2010 International Symposium on Nonlinear Theory and its Applications
NOLTA2010, Krakow, Poland, September 5-8, 2010

Improvement of Accuracy and Processing Speed of a Maximum Neural Network Algorithm for the Channel Assignment Problem

Kazunori Nemoto† and Junji Kitamichi‡

†School of Computer Science and Engineering, The University of Aizu
‡Graduate School of Computer Science and Engineering, The University of Aizu
Aizuwakamatsu, Fukushima 965-8580, JAPAN
Email: kitamiti@u-aizu.ac.jp

Abstract—In recent years,the popularity of cellular mobile communication systems increases steady. However, the usable frequency spectrum or channels for the Cellular Radio Networks(CRNs) are limited. Thus, algorithms for an efficient utilization of channels: the Channel Assignment Problem(CAP) have become important. In this paper, we propose an improved Neural Network Algorithm (NNA) with parallelism to improve solution accuracy and speed-up processing for the static CAP in CRNs. Our proposed algorithm achieves the improved solution accuracy because of large hill-climbing. In addition, parallel processing using multi-thread can achieve faster processing. We verify performance through simulations using benchmark problems and our proposed algorithm can search better solutions and obtains faster processing speeds than the existing one.

1. Introduction

Recently, the popularity of cellular mobile communication systems increases steady. However, the usable frequency spectrum or channels for the Cellular Radio Networks(CRNs) are limited. Thus, algorithms for an efficient utilization of channels:the Channel Assignment Problem(CAP) have become increasingly important[1]-[4],[6]. CRNs consist of a number of fixed Base Transceiver Stations (BTSs) and a larger number of connection requests from cellular phone users. These users can receive cellular phone service due to the channel assignments allocated by a BTS. The CAP entails allocating channels to cells in the CRN such that the effective assignment of required channel numbers minimizes mutual interference while satisfying electromagnetic compatibility constraints as possible. We consider the following three constraints [6]. (1)For a certain pair of radio cells, the same channel can not be used simultaneously. (2)Any pair of channels assigned to a cell should have a certain distance between them. (3)The adjacent channels in the frequency domain can not be assigned to adjacent cells simultaneously.

The CAP can be reduced from the graph-coloring problem which is NP-complete [7], and its computation time grows exponentially for a large-scale network. Thus, an exact search for the optimal solution is impracticable. For this reason, many researchers have investigated approximate algorithms for the CAP. Algorithms using Neural Network Algorithms(NNAs) have been proposed[1]-[4],[6].

In this paper, an expanded Maximum Neural Network (MNN) algorithm, proposed by Ikenaga and colleagues[2], is verified, and for improvement of the solution accuracy, large hill-climbing is suggested. The addition of large hill-climing causes slower processing speeds than the original one. Thus, speed-up methods using multi-thread are proposed. We verify performance of proposed algorithm through simulations using some benchmarks and our proposed algorithm can search better solutions and obtains faster processing speeds than the existing one.

2. Channel Assignment Problem

The Channel Assignment Problem(CAP) consists of a number of users and Base Transceiver Stations(BTSs). In this paper, a BTS is assumed to have a hexagonal-shaped management domain (cell) as shown in Figure 1. If a user associated with a cell requests a connection, the BTS assigns a channel to accommodate it. Then, the effective channel assignment is required for each cell on the cellular radio network as the CAP.

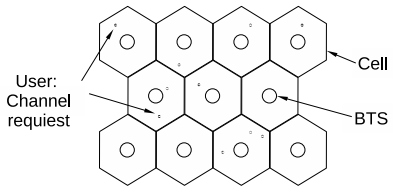


Figure 1: Cellular radio network

We assume N cells and M channels in the CRNs. For a demand vector D (the channel requirements for cells) and a $N \times M$ compatibility matrix C, a channel assignment minimizing total interference is required. The interference between cells is represented by interference matrix E, a three dimensional $N \times N \times M$ matrix. An element e_{ijk} in E indicates interference of any channels as having a distance of k from channel j assigned to cell i. The demand vector D has N elements and an element d_i indicates the number of required channels form cell i.

In this paper, the interference matrix E assumes that the compatibility matrix C gives E as defined by Smith[1].

$$e_{ij0} = c_{ij} \ , \ e_{ijk} = \begin{cases} 0: & (if \ \ c_{ij} \le k) \\ c_{ij} - k: & (if \ \ c_{ij} > k) \end{cases} \qquad (1)$$

where c_{ij} (i=1, $\cdots$, N, j=1,$\cdots$, N) is a element of C and is given by Gamst's compatibility matrix of the CAP.

