A Particle Swarm Optimization-based Approach towards the Solution of the Dynamic Channel Assignment Problem in Mobile Cellular Networks

Mithun Chakraborty1, Rini Chowdhury2, Joydeep Basu3, R. Janarthanan4, Amit Konar5

1, 2, 3, 5 Dept. Of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata – 700 032.
4 Dept. of IT, Jaya Engineering College, Chennai.

1 mithun.chakra108@gmail.com, 2 chowdhury.rini@gmail.com, 3 joydeep_etce@yahoo.co.in,
4 srmjana_73@yahoo.com, 5 konaramit@yahoo.co.in

Abstract-The limited availability of channel resources offers a bottleneck on the allocation of channels to subscribers in wireless mobile communication systems. This paper provides a novel technique for dynamic channel assignment in mobile cellular networks by constructing a suitable objective function, the optimization of which yields a solution to the problem. The objective function has been optimized using Particle Swarm Optimization (PSO) Algorithm. The proposed channel allocation scheme has been studied with six typical benchmarks, available in the literature, and the results are promising. For instance, call rejection probability evaluated in the present context is as low as 20% even for the busiest network considered.

I. INTRODUCTION

The entire geographical region covered by a mobile service provider is divided into a number of smaller service areas called cells. Each cell has a base station, provided with radio transceiver equipment, and several mobile terminals (subscribers). Whenever a terminal in a cell makes a call, it must be provided with a channel for the requested communication to be established. For example, in FDMA (Frequency Division Multiple Access) scheme, a channel is nothing but a pair of frequency bands (forward link + uplink) on the available radio spectrum. The problem of assigning channels to callers while maintaining an optimum quality of service is called the Channel Assignment Problem (CAP). The commonly used schemes employed for solving the CAP are:

- **Fixed Channel Assignment** (FCA): In this simple but non-adaptive scheme [6], each cell in the network is permanently assigned a set of channels, avoiding interference.

- **Dynamic Channel Assignment** (DCA): In this scheme [5], all the channels are kept in a central pool so that every cell has access to every channel; channels are assigned to calls as and when they arrive, depending on certain conditions. DCA may either be centralized, when a single central controller, to which all base stations are connected, decides the fate of all calls in the entire network, or it may be distributed, when the relevant decision is taken by the concerned base station on the basis of available information on the neighbourhood.

- **Hybrid Channel Assignment** (HCA): In this scheme [8], the set of available channels is partitioned into two subsets of convenient sizes, one of which is allocated to the network according to the FCA scheme and the other as per DCA scheme.

In this paper, we propose a DCA scheme that employs Particle Swarm Optimization Algorithm.

In the DCA scheme, whenever a fresh call arrives, the entire network should, ideally, be subjected to rearrangement of channel assignment to improve the quality of service and reduce the call blocking probability; however, as this is highly time-consuming and computationally complex, the rearrangement process is restricted to the cell involved in new call arrival [1].

II. THE CELLULAR MODEL AND THE CHANNEL ASSIGNMENT PROBLEM (CAP)

A. Constraints in CAP

In any cellular network, whenever two cells use the same channel or when two cells use channels adjacent to each other on the spectrum or when two channels are assigned to the same cell, interference is bound to occur; these types of interference are called Co-channel Interference, Adjacent Channel Interference and Co-site Interference respectively [5]. They lower the signal-to-noise ratio at the receiving end, leading to the deterioration of system performance. Though the computation of the actual level of interference is tough, primarily owing to its dependence on the topology of the real environment, experiments show that the effect of interference is reasonably low if the following three constraints are satisfied:

- **Co-channel Constraint** (CCC): The same channel cannot be simultaneously allocated to a pair of cells unless there is a minimum geographical separation between them.

- **Adjacent Channel Constraint** (ACC): Adjacent channels cannot be assigned to a pair of cells unless there is a minimum distance between them.

- **Co-site Constraint** (CSC): A pair of channels can be employed in the same cell only if there is a minimum separation in frequency between them.
These constraints are called Electromagnetic Compatibility Constraints, which together with the traffic demand constraint, are known as hard constraints.

Apart from the hard constraints, another set of constraints called soft constraints is also considered, which may be described as follows:

The packing condition requires that a channel, in use in one cell, should be reused in another cell as close as possible (but obviously not interfering with the former) so that the number of channels used by the network is minimal, thereby lowering the probability of future call blocking in other cells.

The resonance condition tries to ensure that same channels are assigned to cells belonging to the same reuse scheme [12], as far as possible.

Another soft constraint is that, when a call arrives, minimum number of channel reassignment operations should be performed because excessive reallocation in a cell may lead to increase in blocking probability.

A solution to the CAP must satisfy the hard constraints whereas a soft constraint may be violated; the latter only helps maximise the utilisation of resources and/or improve the quality of service.

Apart from the traffic demand constraint, the only other hard constraint that we have taken into account is Co-channel Constraint; other sources of interference are assumed to be absent, as done by Battiti et al [2], Vidyarthi et al [1].

