Wireless Networks 12, 45–52, 2006
© 2006 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Frequency Insertion Strategy for Channel Assignment Problem

WON-YOUNG SHIN, SOO Y. CHANG [*], JAEWOOK LEE and CHI-HYUCK JUN
Department of Industrial Engineering, Pohang University of Science & Technology, San 31 Hyoja-dong, Pohang 790-784, Korea

Abstract. This paper presents a new heuristic method for quickly finding a good feasible solution to the channel assignment problem (CAP). Like many other greedy-type heuristics for CAP, the proposed method also assigns a frequency to a call, one at a time. Hence, the method requires computational time that increases only linear to the number of calls. However, what distinguishes the method from others is that it starts with a narrow enough frequency band so as to provoke violations of constraints that we need to comply with in order to avoid radio interference. Each violation is then resolved by inserting frequencies at the most appropriate positions so that the band of frequencies expands minimally. An extensive computational experiment using a set of randomly generated problems as well as the Philadelphia benchmark instances shows that the proposed method perform statistically better than existing methods of its kind and even yields optimum solutions to most of Philadelphia benchmark instances among which two cases are reported for the first time ever, in this paper.

Keywords: assignment, cellular system, Philadelphia benchmark

1. Introduction

Due to the rapid increase in demands for the cellular mobile systems, the radio electromagnetic frequency spectrum is becoming one of the most valuable resources of our time. The channel assignment problem (or CAP for short) deals with the optimum use of the frequency spectrum allocated to the cellular mobile systems where the service area of the system is divided into a large number of cells with customers simultaneously requesting channels for their communication.

The system must accommodate each and every customer request (or call) by assigning a channel (or a frequency spectrum) while satisfying the constraints imposed to avoid the radio interference among channels assigned to the same cell or in relatively adjacent cells. Quite a few different ways to specify such constraints are discussed in the literature [6]. Nevertheless, such constraints can be conveniently summarized by a symmetric matrix, called, the compatibility matrix denoted by $C = [c_{ij}]$. An element in the compatibility matrix, c_{ij} specifies the minimum allowed difference of the frequency spectrum assigned to an arbitrary pair of calls when one of the pair is demanded in the i-th cell and the other in the j-th cell.

If we define f_{ik} be the decision variable specifying the frequency assigned to k-th call in the i-th cell, the mathematical model for CAP can be written as;

$$\text{Minimize} \quad \max_{i,k} f_{ik} \tag{1a}$$

$$\text{subject to} \quad |f_{ik} - f_{jl}| \geq c_{ij}, \quad \text{for all } i, j, k, l\ (k \neq l, \text{if } i = j) \tag{1b}$$

$$f_{ik} \quad \text{is a non-negative integer} \tag{1c}$$

[*] Corresponding author.
E-mail: syc@postech.ac.kr

In code division multiple access (CDMA) systems, each communication channel is established using multiple radio frequencies instead of one. Furthermore, since the radio frequencies are used in such a way that the theoretical frequency reuse factor is one, CAP seems to be completely irrelevant to CDMA systems. Nevertheless, the radio interference does occur among CDMA channels. In effect, the problem of minimizing total number of CDMA channels assigned to various traffic classes requiring different levels of communication quality is shown to have combinatorial structure similar to CAP [3]. And so does the problem of maximizing radio frequency utilization in a direct sequence CDMA (DS-CDMA) system while guaranteeing a flexible quality of service (QoS) control [14]. Hence, CAP seems to have some relevance to wide variety of wireless communication systems. Unfortunately, however, CAP is known to be NP-complete [9]. Hence, it seems quite unlikely to develop an efficient optimal algorithm for the problem. However, several heuristic approaches have been proposed for CAP [2,7,8,10–13,20–23] during the last two decades. Most of these heuristics employ the common strategy of taking two stages in finding satisfactory solution, namely, the feasible assignment stage and optimal assignment stage. At the feasible assignment stage, simply a good feasible assignment satisfying all the constraints is sought. The 'frequency exhaustive strategy' (FES) and the 'requirement exhaustive strategy' (RES) in [19] and 'randomized saturation degree' (RSD) in [2] are most popular methods widely employed at this stage. Among these heuristics, RSD combined with a local search procedure is reported to yield optimum solutions to wide variety of problem instances and is considered to be the state-of-the-art heuristic up until the year 1999. At the optimal assignment stage, on the other hand, a better assignment is sought starting from the feasible assignment obtained in the feasible assignment stage. Local search and other optimization techniques such as neural network, simulated annealing, taboo search and genetic algorithm have been proposed for this stage. Most of these methods tend to require intensive computing time and the quality of the obtained solution tends to be quite sensitive to the quality of the initial solution found in the feasible assignment stage.

However, the development of a fast and efficient method for the feasible assignment stage is important from an engineering point of view, since, the computing time available for finding a good solution to CAP is never enough in practice. In fact, due to the apparent need for accommodating frequent fluctuations in communication demand patterns, the amount of allowed computing time is often barely enough for the acquisition of the feasibility let alone the optimality. Hence, in this paper, we propose a new heuristic method that finds a good feasible assignment within computing time that grows only linear to the number of calls to be accommodated.

In Section 2, we briefly summarize all relevant works for CAP in the literature. Then, we propose a new heuristic method named "frequency insertion strategy", in Section 3. In Section 4, performance of the proposed algorithm is evaluated using the Philadelphia benchmark instances as well as a set of randomly generated problems. Finally, the conclusion follows in Section 5.