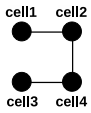
Figure 2 shows an example of the CAP with four cells. Figure 2(a) shows compatibility matrix C and demand vector D. Figure 2(b) illustrates interference matrix E. Figure 2(c) shows the corresponding networks topology of matrix C. The solution for the minimum number of channels needed for no interference assignment in this example is 11 because 11 channels would be needed for cell 4 as can be seen in Figure 2. Figure 2(d) shows an optimal solution in this example, where the total interference is 1.

$$C = \begin{pmatrix} 5 & 4 & 0 & 0 \\ 4 & 0 & 0 & 1 \\ 4 & 5 & 0 & 1 \\ 0 & 1 & 2 & 5 \end{pmatrix} \qquad D = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 3 \end{pmatrix}$$

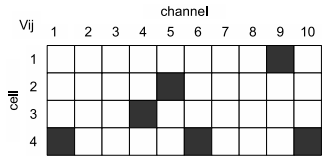
(a)Compatibility matrix and demand vector

$$e_{ij1} = \begin{pmatrix} 4 & 3 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{pmatrix} \qquad e_{ij2} = \begin{pmatrix} 3 & 2 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

(b)Interference matrix



(c) The corresponding network topology



(d) Example of a channel assignment result

Figure 2: Example of CAP

3. Neural network algorithm and Ikenaga's algorithm

A Neural Network Algorithm(NNA) is a heuristic algorithm and a mathematical model based on a neural network. A neural network model is decided by ways of aggregates of neurons. A neuron has multi inputs and one output which collects other neurons outputs as input then calculates own state and decides the output. In this paper, the Hopfield neural network [5] as the neural network and an extended maximum neuron as a neuron model [2] are employed. To solve problems using NNA, problems need to be represented by neurons, an energy function that is a non-negative function and intends states of whole neurons need to be set the minimum if state is the optimal or sub-optimal solution, and the minimum solution is searched by a motion equation using the steepest descent method[5].

We describe Ikenaga's NNA algorithm. A two dimensional $N \times M$ array is used as neuron representation and a neuron ij, an assignment for cell i and channel j, has integral U_{ij} as input and V_{ij} whose value is 0 or 1 as output. "The V_{ij} is 1" means that channel j is assigned to cell i and "the V_{ij} is 0" means that there is no assignment.

M neurons of each cell is sorted in descending order, and the outputs of top to D_i-th neurons are 1 and others are 0.

$$V_{ij} = \begin{cases} 1: & (if\ \ U_{ij} \geq U_{i-th}) \\ 0: & (otherwise) \end{cases} \tag{2}$$

where U_{i-th} is the d_i-th largest value in $U_{i1},\cdots,U_{iM}$, If more than d_i neurons satisfy $U_{ij} \geq U_{i-th}$, the neurons which output 0 previously are selected preferentially because the search space is expanded by the changes in the assignment result and more too, neurons which output 1 are determined randomly from acceptable neurons. By employing the expanded maximum neuron, the energy function has only one term which represents the total interference.

$$E = \frac{A}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{p=1}^{N} \sum_{q=1,(i,j)\neq(p,q)}^{M} e_{ip|j-q|} V_{pq} V_{ij} \tag{3}$$

where (i,j)≠(p,q) represents q≠j when i=p.

The motion equation is defined as following.

$$\begin{aligned} \frac{dU_{ij}}{dt} = -\frac{\partial E}{\partial V_{ij}} = \quad & -A_1 \sum_{p=1}^{N} \sum_{q=1,(i,j)\neq(p,q)}^{M} e_{ip|j-q|} V_{pq} \\ & (if (t\ \ mod\ \ T_\omega) \geq \omega) \\ & -A_2 \sum_{p=1}^{N} \sum_{q=1,(i,j)\neq(p,q)}^{M} e_{ip|j-q|} V_{pq} V_{ij} \\ & (if (t\ \ mod\ \ T_\omega) < \omega) \\ & +B \cdot h(\sum_{p=1}^{N} \sum_{q=1,(i,j)\neq(p,q)}^{M} e_{ip|j-q|} V_{pq} V_{ij}) \\ & +C(1 - V_{ij}) \end{aligned} \tag{4}$$

where the term A_1 is obtained by differentiating partially the energy function with respect to V_{ij}. In the case of $(t\ mod\ T_\omega) \geq \omega)$, the term$A_1$ is used, and in the other case the term A_2 is used. The omega function ejects solution from the local minimum by encouraging competition between neurons. The term B is a hill-climbing term which serves escaping from the local minimum, by assigning channels have no interference from surrounding cells. The function h(x) is 1 if $x < 0$, in other cases h(x)=0. The term C is shaking term. By deterring immobilization of the channel assignment, the shaking term encourage escaping from local minimum.

$$T_n = T_s \times \alpha^n \tag{5}$$

where T_n represents used hours of shaking term, T_s represents used interval and A_1, A_2, B, and C is coefficient.

