Available online at www.elixirpublishers.com (Elixir International Journal)

Network Engineering

Elixir Network Engg. 61 (2013) 16675-16680

ISSN: 2229-712X

A three-stage heuristic combined Hopfield neural network for channel assignment problem in cellular mobile system

Omid Moradi

ACECR(jahad-e-daneshgahi), Isfahan University of Technology, Isfahan, Iran.

ARTICLE INFO

Article history:

Received: 26 May 2013;

Received in revised form:

24 July 2013;

Accepted: 31 July 2013;

Keywords

Neural network,

Channel assignment,

Cellular system,

Three stage.

ABSTRACT

A three-stage algorithm of combining sequential heuristic methods into a Hopfield neural network is presented for the channel assignment problem in cellular mobile communication systems in this paper. The goal of this problem is to find a channel assignment to requested calls with the minimum number of channels subject to interference constraints between channels. The three-stage algorithm consists of: 1) the regular interval assignment stage; 2) the greedy assignment stage; and 3) the neural-network assignment stage. In the first stage, the calls in a cell determining the lower bound on the total number of channels are assigned channels at regular intervals. In the second stage, the calls in a cell with the largest degree and its adjacent cells are assigned channels by a greedy heuristic method. In the third stage, the calls in the remaining cells are assigned channels by a Hopfield neural network. The performance is verified through solving well-known benchmark problems. Especially for Sivarajan's benchmark problems, my three-stage algorithm first achieves the lower bound solutions in all of the 12 instances.

Introduction

There is a continuously growing demand for mobile telecommunication. However, the number of usable frequencies which are necessary for the communication between mobile users and the base stations of cellular radio networks is very limited. So this restricted number of frequencies constitutes an important bottleneck for the capacity of mobile cellular systems. Consequently, it becomes more and more important to use the frequencies as efficiently as possible. This means that, assigning the frequencies to the different base stations, it is desirable to reuse the same frequency as much as possible. Thereby, it is important to avoid possible interferences between different mobile users. At the same time, the number of frequencies assigned to each base station must be chosen large enough to satisfy the given demand in the corresponding cell. The channel assignment problem has been extensively studied to solve this important task in cellular mobile communication systems [1]-[8].

The channel assignment problem (CAP) in this paper is based on a common model. The service area of the system is divided into a large number of hexagonal cells. A cell composes a unit area to provide communication services, where every user is located in one cell. When a user requests a call for this system, a channel or frequency spectrum is assigned there to provide the communication service. This channel assignment must satisfy the constraints to avoid the radio interference between channels. Three types of constraints have usually been considered in CAP.

1) *The Cochannel Constraint (CCC):* The same channel cannot be reused in the cells within a certain distance from each other. A set of channel-reuse forbidden cells is called a cluster, where a different channel must be assigned to every call.

2) *The Adjacent Channel Constraint (ACC):* Adjacent channels cannot be assigned to adjacent cells simultaneously. In other words, any pair of channels in adjacent cells must have a specified distance. Note that the distance indicates the difference in the channel domain.

3) *The Cosite Constraint (CSC):* Any pair of channels in the same cell must have a specified distance. This distance for CSC is usually larger than that for ACC.

The goal of CAP is to find a channel assignment to every requested call with the minimum number of channels subject to the above three constraints[2].

Many researchers have investigated the CAP in telephone networks[2]-[8]. Funabiki and Takefuji [2] proposed a neural-network parallel algorithm for channel-assignment problem. All input values are sequentially updated, while all output values are fixed. Then, all output values are sequentially updated, while all input values are fixed. Their neural-network model is composed of the hysteresis McCulloch–Pitts neurons. In the Funabiki and Takefuji model, four heuristics were used to improve the convergence rate of channel assignment. Kunz [3] used the continuous Hopfield network, where the output of each neuron V_i was a fixed function f of the internal state u_i , i.e., $V_i=f(u_i)$, where $f(x)=1/2(1+\tanh(\lambda x))$. The Kunz neural-network model required a large number of iterations in order to reach the final solution, and there were also difficulties in finding the proper values for λ and the parameters in the interconnection weights and energy function. Kunz considered cochannel and cosite interference in his neural-network model [3]. Chan *et al.* [5] used a feedforward neural network, which had a learning process prior to actual channel assignment. For the learning process, they used training data that was dependently obtained by other assignment methods. The performance of their algorithm is totally dependent on the used training data. Also, in [5], only the cochannel constraint (CCC) was considered. Vidyarthi *et al.*[6] proposed a hybrid channel assignment approach using an efficient evolutionary strategy. They developed an evolutionary strategy(ES) which optimized the

Tele:

E-mail addresses: moradi.acecr@gmail.com

channel assignment. Moradi in [7] proposed fixed channel assignment and neural network algorithm for CAP. He also investigated a Hopfield neural network in [8]. The channel assignment problem is formulated as an energy-minimization problem such that the energy is at its minimum when all the constraints are satisfied and the number of assigned frequencies are the same as required channel number in each cell.