B. Assumptions on the cellular model and call arrival
1. The geographical model is a set of contiguous, non-overlapping cells assumed to be of hexagonal shape and collectively forming a parallelogram [1], [2].

![Figure 1. Cellular network model](Vidyarthi et al)

2. We consider not the start-up situation but the situation at a certain intermediate time-instant \( t \) when a certain number of calls is already being served by the network.
3. At time \( t \), only one new call arrives at only one cell, called the host cell, all other conditions in the entire network remaining unaltered.
4. We set a minimum “reuse distance” \( RD \), which represents the minimum allowable normalised distance between two cells which may use the same channel at the same time. This defines an “interference region” extending up to \( RD - 1 \) cells in all directions from the host cell [1].

III. Problem Representation

A. The Allocation Matrix and the solution vector

Let \( N_c \) denote the number of cells in a system and \( N_{ch} \) be the total number of channels. Then, for this system, an \( N_c \times N_{ch} \) matrix \( A \), called the allocation matrix [1], may be defined whose \((i,j)\)-th element is given by

\[
A_{ij} = \begin{cases} 
1 & \text{if channel } j \text{ is currently being used in cell } i, \\
0 & \text{otherwise}
\end{cases}
\]

\( \forall i = 1, 2, \ldots, N_c, j = 1, 2, \ldots, N_{ch} \).

We assume that the new call demand is placed at cell \( k \) which is already serving \( \text{traf}(k) - 1 \) calls where \( \text{traf}(k) \) denotes the total traffic load (ongoing + incoming) in cell \( k \) at time \( t \); there are no pending calls and no ongoing call is terminated in the entire network. Our problem is to assign an available channel to the incoming call with possible reassignment of channels to the calls in progress in cell \( k \).

A candidate solution is a vector of channel numbers, denoted by \( X \), of length \( \text{traf}(k) \), which represents a set of channels assigned to the \( \text{traf}(k) \) number of calls at \( k \) at the time-instant concerned [1].

For each candidate solution, the information about the new channel allocation in cell \( k \) is stored in a \( \{0, 1\}\)-valued vector of length \( N_{ch} \), denoted by \( V \); e.g. if \( k = 4, N_{ch} = 10 \) and a candidate solution is \([1 2 7]\), then \( V = [1 1 0 0 0 0 1 0 0 0] \). Clearly, \( V \) is a one-to-one function of \( X \).

B. Formulation of the fitness function

We use the objective function or fitness function, suggested by Battiti et al [2], whose minimum value is likely to correspond to a good solution \((X_{best})\) for the new channel allocation in cell \( k \). This function is a linear combination of the following terms:

1. \( f_1(X) = \sum_{j=1}^{N_{ch}} \sum_{i=1}^{N_c} V_i A_{ij} \text{interf}(i,k) \)

   where \( V_i \) denotes the \( i \)th element of \( V \) and \( \text{interf}(i,k) \) is a function which returns 1 if the cells \( i \) and \( k \) interfere, otherwise it returns 0. This term contributes 1 for each cell, interfering with \( k \), which uses a channel employed in \( k \). It thus ensures that solutions with no interference give better (smaller) fitness values.

2. \( f_2(X) = -\sum_{j=1}^{N_{ch}} \sum_{i=1}^{N_c} V_j A_{ij} \frac{1-\text{interf}(i,k)}{d_{ik}} \)

   where \( d_{ik} \) is the normalised Euclidean distance between the centers of the cells \( i \) and \( k \). This term takes care of the packing condition. Clearly \( \text{interf}(i,k) = 0 \) if \( d_{ik} \geq RD \).

3. \( f_3(X) = -\sum_{j=1}^{N_c} V_j A_{kj} \)

   which subtracts 1 whenever a channel already being used by cell \( k \), before the arrival of the new call, is considered in the candidate solution (i.e. in the new configuration) so that
mobile terminal being served need not change its channel too often.

4. \[ f_4(X) = \sum_{j=1}^{N_x} \sum_{k=1}^{N_y} V_j A_k \{ 1 - \text{res}(i,k) \} \]

where \( \text{res}(i,k) \) is a function which returns 1 if cells \( i \) and \( k \) belong to the same “reuse scheme”, else it returns 0.

N.B. : The term of the form \( \left[ \text{trf}(k) - \sum_{j=1}^{N_x} V_j \right]^2 \) used by Battiti et al, has been discarded because in our scheme, this term is always equal to zero.

Finally, the fitness function \( F(X) \) is given by

\[ F(X) = W_1 f_1(X) + W_2 f_2(X) + W_3 f_3(X) + W_4 f_4(X) \]

where \( W_1, W_2, W_3, W_4 \) are weights that determine the importance of various terms. Clearly, \( f_i(X) \) accounts for the hard constraint, which should be given primacy over the other terms that are associated with the soft constraints. Thus, we use the same set of coefficient values as Battiti et al [2] i.e. \( W_1=7000, W_2=1.2625, W_3=0.01, W_4=4.17625. \) Our task now is to determine the minimum of this fitness function.