For advancement of solution accuracy the regular interval assignment for the most congested cell is adapted. The interference is minimized in the cell because cells which have many demands tend to generate interference.

For profiling this algorithm, we implement this MMN in C language. 10 CAP benchmark problems are used in [1] shown in Figure 3. The same parameter values as [2] used in simulations. The compatibility matrix C_5 and the demand vector D_5 which are used in the KUNZ problems are obtained by considering only the first 10 regions in the KUNZ1 problem, 15 regions in the KUNZ2, 20 regions in the KUNZ3 and the entire data set in the KUNZ4.

Problem	N	M	Req	D	C
EX1	4	5	6	D_1	C_1
EX2	5	17	13	D_2	C_2
HEX1	21	37	120	D_3	C_3
HEX2	21	91	120	D_3	C_4
HEX3	21	21	112	D_4	C_3
HEX4	21	56	112	D_4	C_4
KUNZ1	10	30	72	$[D_5]_{10}$	$[C_5]_{10}$
KUNZ2	15	44	113	$[D_5]_{15}$	$[C_5]_{15}$
KUNZ3	20	60	140	$[D_5]_{20}$	$[C_5]_{20}$
KUNZ4	25	73	167	D_5	D_5

(a) Input sizes of simulated benchmarks

$$C_1 = \begin{pmatrix} 5 & 4 & 0 & 0 \\ 4 & 5 & 0 & 0 \\ 0 & 0 & 5 & 2 \\ 0 & 0 & 2 & 5 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 5 & 4 & 0 & 0 & 1 \\ 4 & 5 & 0 & 1 & 0 \\ 0 & 0 & 5 & 2 & 1 \\ 0 & 1 & 2 & 5 & 2 \\ 1 & 0 & 1 & 2 & 5 \end{pmatrix}$$

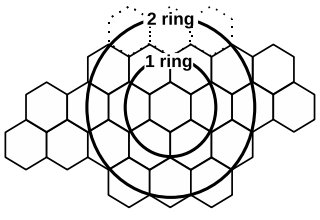
$$C_3 : C_{ij} = \begin{cases} 2: & (if\ in\ the\ same\ cell) \\ 1: & (if\ the\ 2\ ring\ cell) \\ 0: & (else) \end{cases}$$

$$C_4 : C_{ij} = \begin{cases} 3: & (if\ in\ the\ same\ cell) \\ 2: & (if\ the\ 1\ ring\ cell) \\ 1: & (if\ the\ 2\ ring\ cell) \\ 0: & (else) \end{cases}$$

$C_5 =$

```
21101 01111 01111 00000 00000
12101 01101 01111 00000 00000
11211 11111 11110 00000 00000
00120 01111 11100 00000 00111
11102 00001 11111 10000 00000

00100 21111 00000 00000 00000
11110 12111 11100 00000 00000
11110 11211 11100 00000 00010
10110 11121 11000 00000 00011
11111 11112 11111 10000 01010

00111 01111 20111 10111 11111
11111 01111 02110 00000 00000
11111 01101 11211 11111 00000
11101 00001 11121 11111 00000
11001 00001 10112 11111 11000

00001 00001 10111 21111 00000
00000 00000 00111 12110 00000
00000 00000 10111 11211 00000
00000 00000 10111 11121 11100
00000 00000 10111 10112 11100

00000 00000 10001 00011 21100
00000 00001 10001 00011 12111
00010 00000 10000 00011 11211
00010 00111 10000 00000 01121
00010 00010 10000 00000 01112
```

(b) Example of compatibility matrices.



(c)Hexagonal network for HEX benchmark

D_1^T = (1, 1, 1, 3)
D_2^T = (2,2,2,4,3)
D_3^T = (2,6,2,2,2,4,4,13,19,7,4,4,7,7,9,14,7,2,2,4,2)
D_4^T = (1,1,1,2,3,6,7,6,10,10,11,5,7,6,4,4,7,5,5,5,6)
D_5^T = (10,11,9,5,9,4,5,7,4,8,8,9,10,7,7,6,4,5,5,7,6,4,5,7,5)

(d) Examples of demand vectors.