In this paper, I propose a three-stage algorithm for CAP by combining sequential heuristic algorithms into a Hopfield neural network algorithm. The three-stage algorithm consists of the regular interval assignment stage, the greedy assignment stage, and the neural-network assignment stage. The performance is verified through solving the benchmark problems by Sivarajan, Kunz, and Kim. For Sivarajan's and Kunz's problems, my algorithm first achieves the lower bound solutions in all the instance and with the comparable average iteration number and the convergence rate, whereas my algorithm provides the better solution quality than Moradi's algorithm in[8].

II. Problem Formulation of CAP

The three constraints for the channel interference in the *N*-cell system are described by an $N \times N$ symmetric compatibility matrix *C*. The nondiagonal element c_{ij} $(i \neq j)$ of *C* represents the minimum distance between a channel assigned to cell $\neq i$ and a channel to cell $\neq j$. The diagonal element C_{ii} of *C* represents the distance between a pair of channels assigned to cell $\neq i$. Thus, *CCC* is described by C_{ij} =1, *ACC* is by $C_{ij} \geq 2$ and *CSC* is by $C_{ii} \geq 1$ in *C*, respectively. A set of requested calls in the *N*-cell system is given by an *N*-element demand vector *D*. The *i*th element d_i of *D* represents the number of channels for the requested calls in cell $\neq i$. Let f_{ik} be the *k*th channel assigned to cell $\neq i$ for $i=1,\ldots,N$ and $k=1,\ldots,d_i$. Then, the total number of required channels *M* can be represented by:

$$M = \max imum \ \ of \ \ \{f_{ik}\} \ \ for \ i = 1,\ldots, N \ \ and \ \ k = 1,\ldots, d_i . \tag{1}$$

Given a pair of a compatibility matrix *C* and a demand vector *D*, the goal of channel assignment problem is to find a channel assignment $\{f_{ik}\}$ with the minimum value of *M* subject to the interference constraints:

$$\left| f_{ik} - f_{jl} \right| \geq c_{ij} \ \ for \ i = 1,\ldots, N, \ \ j = 1,\ldots, N, \ \ k = 1,\ldots, d_i \ \ and \ \ l = 1,\ldots, d_j \ \ except \ \ for \ i = j \ and \ k = l. \tag{2}$$

III. Three-Stage Algorithm for CAP

A. Overall Structure of Three-Stage Algorithm

The three-stage algorithm for CAP consists of: 1) the regular interval assignment stage; 2) the greedy assignment stage; and 3) the neural-network assignment stage. In the first stage, the calls in a cell, which determines the lower bound on the total number of channels, are assigned channels at regular intervals. In the second stage, the calls in the greedy region are assigned channels by a greedy heuristic method sequentially. Initially, the greedy region is composed of a cell with the largest degree and its adjacent cells because the channel assignment to cells with large degrees is usually more difficult than the assignment to other cells. Then, every time the whole assignment is failed, the greedy region is expanded by additionally including the cells adjacent to the original region. In the third stage, the calls in the remaining cells are assigned channels by a binary neural network in parallel.

The overall procedure of the three-stage algorithm is described as follows.

1) Input a compatibility matrix *C* and a demand vector *D*.
2) Initialize the total number of channels *M* by the lower bound or its approximation.
3) Compute the degree of cell # *i* , $\deg_i$, for $i=1,\ldots,N$

$$\deg_i = \left(\sum_{j=1}^{N} d_j c_{ij} \right) - c_{ij} \tag{3}$$

4) Apply *the regular interval assignment stage* to the cell which determines the lower bound of *M* , if it exists.
5) Initialize the greedy region by a cell with the maximum degree and its adjacent cells.
6) Apply *the greedy assignment stage* to the greedy region while the assignment by 4) is fixed. If this assignment is failed, then increment *M* by one ($M=M+1$) and go to 4).
7) Apply *the neural-network assignment stage* to the remaining cells while the assignment by 4) and 6) is fixed. If this assignment is failed, then expand the greedy region by additionally including its adjacent cells and go to 6). If no expandable cell exists, then increment *M* by one and go to 4).