2. Related works

Most of the existing heuristic algorithms take the approach of listing the calls in some order and assigning frequency to each call, one at a time, according to the predetermined heuristic scheme. Hence, the ordering of calls is the primary determinant of the solution quality in these heuristics. Hence, many researchers seek to find a good ordering scheme which would yield a good, hopefully near optimal, solution. To this end, the concept of 'the degree of cell' is developed and used, where the degree of the i-th cell, d_i, is defined as;

$$d_i = \left(\sum_{j=1}^{N} c_{ij} m_j \right) - c_{ii}, \quad 1 \le i \le N, \tag{2}$$

where N is the number of distinct cells and m_j is the number of calls requested in j-th cell. This measure is interpreted as the difficulty of assigning a frequency to a call in i-th cell. The node-degree and node-color degree ordering policies are proposed using this measure by Zoellner and Beall [25]. In the node-degree ordering, for example, cells are arranged in decreasing order of their degrees.

The frequency exhaustive strategy (or FES) and requirement exhaustive strategy (or RES) proposed by Sivarajan et al. [19] are deployed most widely as the procedure for finding an initial feasible solution. FES starts with the list of calls sorted in some ordering and assigns to each call the smallest possible frequency which does not violate the constraints. RES, on the other hand, searches all possible calls that may occupy the first frequency without violating the constraints and assigns the frequency to all those calls. Then, it identifies all possible calls that may occupy the next frequency and let them occupy the frequency. The method continues in this manner until each and every call obtains a frequency.

In an effort to find a better solution from any given feasible solution, Wang and Rushforth [24] present a local search method. This method first assigns frequencies by FES and obtains the number of frequencies required. Then, it selects and swaps the positions of a pair of calls in the current ordering and calculates the number of frequencies required using FES. If the number is lowered, keep the altered ordering and keep the original ordering, otherwise. The method repeats such swapping as many times as a predetermined limit.

Several neural network algorithms have been also proposed for solving CAP. Kunz [16] uses the Hopfield neural network and obtains optimal solutions to some special instances of CAP. Funabiki and Takefuji [5] proposed a parallel algorithm that is based on a neural network, where they used the Hysteresis McCulloch-Pitts neuron model, instead of the Hopfiled neural network. The approach, however, may be trapped at a local optimum. To overcome such trapping, Funabiki et al. [6] modified the algorithm and proposed a three-stage algorithm which combines a sequential heuristic method with the parallel neural computing model. Several researchers also developed algorithms using evolutionary search algorithms. Duque-Anton et al. [4] as well as Mathar and Mattfeldt [18] used simulated annealing to solve CAP. Lai and Coghill [17] proposed the genetic algorithm to solve CAP. Kim and Kim [15] proposed a two-phase algorithm for CAP based on the compact pattern approach. Recently, Ghosh, Sinha and Das proposed an elegant heuristic which yields optimal solutions to some Philadelphia benchmark instances [8].

3. Frequency insertion strategy

This paper proposed a new heuristic method named frequency insertion strategy. In order to demonstrate the potential merit of the proposed algorithm, we present a small example in the followings.

Suppose that we are given four cells, cell #1 through #4, and the compatibility matrix C and six calls are requested in the given cells as specified in the vector m;

$$C = \begin{bmatrix} 5 & 2 & 2 & 2 \\ 2 & 5 & 2 & 2 \\ 2 & 2 & 5 & 2 \\ 2 & 2 & 2 & 5 \end{bmatrix} \quad m = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 3 \end{bmatrix}.$$

Suppose further that the six calls are ordered as (4,4,4,1,2,3) by a certain ordering rule, where the number in the list represents the cell number that the call belongs to. Then, one can easily verify that FES would yield the assignment requiring thirteen frequencies as shown in figure 1(a). However,

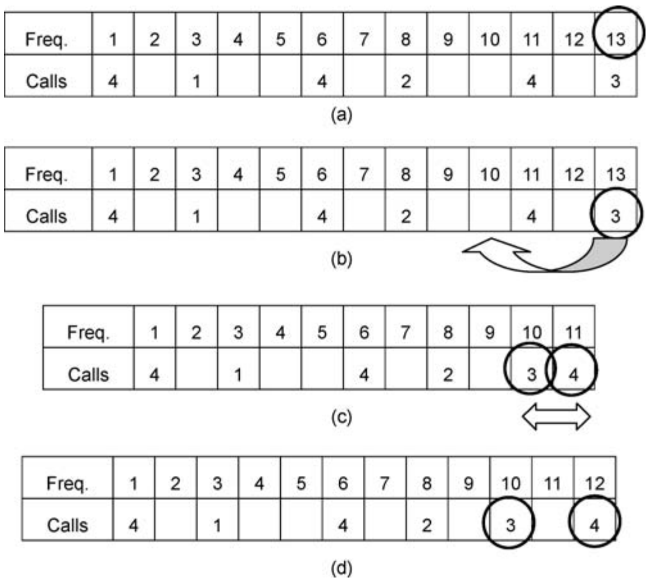


Figure 1. Illustration of the insertion strategy.

we can reduce the required number of frequencies down to twelve by taking the following three steps at the moment right before the last call, numbered 3, is assigned to the thirteenth frequency.

Step 1. Pretending as if we have only 11 frequencies, assign frequency #10 instead of #13 to the last call in the cell #3, as illustrated in figure 1(b).

