more than a decade of research. *INFORMS Journal on Computing*, Vol. 11, No. 1, 1999.
- [21] J. Wang. Analysis and design of a recurrent neural network for linear programming. *IEEE Trans. on Circuits System*, Vol. 40, pp. 613-618, 1993.
- [22] Y. Xia. A new neural network for solving linear programming and its application. *IEEE Trans. on Neural Networks*, Vol. 7, No. 2, pp. 525-529, 1996.
- [23] X. Xu and W. T. Tsai. Effective neural algorithms for Traveling Salesman Problem. *Neural Networks*, Vol. 4, pp. 193-205, 1991.
- [24] X. S. Zhang. *Neural networks in optimization*, Volume 46 of Nonconvex Optimizations and Its Applications, Kluwer, 2000.
- [25] X. S. Zhang and H. C. Zhu. A neural network model for quadratic programming with simple upper and lower bounds and its application to linear programming. *Lecture Notes in Computer Science*, 834, pp. 119-127, Springer-Verlag, 1994.
- [26] X. J. Zhuo and X. S. Zhang. Hopfield-type neural network for solving four-coloring map problems. *OR Transactions*, Vol. 3, No. 3, pp. 35-43, 1999. (in Chinese)