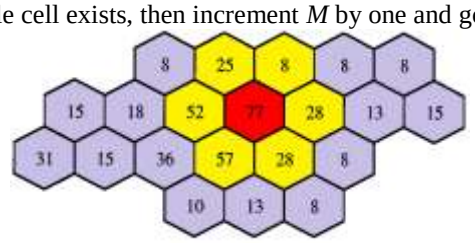

Figure 1. The 21-cell system in Sivarajan's benchmark problems

Fig.1 shows a cellular system of 21 cells in Sivarajan's benchmark problems [9]. Each number inside a cell represents the number of requested calls. In our three-stage algorithm, the regular interval assignment stage is applied to the cell with 77 calls. The greedy assignment stage is initially applied to its adjacent six cells because the 77-call cell has the maximum degree. Then, the neural-network assignment stage is applied to the other cells.

B. Regular Interval Assignment Stage

The regular interval assignment stage assigns channels to the calls in the cell which determines the lower bound on *M*, at a regular interval of satisfying *CSC* in order to minimize the total number of channels *M*. Let us consider the channel assignment to cell #*i*, where C_{ii} and d_i represent *CSC* and the number of requested channels for cell #*i*, respectively. Each of d_i calls is assigned a channel at an interval of either C_{ii} or (C_{ii}+1), after the first and last channels are assigned to cell #*i*. Actually, the first *x* calls are assigned at the interval of C_{ii} , whereas the rest (d_i-*x*) calls are assigned at the interval of (C_{ii}+1).

$$f_{ik} = 1 + c_{ii}(k-1) \qquad for \ k = 1,\ldots, x$$
$$f_{ik} = 1 + c_{ii}(x-1) + (c_{ii}+1)(k-x) \qquad for \ k = x+1,\ldots, d_i \tag{4}$$

The value of *x* is given by solving the linear equation

$$1 + c_{ii}(x-1) + (c_{ii}+1)(d_i - x) = M \tag{5}$$

C. Greedy Assignment Stage

The greedy assignment stage assigns channels sequentially by using the assignment list, which is made of calls in the greedy

region after sorted in the descending order of the assignment difficulty in[10]. Each call in this list is assigned a channel by the requirement exhaustive strategy in [11]. The procedure of this stage is as follows.

1)Make a list of the calls in the greedy region in the ascending order of the cell index number.

2)Initialize the assignment difficulty of every call in the list by zero and the iteration number *t* by one.

3)Initialize the channel number *j* by one.

4)Check whether or not channel *#j* can be assigned to each unassigned call in the list from the top to the bottom sequentially. If it can be assigned there without interference, then assign channel *#j* to that call.

5)If every call in the list is assigned a channel, then terminate the procedure as success, else if $j<M$ then increment by one and go to 4).

6)If the iteration number *t* reaches its upper limit *G* max, then terminate the procedure as failure, else increment *t* by one.

7)Add a randomized real number between zero and one to the assignment difficulty for each unassigned cell in this iteration, clear the assignment result by this stage, remake the list by sorting the calls in the descending order of the assignment difficulty, and go to 3).

II. Hopfield Neural Network Assignment Stage

The Hopfield neural-network stage is base on the algorithm in [8]. Each processing element (neuron) is fully interconnected in the Hopfield network. The *i*th neuron is described by its state, which is denoted by V_i . Each neuron has two possible states. The value of each state is determined by the total input from other neurons followed by a thresholding operation. The input of the *i* th neuron is derived from two sources: the outputs of other neurons scaled by the connection weights and an appropriate external input. The total input to neuron #*i* is denoted by U_i

$$U_i = \sum_j W_{ij} V_j + I_i \tag{6}$$

where W_{ij} is the connection weight from neuron #*j* to neuron #*i* and I_i is the external input. Each neuron updates its own state according to a thresholding rule with threshold *THD* as shown by

$$V_i = \begin{cases} 1 & \text{if } U_i > THD \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The thresholding rule can be applied asynchronously (in series) or synchronously (in parallel). In the asynchronous mode, this rule is applied sequentially to each neuron, and the state of each neuron is updated individually. In the synchronous mode, this thresholding operation is simultaneously applied to every neuron, and the states of all neurons are updated at the same time. The updating operation is terminated when the states are unchanged or the energy has reached a minimum value. The energy function *E* is defined as

$$E = -\frac{1}{2}\sum_i \sum_j W_{ij} V_i V_j - \sum_i V_i I_i \tag{8}$$

This energy function is minimized by the Hopfield neural network updating procedure. The successive application of the updating procedure will force the network to converge such that the energy of the network becomes smaller during the updating procedure. When the network reaches a stable state, it has fallen into minimum energy state, where this could be a local or global minimum. W_{ij} and I_i should be set appropriately for the applications so that *E* represents the function, which is minimized to solve the combinatorial optimization problems. The energy function should represent all the constraints of the problem.