Figure 3: The bench mark problems

For each benchmark problem, using different initialized values for each neuron, the CAP is simulated 10 times by the implemented algorithm and average maximum renewal times are obtained by results. Where, the maximum renewal time means a maximum taken time for updating minima of total interference. Also, proportion of demands to elements of $N \times M$ matrix is added for data. The simulation results are shown in Table 1. *Req* represents the total number of requirements, *Per* does proportion of demands, *Sol* does the average solving times and *Ren* does the average maximum renewal times. The *Sol* and *Ren* increases in proportion to the *Per* is seen. For each trial, over 300 updating, escaping from the local minimum tends to difficult because overlarge value of U_{ij} interrupts.

Problem	N	M	Req	Per	Sol	Ren
EX1	4	5	6	0.13	8.01	3.37
EX2	5	17	13	0.15	41.1	13.7
HEX1	21	37	120	0.15	276	53.4
HEX2	21	91	120	0.06	195	40.4
HEX3	21	21	112	0.25	291	90.3
HEX4	21	56	112	0.10	188	46.7
KUNZ1	10	30	72	0.24	249	82.3
KUNZ2	15	44	113	0.17	238	47.1
KUNZ3	20	60	140	0.11	198	42.5
KUNZ4	25	73	167	0.09	177	37.8

Table 1: Data for profiling and simulation results.

4. Proposed improvement methods for MNN

To improve the solution accuracy, a large hill-climbing is executed for neurons after a certain period of renewal times. Using the values of U the large hill-climbing makes neurons a partway convergent state and it leads to better convergence for the solution.

$$R = D_i * 0.6 \tag{6}$$

$$U_{ij} = \begin{cases} 2: & (if\ U_{ij} \geq U_R) \\ 0: & (else) \end{cases} \tag{7}$$

where R is an integer used for thresholds and the eq.(7) shows six out of ten D_i are assigned to R. The inputs U_{ij} of neurons in each cell are sorted in the descending order and the beginning to the R-th neurons are selected and there initialized input data is 2 and other neurons input data is 0.

$$S = 280 * P + 50 \tag{8}$$

Eq.(9) is a linear equation relating with the proportion of demands P and the large hill-climbing time S obtained from analyzed result of benchmarks. If the minimum of total interference is not updated for S times, the energy function is assumed to lapse into the local minimum and large hill-climbing is executed to escape.

Multi-thread processing is a measure to perform plural threads in parallel in a process. There is no need to switch memory spaces at threads switching and through the memory threads can share data mutually thus processing is executed with few resources and overheads.

The procedures of the NNAs are following.

1. Assigning initial values for neurons input U.
2. Deriving neurons output V from U**(U)** .

3. Updating U with the motion equation**(V)**.
4. Calculating total interference and keeping the minimum value**(W)**.
5. Goto to 2 until termination conditions are satisfied.

Multi-threaded processing of procedure **2** ,**3** and **4** for neurons groups is employed. Global variables are used for shared data. Neuron outputs V is shared data and each thread have neurons inputs U, copied neuron output V and total interference as local data. While to prevent competing data synchronization is needed, each thread has thread-ID and cells are processed based on thread-ID thus competing not happened at updating V to global from local memory. Hence, synchronization is needed only when shared data reading. Additionally copying local data from global data every renewal to minimize dealing with shared data decreases the synchronization. For additionally faster processing the multi-threaded processing for procedure **4**, which is a bottleneck, is proposed. **W** holds neuron outputs V and total interference as shared data and copied V as local data. In **U**, because of multi-threaded processing, total interference translate into shared data and synchronization are needed at data sharing. Additionally speculative execution of **U** and **V** during processing **W** is used. There is dependency between **U** and **W**. However, synchronizing start of data read from shared data resolves it. The numbers of threads used for executing **U,V** and **W** are $K/2$ and$K/2$ (K is even) or $(K-1)/2+1$ and $(K-1)/2$ (K is odd) respectively, where K is the total number of threads in parallel. Figure 4 shows an example if the number of threads is 4.

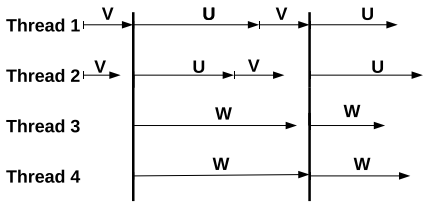


Figure 4: Multi-threads processing with speculative exec.