Step 2. Check the violations caused by the last assignment, as illustrated in figure 1(c).

Step 3. Resolve the violation by inserting frequencies between the calls causing violations. In this case, there is only one violation and we need one frequency between the two calls causing violation. As illustrated in figure 1(d), we insert one frequency by sliding the calls assigned to the right of the inserted frequency to their right, resulting larger minimum required frequency which, in this case, becomes 12.

As suggested in the above example, the frequency insertion strategy initially permits constraint-violating assignment pretending that we do not have enough frequencies. Then, insert necessary frequencies to resolve the violation by sliding the relevant calls to their right, increasing the number of frequencies required. The name of frequency insertion strategy comes from this act of inserting frequencies.

We introduce the following notations for the formal description of our algorithm. We included some of the definitions already introduced for completeness sake.

N	the number of distinct cells
$m = (m_1, ...m_N)^T$	demand vector of calls
$C = (c_{ij})$	compatibility matrix of frequencies
a_{ik}	a symbol representing the kth call in the ith cell, $i = 1, \dots, N$ and $k = 1, \dots, m_i$.
f_{ik}	a frequency assigned to a_{ik}, $i = 1, \dots, N$ and $k = 1, \dots, m_i$.
d_i	degree of the i-th cell as defined in equation (2), $i = 1, \dots, N$
$m_{\max}$	the number of calls in the cell with the maximum degree
max_freq	maximum frequency of all assigned frequencies, $(= \max_{i,k} f_{ik})$
dist(x, y)	frequency difference between two calls x and y

Presumably, c_{ii} should be greater than c_{ij} for all j not equal to i. Hence, when a frequency, say f, is about to be assigned to a call demanded in the cell #i, the conflicts may occur only between this call and the other calls assigned to the frequencies ranging between $f - c_{ii}$ and $f + c_{ii}$. Concerning the calls already assigned in this frequency range, we define the following sets.

$A(f)$	a set of calls having frequency range between $f - c_{ii}$ and $f + c_{ii}$
$Aleft(f)$	a set of calls having frequency range between $f - c_{ii}$ and f
$Aright(f)$	a set of calls having frequency range between f and $f + c_{ii}$
$Acenter(f)$	a set of calls having frequency f

Obviously, we note that, $A(f) = Aleft(f) \cup Aright(f) \cup Acenter(f)$.

Again, when a frequency, say f, is about to be assigned to a call demanded in the cell #i, the conflicts may occur with respect to the calls in the set $Aleft(f)$, $Aright(f)$ and $Acenter(f)$. To see how severe the potential conflicts may be caused by the calls in these three sets, we calculate,

$$cleft = \max_{a \in Aleft(f)} (c_{i,a} - dist(i, a))^+$$

$$cright = \max_{a \in Aright(f)} (c_{i,a} - dist(i, a))^+$$

$$ccenter = \max_{a \in Acenter(f)} (c_{i,a} - dist(i, a))^+$$

where $(x)^+$ denotes $\max(0, x)$. From these three measures, we estimate the total potential conflicts by

$$conflict = cleft + cright + (\min\{ccenter - cleft, ccenter - cright\})^+ \tag{3}$$

Then, our algorithm can be seen as a method consisting of seven steps as specified below:

Step 1. Find the cell with the maximum degree, say the cell #i. Then, $m_i = m_{\max}$ by definition. Then, for $k = 1, \ldots, m_{\max}$, assign

$$f_{ik} = (k-1) \times c_{ii} + 1$$

resulting,

$$max\_freq = (m_{\max} - 1) \times c_{ii} + 1$$

which defined the initial variety of frequencies that we use.

Step 2. List the calls except the ones handled in Step 1, in the node-degree or some other ordering.

Step 3. Select the first call in the list.

Step 4. Calculate the conflict assuming that the selected call is assigned with frequency f ($f = 1, \ldots, max\_freq$). Find the frequency with the minimum conflict (say $f_{\min}$).

Step 5. Assign the frequency having the minimum conflict to the selected call. If the minimum conflict is zero, then go to Step 7. Otherwise, go to Step 6.

Step 6. **[Frequency Insertion Step]** Resolve conflicts by;