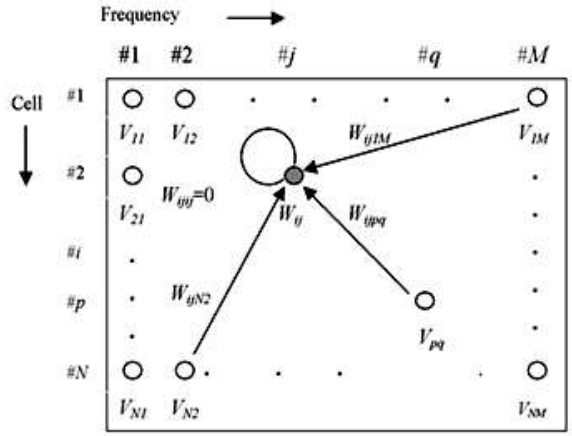

Figure 2. A 2-D Hopfield network for the channel assignment problem.

In the simulation, I considered a mobile radio network that has *N* cells (or base stations) and *M* available frequency channels. The value of *M* is set to be the lower bound of required number of channels *LB* for a given channel assignment problem. Fig.2 shows a two-dimensional (2-D) Hopfield network for the channel assignment problem. Each *i*th cell can carry any of the d_i frequencies among the *M* frequencies if the carriage of frequencies does not violate the imposed constraints of the channel-assignment problem. The value of the processing unit V_{ij} , $1 \le i \le N$ and $1 \le j \le M$ indicates if frequency #*j* is assigned to cell #*i*: $V_{ij} = 1$ meaning that the frequency *j* is assigned to cell #*i* and $V_{ij} = 0$, which means that the frequency #*j* cannot be assigned to cell #*i*. In Fig.2, *i* and *p* represent the cell number; *j* and *q* indicate the frequency number, respectively. The state of the current processing neuron to be updated is represented by V_{ij} . An energy function is derived to represent the three constraints of the channel-assignment problem. In *CSC*, a frequency f_{iq} cannot be assigned to cell #*i* if the distance of f_{iq} and any assigned frequency f_{ij} , $1 \le j \le M$ is less than c_{ii} (minimum frequency distance for *CSC*). The energy function for cell #*i* (for *CSC*) can be defined as

$$E^i_{CSC} = \sum_{j=1}^{M}\sum_{q=1}^{M} V_{ij} V_{iq} X_{ijq} \tag{9}$$

Where

$$X_{ijq} = \begin{cases} 1, & \text{if } q \ne j \quad \text{and} \\ & j-(c_{ii}-1) \le q \le j+(c_{ii}-1) \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

For both *ACC* and *CCC*, the frequency f_{ij} cannot be assigned to cell #*i* if the distance of f_{ij} and any other assigned frequency f_{pq} is less than c_{ip} (minimum frequency separation between the frequencies in cell #*i* and cell #*p*). The energy function of *i*th cell for *ACC* and *CCC* can be defined as

$$E^i_{(ACC,CCC)} = \sum_{j=1}^{M}\sum_{p=1}^{N}\sum_{q=1}^{M} V_{ij} V_{pq} Y_{ijpq} \tag{11}$$

Where

$$Y_{ijpq} = \begin{cases} 1, & \text{if } p \ne i \quad \text{and} \\ & c_{ip} > 0 \text{ and } j-(c_{ip}-1) \le q \le j+(c_{ip}-1) \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

In addition to the three constraint conditions, the total assigned channel numbers (ACN's) in the *i*th cell must be the same as the required channel numbers (RCN's) for cell #*i*. The energy function for the ACN can be defined as

$$E_{ACN}^{i} = \left(d_i - \sum_{j=1}^{M} V_{ij} \right)^2 \tag{13}$$

From (9), (11), and (13), the energy function for *i*th cell can be defined as

$$E^{i} = \left(d_i - \sum_{j=1}^{M} V_{ij} \right)^2 + \sum_{j=1}^{M} \sum_{q=1}^{M} V_{ij} V_{iq} X_{ijq} + \sum_{j=1}^{M} \sum_{p=1}^{N} \sum_{q=1}^{M} V_{ij} V_{pq} Y_{ijpq}. \tag{14}$$