5. Simulation Results

The proposed methods are implemented in C language on dual Xeon X5450, 3.0GHz, 4 core, 16G Mem, Linux 2.6.18, GCC 4.1.2 and compiler option -O2, and we simulate 10 times with different initial random values for each benchmark problem. We show the results in Table 2, which are the solution accuracy of the proposed method and Ikenaga's one. "AV" represents the average of the objective values and "Min" represents the minimum solutions. As for the average values, the proposed algorithm is the same as or better than Ikenaga over all problems and as for the minimum value better values are seen in HEX2 and HEX3.

Table 3 shows the computation times of proposed method and existing algorithm(Ike.), where the number of threads is 2 to 8. Minimum time is seen from 7 threads processing. 7 threads processing leads to better outcomes than above for the all problems and achieve up to twice faster processing and about up to 4 times faster processing than the existing algorithm.

Problem	Ikenaga		Proposed algorithm	
	AV.	Min	AV.	Min
EX1	0.0	0	0.0	0
EX2	0.1	0	0.0	0
HEX1	46.2	46	**46.0**	46
HEX2	16.7	16	**16.5**	**15**
HEX3	78.6	76	**77.2**	**74**
HEX4	17.0	15	**16.5**	15
KUNZ1	20.8	20	**20.6**	20
KUNZ2	31.2	30	**30.9**	30
KUNZ3	13.0	13	13.0	13
KUNZ4	0.6	0	0.0	0

Table 2: Simulation results for solution accuracy

Problem	2(1:1)	3(2:1)	4(2:2)	5(3:2)	6(3:3)	**7(4 : 3)**	8(4:4)	Ike.
EX1	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00
EX2	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00
HEX1	2.16	1.55	1.14	0.96	0.89	**0.79**	0.97	3.10
HEX2	13.65	9.67	6.95	5.42	4.91	**4.12**	4.81	19.48
HEX3	0.73	0.56	0.44	0.42	0.38	**0.41**	0.43	1.01
HEX4	5.01	3.52	2.57	2.05	2.06	**1.67**	1.98	7.14
KUNZ1	0.39	0.38	0.28	0.28	0.24	**0.27**	0.29	0.53
KUNZ2	1.67	1.30	0.98	0.84	0.80	**0.68**	0.81	2.36
KUNZ3	5.26	3.75	2.81	2.21	2.01	**1.78**	2.14	7.47
KUNZ4	12.03	8.26	6.59	4.97	4.59	**3.85**	4.45	13.43

Table 3:Execution times of proposed multi-threads processing (sec).

6. Conclusions

In this paper, we present a faster and more accurate processing methods for the existing neural network algorithm for the channel assignment problem in cellular radio networks and evaluate it's performance. A large hill-climbing method is used to improve solution accuracy and multi-thread processing is employed to speed-up processing up to twice faster and about up to 4 times faster than the existing algorithm.

As future works, we will address the dynamic channel assignment problem and hardware implementation with the proposed algorithm.

References

[1] K. Smith, and M. Palaniswami,"Static and dynamic channel assignment using neuralnetworks,"IEEE Jornal of Commun., vol. 15, no. 2, pp. 238-249, Feb. 1997.

[2] K. Ikengaga, et al.,"An expanded maximum neural network algorithm for a channel assignment problem in cellular radio networks," IEICE Trans. of Fundamentals, vol, E82-A, no. 5, pp. 683-690, May. 1999.

[3] K. Okutani, et al."A study on a two-phase neural network algorithm for channel assignment in cellular radio networks," IEICE, Tech. Rep. ss96-52, Jan. 1997.

[4] K. Ikenaga. et al."A maximum neural network algorithm for a channel assignment problem in cellular radio networks," IEICE, Tech. Rep. NLP97-172, Mar. 1998.

[5] Y. Takefuji and T. Saito,"Models of neuralnetworks," Handbook of applied cases: Neural computing, Corona, 2001, pp.17-56.

[6] N. Funabiki and Y. Takefuji,"A nerural network parallel algorithm for channel assignment problems in cellular radio networks," IEEE Trans. of Vehiclular Technology, vol. 41, no. 4, pp. 430-437, Nov. 1992.

[7] W.K.Hale,"Frequency Assignment: Theory and Applications," Proc.IEEE, vol.68, no.12, pp. 1497-1514, Dec. 1980.