Case 1 cleft > cright

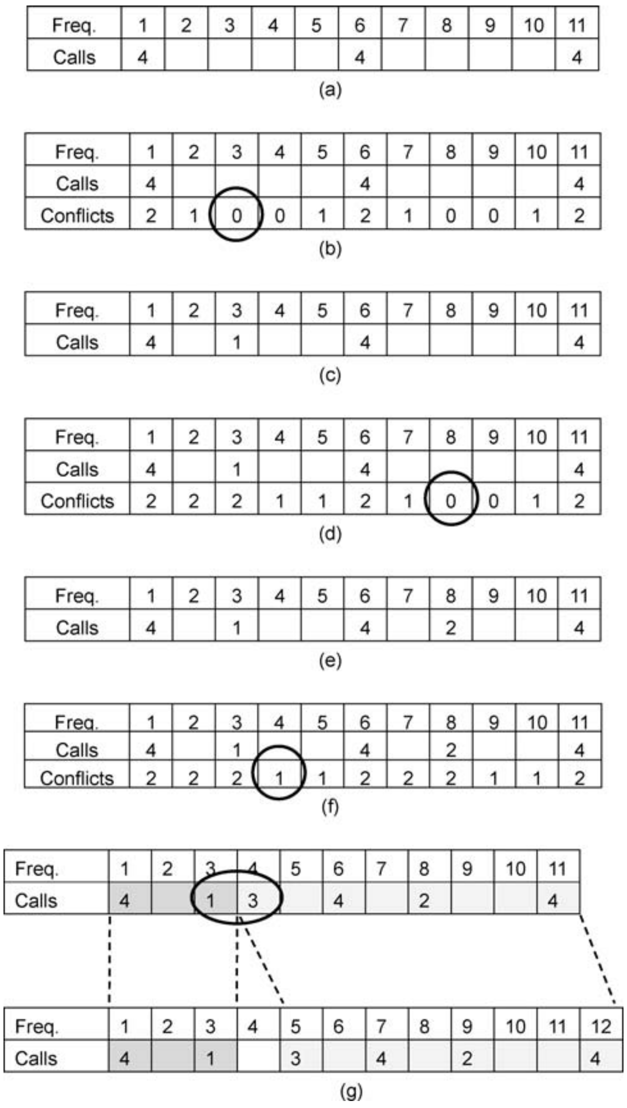
1. Change all previously assigned frequencies greater than f by adding max{*cleft*, *ccenter*} + *cright*.
2. Assign $f_{\min}$ + max{*cleft*, *ccenter*} to the selected call.

Case 2 cleft ≤ cright

1. Change all previously assigned frequencies greater than or equal to f by adding max{*cright*, *ccenter*}+ *cleft* to all these frequencies
2. Assign $f_{\min}$ + *cleft* to the selected call.

Step 7. If the call is last in the call list, stop. Otherwise, update the call list by removing the assigned call, update *max_freq*, and go to Step 3.

In Step 4, ties may occur in finding the frequency $f_{\min}$. Our computational experiment suggests that the way of breaking such ties may affect the solution quality. Many different policies can be taken for breaking the ties. In fact, the simplest policy would be the *smallest-frequency first* where we simply pick the smallest frequency when tie occurs. However, we found that the policy that we call, the *modified smallest-frequency first* policy seem to yield the best results. Under this policy, we remember the frequency chosen in the previous step, say f which is initially set to be one. Then, when the tie occurs, we select the first frequency in the range $[f + 1, max\_freq]$, where the tie occurs. If no such frequency is found in the range, we do the same with the range $[1, f]$. Our experiments show that the *modified smallest-frequency first* policy tends to yield superior solution quality, especially when node-degree ordering is used.



Freq.	1	2	3	4	5	6	7	8	9	10	11
Calls	4					4					4

(a)

Freq.	1	2	3	4	5	6	7	8	9	10	11
Calls	4					4					4
Conflicts	2	1	0	0	1	2	1	0	0	1	2

(b)

Freq.	1	2	3	4	5	6	7	8	9	10	11
Calls	4		1			4					4

(c)

Freq.	1	2	3	4	5	6	7	8	9	10	11
Calls	4		1			4					4
Conflicts	2	2	2	1	1	2	1	0	0	1	2

(d)

Freq.	1	2	3	4	5	6	7	8	9	10	11
Calls	4		1			4		2			4

(e)

Freq.	1	2	3	4	5	6	7	8	9	10	11
Calls	4		1			4		2			4
Conflicts	2	2	2	1	1	2	2	2	1	1	2

(f)

Freq.	1	2	3	4	5	6	7	8	9	10	11
Calls	4		1	3		4		2			4

Freq.	1	2	3	4	5	6	7	8	9	10	11	12
Calls	4		1		3		4		2			4

(g)

Figure 2. Algorithm executed on the example.

We now revisit the same example that we used and illustrate how the algorithm proceeds. Suppose that we use the same ordering of calls, as (4,4,4,1,2,3).

Iteration 1

Step 1. As in figure 2(a), assign frequencies to the three calls in the cell #4 yielding *max_freq* = 11.

Step 2. List the remaining calls as $(a_{11}, a_{21}, a_{31}) = (1, 2, 3)$.

Step 3. Select the first call ($=a_{11}$) from the list.

Step 4. Calculate conflicts for frequency #1 through frequency #11. Find the frequency with the minimum conflict (which, in this case, $f_{\min} = 3$, as shown in figure 2(b)).

Step 5. As shown in figure 2(c), assign the frequency #3 to a_{11}.

Step 7. Update the list as $(a_{21}, a_{31}) = (2, 3)$ and go to Step 3.

Iteration 2

Step 3. Select the first call ($=a_{21}$) in the list

Step 4. Calculate conflicts for frequency #1 through frequency #11. Find the frequency with the minimum conflict (which, in this case, $f_{\min} = 8$ as shown in figure 2(d)).

Step 5. As shown in figure 2(e), assign frequency # 8 to a_{21}

Step 7. Update the list as $(a_{31}) = (3)$ and go to Step 3.

Iteration 3

Step 3. Select the first call ($=a_{31}$) in the list

Step 4. Calculate conflicts for frequency #1 through frequency #11. Find the frequency with the minimum conflict (which, in this case, $f_{\min} = 4$ as shown in figure 2(f)).