The total energy function for the channel-assignment problem is as follows

$$E = \sum_{i=1}^{N} \left(\left(d_i - \sum_{j=1}^{M} V_{ij} \right)^2 + \sum_{j=1}^{M} \sum_{q=1}^{M} V_{ij} V_{iq} X_{ijq} + \sum_{j=1}^{M} \sum_{p=1}^{N} \sum_{q=1}^{M} V_{ij} V_{pq} Y_{ijpq} \right) \tag{15}$$

This energy function can be minimized by an equivalent Hopfield neural network with the appropriate interconnection weights and the external inputs. The interconnection weights must represent the constraints of the optimization problem such as the RCN's for each cell and the three constraint of the channel-assignment problem. Each of the constraints are invoked by inhibitory and excitatory support. If neuron V_{ij} takes a value of unity, which means frequency f_{ij} can be used at cell #*i*, then the neuron V_{iq} within the interference must be inhibited by the *CSC* condition. The constraint can be specified in the form

$$W_{CSC} = -\delta_{ip} \alpha_{jq}(c_{ii}) \tag{16}$$

where δ is the Kronecker delta function and defined with α_{ij} as follows:

$$\delta_{ij} = \begin{cases} 1. & if\ i = j \\ 0 & otherwise \end{cases} \tag{17}$$

$$\alpha_{ij}(x) = \begin{cases} 1, & if\ |i - j| < x \\ 0, & otherwise. \end{cases} \tag{18}$$

In both *ACC* and *CCC*, if neuron V_{ij} takes a value of unity, then a neuron V_{iq} within the interference must be inhibited. This constraint can be specified in the form

$$W_{(ACC,CCC)} = -|(1 - \delta_{ip})| \alpha_{jq}(c_{ij}) \tag{19}$$

To make ACN's the same as the RCN's, I have to feed inhibitory support to the neuron by an amount proportional to the number of assigned frequencies. If RCN is less than or equal to the ACN's, the additional frequencies cannot be assigned to a cell. This constraint can be expressed as

$$W_{ACN} = -\delta_{ip} |(1 - \delta_{jq})| \tag{20}$$

Equation (20) shows that the self-inhibition is not allowed. The purpose of W_{ACN} is to give the inhibitory support to neurons in the same cell in order to avoid the case of ACN>RCN.

From (16), (19), and (20), the total interconnection weight is

$$W_{ijpq} = -\delta_{ip} \alpha_{jq}(c_{ii}) - |(1 - \delta_{ip})| \alpha_{jq}(c_{ij}) - \delta_{ip} |(1 - \delta_{jq})| \tag{21}$$

The interconnection weight W_{ijpq} between V_{ij} and V_{pq} is symmetrical, i.e., $W_{jpq} = W_{pqij}$ for $1 \le i$, $p \le N$ and $1 \le j$, $q \le M$. Self-feedback is not allowed, i.e., $W_{ijij} = 0$.

According to new strategy In [8], to increase the convergence rate, a nonlinear function is applied on the weights as follows

$$W_{ijpq(new)} = A \times sign(W_{ijpq}) \times \exp(|W_{ijpq}|) \tag{22}$$

The external input I_i is defined as

$$I_i = (d_i - 1) \tag{23}$$

The external input I_i is used to give excitatory support to neurons in the same cell to make them to be satisfied by the traffic demand constraint. If a frequency assignment satisfies all the three constraints (*CSC*, *CCC*, and *ACC*) for a cell and the ACN's are less than the RCN's, then that cell must receive excitatory support as it relates to reinforcing that assignment. The summation of inputs from all neurons for the current updating neuron is $-(d_i - 1)$ which should be given an excitatory bias. The input to each neuron (*i*,*j*) in the original Hopfield neural net is defined as

$$U_{ij} = \sum_{p=1}^{N} \sum_{q=1}^{M} W_{ijpq} V_{pq} + I_i \tag{24}$$

When a neuron receives an input, only frequencies which are satisfied by the three channel-assignment constraints are selected as usable frequencies for the cell. A local minimum can be achieved for the channel assignment given in (24) by using the Hopfield updating procedure. If all the assigned frequencies are satisfied by the three channel-assignment constraints, but one or more of the cells have fewer channels than the RCN's, the frequency assignment will never change and the energy value cannot reach the global minima even though more iterations are performed. To prevent this from occurring, I incorporate a forced assignment method, which allows for a frequency to be assigned to a cell by another excitatory input, even though the channel-assignment constraints are violated and the energy is increased. The forced assigned frequency can change the frequency assignment for other cells too. The algorithm will then search for another solution space in order to attempt to reach the global minima when the current assigned channels does not satisfied the traffic demand constraint. The algorithm checks for the ACN with the value of RCN . The difference between the RCN and the ACN constraints is used as an additional excitatory input, which is given by