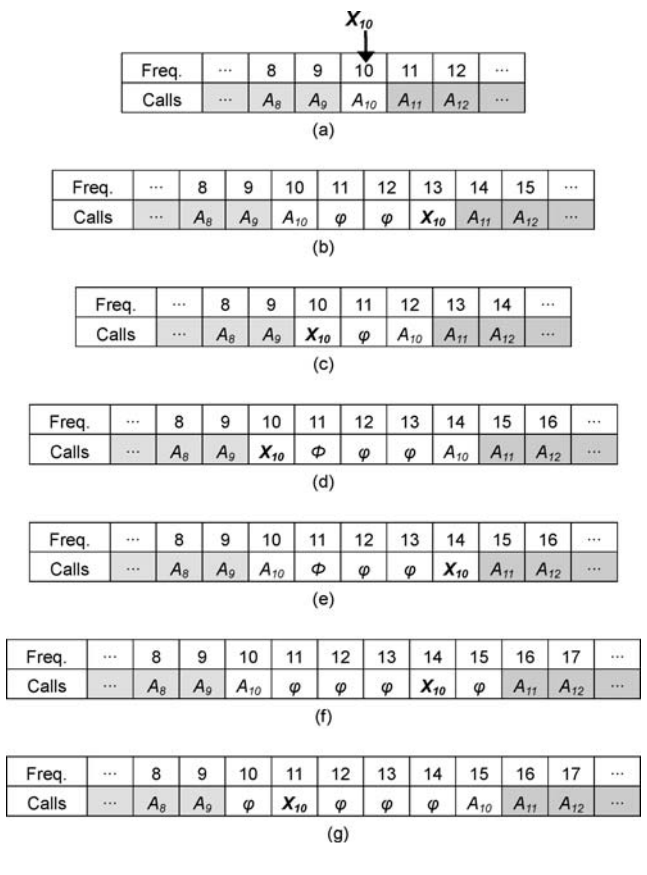
Step 5. Assign frequency #4 to a_{31}.

Step 6. There is constraint violation between a_{11} and a_{31}. So we insert unassigned frequency between a_{11} and a_{31}, as shown in figure 2(g), and adjust the assignments in the higher frequencies accordingly. Note that here (*cleft, cright, ccenter*) = (1, 0, 0) and so, Case 1 applies.

Step 7. Because the updated call list is empty, exit the algorithm.

Figure 3 shows a set of examples which enumerate and illustrate how the step 6 (**Frequency Insertion Step**) would operate in different situations. For our illustrative purposes, we assume that the frequency $f_{\min} = 10$ is about to be assigned to the call X_{10} and the frequencies are assigned to the calls as shown in figure 3(a), where A_k denotes the set of calls to which frequency k is assigned and ϕ denotes the empty set.

First, consider when (*cleft*, *cright*, *ccenter*) = (3, 0, 0). In this case, three frequencies are inserted and the frequency #13 is assigned to X_{10}, as shown in figure 3(b). If (*cleft*, *cright*, *ccenter*) = (0, 2, 0), two frequencies are inserted and frequency #10 is assigned to X_{10}, as shown in figure 3(c). If (*cleft, cright, ccenter*) = (0, 0, 4), (0, 2, 4) or (0, 4, 2). Then, four frequencies are inserted and the frequency #10 is assigned to X_{10}, as shown in figure 3(d). If (*cleft, cright, ccenter*) = (3, 0, 4) or (4, 0, 3). Then, four frequencies are inserted and the frequency #14 is assigned to X_{10}, as shown in figure 3(e). If (*cleft, cright, ccenter*) = (2, 1, 4) or (4, 1, 2). Then, four frequencies are inserted and the frequency #14 is assigned to X_{10}, as shown in figure 3(f). If (*cleft, cright, ccenter*) = (1, 2, 4) or (1, 4, 2). Then, five frequencies are inserted and the frequency #11 is assigned to X_{10}, as shown in figure 3(g).


X_{10}

Freq.	···	8	9	10	11	12	···
Calls	···	A_8	A_9	A_{10}	A_{11}	A_{12}	···

(a)

Freq.	···	8	9	10	11	12	13	14	15	···
Calls	···	A_8	A_9	A_{10}	φ	φ	X_{10}	A_{11}	A_{12}	···

(b)

Freq.	···	8	9	10	11	12	13	14	···
Calls	···	A_8	A_9	X_{10}	φ	A_{10}	A_{11}	A_{12}	···

(c)

Freq.	···	8	9	10	11	12	13	14	15	16	···
Calls	···	A_8	A_9	X_{10}	Φ	φ	φ	A_{10}	A_{11}	A_{12}	···

(d)

Freq.	···	8	9	10	11	12	13	14	15	16	···
Calls	···	A_8	A_9	A_{10}	Φ	φ	φ	X_{10}	A_{11}	A_{12}	···

(e)

Freq.	···	8	9	10	11	12	13	14	15	16	17	···
Calls	···	A_8	A_9	A_{10}	φ	φ	φ	X_{10}	φ	A_{11}	A_{12}	···

(f)

Freq.	···	8	9	10	11	12	13	14	15	16	17	···
Calls	···	A_8	A_9	φ	X_{10}	φ	φ	φ	A_{10}	A_{11}	A_{12}	···

(g)

Figure 3. Various cases handled at the Frequency Insertion Step.

4. Computational experiments

Philadelphia benchmark instance, introduced by Anderson [1] consisting of nine problems, is the most widely used problem set for evaluating the algorithms for CAP. The Philadelphia instances are characterized by 21 hexagons denoting the cells of a cellular phone network around Philadelphia (see figure 4(a)). figure 4(b) shows the call demand for the first instance. Table 1 contains the demand vectors of all instances.

Table 1
Call demands of Philadelphia instances.