$$I_{E_{ij}} = \left(d_i - \sum_{q=1}^{M} V_{iq} \right) \tag{25}$$

The values of W_{ACN} and I_i are constant for the neurons in the same cell. However, the value of I_{Eij} is variable based on the current neuron states. In I_{Eij} , the difference of (RCN-CAN) is fed into the neurons in cell #*i*. To count ACN, V_{ij} is included in ACN of (25).

It is possible that the forced reassignment fails, i.e., ACN<RCN although the additional term I_{Eij} is introduced. If ACN<RCN even after the forced reassignment is applied, it means that the assignment for some calls are failed, which will cause the call drops. The number of callers who could not have the channels and their calls are dropped is (RCN-ACN). This case is considered as a *nonconvergence* case. On the other hand,

when all calls have the assigned frequencies without any call dropping, it is considered as *convergence* case. In the simulation results, the convergence rate is the number of cases that all calls have the assigned frequencies without any call dropping before the maximum number of iterations is reached when 100 simulation runs were performed. Input to each neuron of the modified network is defined as

Table I. Specifications for Sivarajan's and Kunz's Benchmark Problems

Instance	Compatibility matrix				Demand Vector	
	N_c	c_{ij}	*acc*	c_{ii}	Call distrib.	Total calls
#1	12	1	2	5	case1	481
#2	7	1	2	5	case1	481
#3	12	1	2	7	case1	481
#4	7	1	2	7	case1	481
#5	12	1	1	5	case1	481
#6	7	1	1	5	case1	481
#7	12	1	1	7	case1	481
#8	7	1	1	7	case1	481
#9	12	1	2	5	case2	470
#10	7	1	2	5	case2	470
#11	12	1	2	7	case2	470
#12	7	1	2	7	case2	470

$$U_{ij} = \sum_{p=1}^{N}\sum_{q=1}^{M} W_{ijpq}V_{pq} + I_i + I_{Eij} \quad (26)$$

The output function consists of a threshold operation. The final output state of the neuron is

$$V_{ij} = f_{out}(U_{ij}) \quad (27)$$

Where

$$f_{out} = \begin{cases} 1, & if\ U_{ij} \geq THD \\ 0, & otherwise \end{cases} \quad (28)$$

and *THD* is the threshold value and *THD*=0 in this paper.

A. Applying Modified Hopfield Network

In this section, the implementation of the algorithm is discussed. The overall algorithm is summarized in the following steps.

1) The initial state of neurons is set to one or zero according to the initialization method.

2) Repeat the following steps until all neurons are picked.

a) Pick neuron V_{ij} according to the updating method.

b) Calculate the input to this neuron by (26).

c) Decide the new state of this neuron by (27) and (28).

3) Compute the energy *E* of the current assignment. If *E=0*, stop and go to Step 4), otherwise, repeat the process from Step 2).

4) The output state of the neurons V_{ij} will be the final assignment based on the compatibility matrix *C* and demand vector *D*.

Initialization method: In general, a random initialization method is used as the initial states for the neurons in the Hopfield network. In my algorithm, initialization technique with the frequency assignment constraints are investigated in order to increase the convergence rate and to decrease the iteration number. The total frequency spectrum is composed of a certain number of blocks for the initialization with consideration of the constraints.

Updating Method: In an asynchronous Hopfield network, the neurons are selected randomly or sequentially by a certain order for the updating. In my algorithm, both random and sequential selection techniques are used along with updating method. The procedural step for updating method is as follows.

Table II. Specification of Simulation Problems Used by Moradi.

	Number of Radio Cells *N*	Lower bound *LB*	Compatibility Matrix *C*	Demand Vector *D*
1	25	73	C_2	D_2
2	21	381	C_3	D_3
3	21	533	C_4	D_3
4	21	533	C_5	D_3
5	21	221	C_3	D_4
6	21	309	C_4	D_4
7	21	309	C_5	D_4

1) Make a list of cells according to the descending order of RCN for each cell.

2) Execute the iteration subroutine as follows until the counter reached to a prespecified maximum iteration number 500 or *E*=0.

a) Choose the cell #*i* according to the order of the cell list.

b) Randomly choose one neuron #*j* in cell #*i* [neuron(*i*,*j*)] and update that neuron.

c) For the next updating neuron, the direction is randomly chosen whether in favor of the left- [neuron(*i* , *j*-1)] or the right-side neuron [neuron(*i* , *j*+1)].

d) After the initial direction is decided by Step *c*), the next neurons are updated sequentially.

e) Repeat Steps a) – *d*) until all frequencies for all the cells are assigned.

3) Repeat Step 2) for the next cell in the list.

TABLE III.

Simulation Results For Sivarajan's And Box's Benchmark Problems

Box		Sivarajan		Three stage			
average	best	average	Best	average	best	*LB*	Instance
443.6	442	498.3	460	427.0	427	427	1
443.4	442	498.3	447	427.0	427	427	2
533.2	533	552.8	536	533.0	533	533	3
533.0	533	550.1	533	533.0	533	533	4
381.0	381	381.0	381	381.0	381	381	5
381.0	381	381.0	381	381.0	381	381	6
533.0	533	533.0	533	533.0	533	533	7
533.0	533	533.0	533	533.0	533	533	8
271.7	270	315.9	283	258.0	258	258	9
261.6	260	294.4	269	258.0	258	258	10
309.0	309	338.5	310	309.0	309	309	11
309.0	309	330.6	310	309.0	309	309	12

TABLE IV.

Summary of Simulation Results

Problem #	*LB*	*M*	Three Stage		Moradi	
			Average Iter. No.	Convergence Rate	Average Iter. No.	Convergence Rate
1	73	73	102.3	98%	156.23	95%
2	381	381	11.13	100%	11.59	100%
3	533	533	3.3	100%	3.34	100%
4	533	533	42.3	100%	44.78	100%
5	221	221	25.51	100%	28.59	98%
6	309	309	28.52	100%	32.44	100%
7	309	309	57.33	85%	64.77	60%

III. Simulation and Discussion

A. Simulated Benchmark Problems

For the comparison, three heuristic algorithms by Sivarajan [9], Box[10] and Moradi [8] have also been implemented. The benchmark problems by Sivarajan [9], and Kunz [3] are used as simulated instances in this paper, where specifications are summarized in Tables I. In the compatibility matrix of these tables, "N_c" is the cluster size for *CCC*, "c_{ij}" is the minimum

channel distance between any pair of cells in the same cluster except for adjacent cells for *CCC*, " *acc*" is the minimum channel distance between adjacent cells for *ACC*, and "c_{ii}" is the minimum distance within the same cell for *CSC*. In the demand vector, "case 1" and "case 2" represent the corresponding channel requirements in [9]. Table II shows the specifications of the problems, which are also used by Moradi in [8]. *LB* is the lower bound on required frequencies for each problem.

B. Simulation Condision

The regular interval assignment stage is applied to the cell with 77 calls in instances #1-8 and to the cell with 45 calls in instances #11 and #12 in Sivarajan's problems because they determine the lower bond on the total number of channels. In the other instances, the first stage is not applied to any cell. In the greedy assignment stage, the iteration limit *G* max is set 100 and a maximum of ten trials using different random numbers is repeated for each greedy region until the assignment succeeds.

A total of ten runs using different random numbers is executed by my algorithm and Box's algorithm for each instance, whereas one run is executed by each of Sivarajan's eight algorithms. Table III shows the lower bound in [9] and [3], best and average solutions found by this algorithm. Here, we need to correct the lower bound on *M* in instances #1 and #2. Consider the channel assignment in istance#1, to the requests in the seven-cell cluster composed of the 77-call cell and its adjacent six cells. In this cluster, a different channel must be assigned to every call to satisfy CCC. When a channel is assigned to each call in the 77-call cell, this channel and its adjacent channels cannot be reused to the other cells in this cluster because of acc=2. Thus, the 77-call cell must occupy 229(=2×2+3×(77-2)) channels exclusively in this cluster. The remaining six cells need at least 198(=25+8+52+28+57+28) different channels.

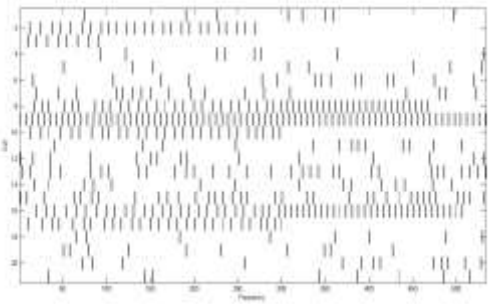

Figure 3. Distribution of the frequency for each cell with 533 frequency in instance #4.