.	Demand vector
#1	(8,25,8,8,8,15,18,52,77,28,13,15,31,15,36,57,28,8,10,13,8)
#2	(8,25,8,8,8,15,18,52,77,28,13,15,31,15,36,57,28,8,10,13,8)
#3	(5,5,5,8,12,25,30,25,30,40,40,45,20,30,25,15,15,30,20,20,25)
#4	(5,5,5,8,12,25,30,25,30,40,40,45,20,30,25,15,15,30,20,20,25)
#5	(20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20)
#6	(20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20)
#7	(16,50,16,16,16,30,36,104,154,56,26,30,62,30,72,114,56,16,20,26,16)
#8	(8,25,8,8,8,15,18,52,77,28,13,15,31,15,36,57,28,8,10,13,8)
#9	(32,100,32,32,32,60,72,208,308,112,52,60,124,60,144,228,112,32,40, 52,32)

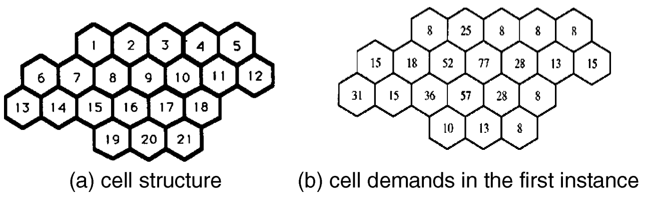


Figure 4. Cell structure in Philadelphia benchmark instances.

Table 2
Frequency interference constraints of the Philadelphia instances.

Instances	Reuse distances
#1, #3, #5, #7, #9	$(2\sqrt{3},\ \sqrt{3},\ 1,\ 1,\ 1,\ 0)$
#2, #4, #6	$(\sqrt{7},\ \sqrt{3},\ 1,\ 1,\ 1,\ 0)$
#8	$(2\sqrt{3},\ 2,\ 1,\ 1,\ 1,\ 0)$

Table 2 presents various frequency interference constraints used in Philadelphia instances. For instance, the case of $(2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0)$ means that two calls assigned to a same frequency must be at least $2\sqrt{3}$ unit distance (or 4 cells) apart to avoid interference, two calls assigned to adjacent frequencies must be at least $\sqrt{3}$ unit distance (or 2 cells) apart to avoid interference and so on. Hence, in this case, the same frequency can only be used to a pair of calls when they are located in at least 5 unit distance (or 5 cells) apart to avoid interference. The other cases of frequency interference constraints should be interpreted in the same manner.

A lot of research has been devoted to lower bounds and upper bounds on the required number of frequencies for the Philadelphia instances. Table 3 summarizes the best results reported so far on the Philadelphia instances.

Table 4 compares the quality of the solutions obtained from three known heuristics (FES, RES, RSD) and our heuristic (FIS). Since, the quality of solution yield from RSD does depend on the amount of computing time, we limit the computing time for RSD be one second for couple of reasons. One second is, in fact, the time limit used when the performance of RSD on Philadelphia instances is measured and reported in [2]. Also, one second of computing time for RSD is considered to be enough since it is about ten times the average computing time used for FES, RES, or FIS to obtain a solution for any Philadelphia instance.

As summarized in Table 4, FIS yields better solution than FES and RES as well as RSD in seven out of nine cases and finds optimum solutions for four out of nine cases. The average computing time for FIS is about 0.1 second. It is worthwhile to note that the optimum solutions to the instance #7 and #9 are found and reported for the first time ever in this paper, as far as we know.

In order to find out about more about the performance of FIS relative to FES, RES and RSD, we have conducted more computational experiments with randomly generated problem instances. For our experiment, thirty cases are generated and solved. Table 5 shows 10 demand vector randomly generated from discrete uniform distribution ranging from 1 to 500, while fixing number of the cells being equal to 21. Then we run all three heuristics with three different frequency interference constraints as being specified by reuse distance vector of $(2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0)$, $(\sqrt{7}, \sqrt{3}, 1, 1, 1, 0)$ and $(2\sqrt{3}, 2, 1, 1, 1, 0)$.

Table 6 shows the result of our experiment. The computing time for RSD is limited to be no more than 10 seconds, since the average computing time for FES, RES as well as FIS is observed to be much less than one second. Our algorithm, FIS, yields the best solution in all 30 cases. Furthermore, on the average, we observed that the solution from FIS requires

Table 3
Summary of results reported so far on the Philadelphia instances.

	Lower bounds			Known	Upper bounds				
Inst.	[4,5]	[6,9]	[11]	optimum	[6]	[6]	[12]	[9]	[13]
#1	426	426	426	426	428	428	426	426	432
#2	426	426	426	426	429	438	426	–	–
#3	–	257	252	257	269	260	258	257	263
#4	252	252	252	252	257	259	253	252	–
#5	–	239	177	239	240	239	239	–	–
#6	177	178	177	178–188	188	200	198	–	–
#7	–	855	855	855–856	858	858	856	–	–
#8	–	524	427	524–527	535	546	527	–	–
#9	–	1713	1713	1713–1724	1724	1724	–	–	–

Note: The shaded cells represent the optimum values.

Table 4
Solution comparison between three heuristics.

Inst.	FES	RES	RSD*	FIS	Optimal value (range)
#1	542	520	484	**426**	426
#2	542	465	485	**426**	426
#3	345	**295**	298	298	257
#4	345	292	292	**263**	252
#5	295	294	292	**268**	239
#6	293	218	**199**	222	178–188
#7	1087	1045	1009	**855**	855
#8	654	543	625	**538**	524–527
#9	2177	2095	1955	**1713**	1713

*computing time limit for RSD was one second.
The shaded cells represent the optimum values.

Table 5
Call demand vector in ten more instances.

#	Demand vector
1	(53,499,106,240,459,354,226,271,232,168,242,406,226,199, 97,211,135,345,66,220,162)
2	(182,251,417,304,484,208,152,131,486,371,369,65,121,202, 191,425,100,393,12,213,169)
3	(49,7,250,303,421,180,27,224,331,487,198,274,198,187,402, 472,123,136,122,193,87)
4	(37,160,455,325,147,162,50,421,37,323,216,324,131,23,129, 248,63,205,224,22,219)
5	(447,429,236,400,339,95,168,395,343,487,422,120,283,266,62, 330,113,235,196,69,433)
6	(105,176,377,232,329,156,133,262,466,434,307,70,220,42,392, 251,401,279,489,314,424)
7	(251,138,396,201,382,144,34,365,447,491,233,147,98,240,317, 490,39,328,80,424,170)
8	(35,123,14,240,237,27,320,429,405,200,21,403,73,314,56,239, 304,155,115,339,89)
9	(255,167,95,290,68,186,229,322,472,327,304,258,444,34,374, 354,134,156,3,318,462)
10	(236,377,40,310,426,455,295,92,223,209,356,221,302,106,416, 354,497,278,455,463,387)

Table 6
Results of additional problems with 3 types of reuse distance

	$d = (2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0)$				$d = (\sqrt{7}, \sqrt{3}, 1, 1, 1, 0)$				$d = (2\sqrt{3}, 2, 1, 1, 1, 0)$			
#	FES	RES	RSD	FIS	FES	RES	RSD	FIS	FES	RES	RSD	FIS
1	3463	3211	3238	3076	3651	3071	3092	2697	4117	4027	3750	3750
2	4333	4226	4262	3703	4335	3621	3970	3398	5228	4842	4990	3703
3	3555	3679	3256	3111	3320	3344	3323	2870	4692	4217	4023	3902
4	3237	3263	3101	2888	3235	2722	3045	2467	3774	3409	3539	3223
5	4543	5170	4369	3891	4888	4087	4069	3536	5881	4954	5136	4772
6	5218	5599	4577	4301	4172	3966	4050	3770	5698	5047	5563	4301
7	4209	4047	4434	3984	4211	3681	3734	3365	5780	5051	5209	4649
8	3264	3143	3207	2806	3214	2797	2963	2585	4216	3535	3900	3478
9	4485	4095	3970	3640	3377	3636	3147	2960	4992	4361	3889	4267
10	4660	4412	4352	3927	5080	4338	4103	3541	4980	5121	5304	4753

about ten percent less frequencies than the solutions obtained from other method.

5. Conclusions

Our computational experiment using a set of randomly generated problems as well as the Philadelphia benchmark instances shows that the proposed heuristic performs statistically better than existing methods of its kind and finds optimum solutions to a number of Philadelphia benchmark instances, among which two cases are reported for the first time ever, in this paper. However, the quality of the solution obtained from the proposed heuristic tends be sensitive to the initial ordering of calls, since the method is after all a sequential greedy heuristic. For this reason, we are actively searching for the policy for ordering calls best suited for our algorithm. Nevertheless, our heuristic, as it stands now, seems to well present itself as being an attractive alternative for generating a good initial solution for other existing sophisticated local search methods developed for the channel assignment problem. Also, since the proposed method never requires heavy computing time, the proposed method may well suited for more realistic circumstances where on-line and real-time reallocation of radio frequency resources must be made continuously.

References

[1] L.G. Anderson, A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system, IEEE Transactions on Communications 21 (1973) 1294–1301.

[2] R. Battiti, A. Bertossi and D. Cavallaro, A randomized saturation degree heuristic for channel assignment in cellular radio networks, IEEE Transactions on Vehicular Technology 50 (2001) 364–374.

[3] M.S. Do, Y. Park and J. Lee, Channel assignment with QoS guarantees for a multiclass multicode CDMA system, IEEE Transactions on Vehicular Technology 51(5) (2002) 935–948.

[4] M. Duque-Antón, D. Kunz and B. Rüber, Channel assignment for cellular radio using simulated annealing, IEEE Transactions on Vehicular Technology 42 (1993) 14–21.

[5] N. Funabiki and Y. Takefuji, A neural network parallel algorithm for channel assignment problems in cellular radio networks, IEEE Transactions on Vehicular Technology 41 (1992) 430–437.

[6] N. Funabiki, N. Okutani and S. Nishikawa, A three-stage heuristic combined neural-network algorithm for channel assignment in cellular mobile systems, IEEE Transactions on Vehicular Technology 49 (2000) 397–403.

[7] S.C. Ghosh and B.P. Sinha, Channel assignment using genetic algorithm based on geometric symmetry, IEEE Transactions on Vehicular Technology 52 (2003) 860–875.

[8] S.C. Ghosh, B.P. Sinha and N. Das, An efficient channel assignment technique for hexagonal cellular networks, in: *Proc. IEEE Int'l Symp. on Parallel Architectures, Algorithms, and Networks (ISPAN22)* (2002).