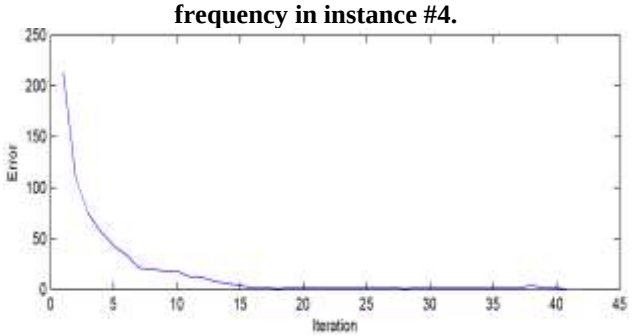

Figure 4. The number of iteration step required to convergence in instance #4.

Therefore, the channel assignment for this cluster requires at least 427(=229+198) channels. Table III indicates that my algorithm first achieves the lower bound solutions in all of the benchmark instances, whereas the existing algorithms cannot find them in instances #1, #2, #3, #9, and #10.

In Table IV, the result of this paper compared with the result in [8]. The average iteration number and the convergence rate to the solution are also shown in Table IV. The average iteration number is the average number of iterations, which are increased until E=0. Convergence rate is the probability that the network has E=0 before the maximum number iteration is reached. In these simulations, the maximum number of iterations is fixed at 500. To investigate the number of iterations and the convergence rates, 100 simulation runs were performed with different initial seed values using a random number generator for each of the seven problems. The simulation results in this paper have a smaller average iteration number and a higher convergence rate than Moradi's results. For example, in problem #1, my algorithm found the solution with 102.3 average iteration number and 98% convergence rate, but Moradi's algorithm found the solution with 156.23 average iteration number and 95% convergence rate. It demonstrates that generally, my algorithm has a better performance (i.e., smaller iteration numbers and higher convergence rates) than Moradi's algorithm. Fig.3 shows the distribution of the frequency for each cell and Fig.4 shows the number of iteration step required to convergence to solution in instance #4.

References

[1] D. Beckmann and U. Killat, "A new strategy for application of genetic algorithm to the channel assignment problem," *IEEE Trans. Veh. Techno*l., vol. 48 , pp. 1261-1269, Jul. 1999.

[2] N. Funabiki and Y. Takefuji, "A neural network parallel algorithm for channel assignment problems in cellular radio networks," *IEEE Trans. Veh. Techno*l., vol. 41 , pp. 430-437, Nov. 1992.

[3] D. Kunz, "Channel assignment for cellular radio using neural networks," *IEEE Trans. Veh. Technol*., vol. VT-40, pp. 188-193, Feb. 1991.

[4] J. –S. Kim, S. H. Park, P. W. Dowd, and N. M. Nasrabadi, " Caellular radio channel assignment using a modified Hopfield network," *IEEE Trans. Veh. Techno*l., vol. 46 , pp. 957-967, Nov. 1997.

[5] P. T. H. Chan, M. Palaniswami, and D. Everitt, "Neural Network based dynamic channel assignment for cellular mobile communication systems," *IEEE Trans. Veh. Technol*., vol. 43, pp. 279-288, May. 1994.

[6] G. Vidyarthi, A. Ngon, and I. Stojmenovic, "A hybrid channel assignment approach using an efficient evolutionary strategy in wireless mobile networks," *IEEE Trans. Veh. Techno*l., vol. 54 , pp. 1887-1895, Sept. 2005.

[7] O. Moradi, " Fixed channel assignment and neural network algorithm for channel assignment problem in cellular radio networks," *Canadian CIS Journal,* vol. 3 , pp.93-103, Nov. 2010.

[8] O. Moradi, "A hopfield neural network for channel assignment problem in cellular radio networks," *Canadian CIS Journal,* vol. 4, pp.116-129, Jan. 2011.

[9] K. N. Sivarajan, R. J. McEliece, and J. W. Letchum, " Channel assignment in cellular radio," in *Proc. 39th IEEE Veh. Technol. Conf.*, 1989, pp. 846-850.

[10] F. Box, "A heuristic technique for assigning frequencies to mobile radio nets," *IEEE Trans. Veh. Technol*., vol. VT-27, pp. 57-64, May. 1978.

[11] N. Funabiki, N. Okutani, and S. Nishikawa, "A three stage heuristic combined neural-network algorithm for channel assignment in cellular mobile systems," *IEEE Trans. Veh. Techno*l., vol. 49 , pp. 397-403, Mar. 2000.