[9] W.K. Hale, Frequency assignment: Theory and applications, Proceedings of IEEE 68 (1980) 1497–1514.

[10] Z. He, Y. Zhang, C. Wei and J. Wang, A multistage self-organizing algorithm combined transiently chaotic neural network for cellular channel assignment, IEEE Transactions on Vehicular Technology 51 (2002) 1386–1396.

[11] S. Hurley, D.H. Smith and S.U. Thiel, A system for discrete channel frequency assignment, Radio Science 32 (1997) 1921–1939.

[12] J. Janssen and K. Kilakos, An optimal solution to the Philadelphia channel assignment problem, Technical Report LSE-CDAM-96-16, Centre for Discrete and Applicable Mathematics, London School of Economics & Political Science, London (1996).

[13] J. Janssen and K. Kilakos, Polyhedral analysis of channel assignment problems: (i) tours, Technical Report LSE-CDAM-96-17, Centre for Discrete and Applicable Mathematics, London School of Economics & Political Science, London (1996).

[14] J.H. Kim, T.S. Kim, Y.W. Kim and D.K. Sung, Hybrid channel assignment scheme for accommodating voice/data traffic in DS-CDMA cellular systems, IEEE Transactions on Vehicular Technology 49(5) (2000) 1566–1577.

[15] S. Kim and S.L. Kim, A two-phase algorithm for frequency assignment in cellular mobile systems, IEEE Transactions on Vehicular Technology 43 (1994) 542–548.

[16] D. Kunz, Channel assignment for cellular radio using neural networks, IEEE Transactions on Vehicular Technology 40 (1991) 188–193.

[17] W.K. Lai and C.G. Coghill, Channel assignment through evolutionary optimization, IEEE Transactions on Vehicular Technology 45 (1996) 91–95.

[18] R. Mathar and J. Mattfeldt, Channel assignment in cellular radio networks, IEEE Transactions on Vehicular Technology 42 (1993) 647–656.

[19] K.N. Sivarajan, R.J. McEliece and J.K. Ketchum, Channel assignment in cellular radio, in: *Proceedings of the 39th IEEE Vehicular Technology Conference* (1989) pp. 846–850.

[20] D.H. Smith, S. Hurley and S.U. Thiel, Improving heuristics for the frequency assignment problem, European Journal of Operational Research 107 (1998) 76–86.
[21] C.W. Sung and W.S. Wong, Sequential packing algorithm for channel assignment under cochannel and adjacent channel interference constraint, IEEE Transactions on Vehicular Technology 46 (1997) 676–685.
[22] C.W. Sung and K.W. Shum, Channel assignment and layer selection in hierarchical cellular system with fuzzy control, IEEE Transactions on Vehicular Technology 50 (2001) 657–663.
[23] C. Valenzuela, S. Hurley and D.H. Smith, A permutation based genetic algorithm for minimum span frequency assignment, Lecture Notes in Computer Science 1498 (1998) 907–916.
[24] W. Wang and C.K. Rushforth, An adaptive local-search algorithm for the channel-assignment problem (CAP), IEEE Transactions on Vehicular Technology 45 (1996) 459–466.
[25] J.A. Zoellner and C.L. Beall, A breakthrough in spectrum conserving frequency assignment technology, IEEE Transactions on Electromagnetic Compatibility, EMC-19(3) (1977) 313–319.

Won-Young Shin was born in Busan, Korea in 1978. He received B.S. in industrial engineering from Pohang University of Science and Technology (POSTECH) in 2001 and M.S in operation research and applied statistics from POSTECH in 2003. Since 2003 he has been a researcher of Agency for Defense Development (ADD) in Korea. He is interested in optimization of communication system and applied statistics.
E-mail: wonyoung@postech.ac.kr

Soo Y. Chang is an associate professor in the Department of Industrial Engineering at Pohang University of Science and Technology (POSTECH), Pohang, Korea. He teaches linear programming, discrete optimization, network flows and operations research courses. His research interests include mathematical programming and scheduling. He has published in several journals including Discrete Applied Mathematics, Computers and Mathematics with Application, IIE Transactions, International Journal of Production Research, and so on. He is a member of Korean IIE, and ORMSS.
E-mail: syc@postech.ac.kr

Jaewook Lee is an assistant professor in the Department of Industrial Engineering at Pohang University of Science and Technology (POSTECH), Pohang, Korea. He received the B.S. degree in mathematics with honors from Seoul National University, and the Ph.D. degree from Cornell University in applied mathematics in 1993 and 1999, respectively. He is currently an assistant professor in the department of industrial engineering at the Pohang University of Science and Technology (POSTECH). His research interests include nonlinear systems, neural networks, nonlinear optimization, and their applications to data mining and financial engineering.
E-mail: jaewookl@postech.ac.kr

Chi-Hyuck Jun was born in Seoul, Korea in 1954. He received B.S. in mineral and petroleum engineering from Seoul National University in 1977, M.S. in industrial engineering from Korea Advanced Institute of Science and Technology in 1979 and Ph.D. in operations research from University of California, Berkeley, in 1986. Since 1987 he has been with the department of industrial engineering, Pohang University of Science and Technology (POSTECH) and he is now a professor and the department head. He is interested in performance analysis of communication and production systems. He has published in several journals including IIE Transactions, IEEE Transactions, Queueing Systems and Chemometrics and Intelligent Laboratory Systems. He is a member of IEEE, INFORMS and ASQ.
E-mail: chjun@postech.ac.kr