IV. REVIEW ON PARTICLE SWARM OPTIMIZATION (PSO) ALGORITHM

A. The Optimization Problem

The optimization problem consists in determining the global optimum (in our case, minimum) of a continuous real-valued function of \( n \) independent variables \( x_1, x_2, x_3, \ldots, x_n \), mathematically represented as \( f(X) \), where \( \vec{X} = (x_1, x_2, x_3, \ldots, x_n) \) is called the parameter vector. The task of any optimization algorithm is to search the \( n \)-dimensional hyperspace to locate a particular point with position-vector \( \vec{X}_o \) such that \( f(\vec{X}_o) \) is the global optimum of \( f(X) \).

B. The Standard PSO Algorithm

PSO [12], [13], [14] developed by Eberhart & Kennedy, is in principle a multi-agent parallel search technique. We begin with a population or swarm consisting of a convenient number, say \( m \), of particles — conceptual entities that “fly” through the multi-dimensional search space as the algorithm progresses through discrete (unit) time-steps \( t = 0, 1, 2, \ldots \), the population-size \( m \) remaining constant.

In the standard PSO algorithm, each particle \( P \) has two state variables: its current position \( \vec{X}_i(t) = [X_{i,1}(t), X_{i,2}(t), \ldots, X_{i,n}(t)] \) and its current velocity \( \vec{V}_i(t) = [V_{i,1}(t), V_{i,2}(t), \ldots, V_{i,n}(t)] \), \( i = 1, 2, \ldots, m \). The position vector of each particle with respect to the origin of the search space represents a candidate solution of the search problem. Each particle also has a small memory comprising its personal best position experienced so far, denoted by \( \vec{P}_i(t) \) and the global best position found so far, denoted by \( \vec{g}(t) \). Here, one position is considered better than another if the former gives a lower value of the objective function, also called the fitness function in this context, than the latter.

For each particle, each component \( X_{i,j}(0) \) of the initial position vector is selected at random from a predetermined search range \([X_{i,j}^L, X_{i,j}^U] \), while each velocity component is initialized by choosing at random from the interval \([-V_{max}, V_{max}] \), where \( V_{max} \) is the maximum possible velocity of any particle in the \( j \)th dimension, \( j = 1, 2, \ldots, n; i = 1, 2, \ldots, m \); the initial settings for \( \vec{P}_i(t) \) and \( \vec{g}(t) \) are taken as \( \vec{P}_i(0) = \vec{X}_i(0), \vec{g}(0) = \vec{X}_k(0) \) such that \( f(\vec{X}_k(0)) \leq f(\vec{X}_i(0)) \) \( \forall i \).

After the particles are initialized, the iterative optimization process begins, where the positions and velocities of all the particles are updated by the following recursive equations (1) and (2). The equations are presented for the \( j \)th dimension of the position and velocity of the \( i \)th particle.

\[ \begin{align*}
V_{i,j}(t+1) &= \omega V_{i,j}(t) + C_1 \varphi_1 (p_{i,j}(t) - X_{i,j}(t)) + C_2 \varphi_2 (g_j(t) - X_{i,j}(t)) \\
X_{i,j}(t+1) &= X_{i,j}(t) + V_{i,j}(t+1)
\end{align*} \] (2)

where the algorithmic parameters are defined as:
\[ \omega : \text{ inertial weight factor, } \]
\[ C_1, C_2 : \text{two constant multipliers called self confidence and swarm confidence respectively, } \]
\[ \varphi_1, \varphi_2 : \text{two uniformly distributed random numbers.} \]

In our problem, the search range for position vector component in each dimension is the set of channel numbers \([1,2,\ldots,N_{ch}] \), \( V_{max} = (N_{ch} - 1) \) \( \forall j \). We take \( m = 40, \omega = 0.729, C_1 = C_2 = 1.494, 0 < \varphi_1, \varphi_2 \leq 1 \).

The iterations are allowed to go on for a certain predetermined number of time-steps (maxiter), or until the fitness of the best particle at a certain time-step is better than a pre-defined value (tolerance). On termination of the algorithm, most of the parameter vectors are expected to converge to a small region around the required global optimum of the search space. The fittest vector of the final population is taken as a possible solution to the problem. We keep maxiter = 5000 as our stop criterion; no tolerance is set as the minimum value of the cost function is not known beforehand.

V. DESCRIPTION OF BENCHMARK PROBLEMS

For simulation purposes, we have considered three different classes of problem available in the literature:

1. The first class consists of the data set, denoted as EX1, suggested by Sivarajan et al [4] as well as a slightly larger extension of EX1, denoted as EX2 [7]. This class of problems is used for experimental purposes only as its size is unrealistically small.

2. The next class comprises test problems denoted by HEX1-HEX4 [7], based on a 21-cell system. However, in
this paper, we have discarded HEX2 and HEX4 because, in these problems, ACC has been taken into account while, in our model, we have assumed the absence of it.

3. The final set of problems viz. KUNZ1-KUNZ4 [7] was generated from the topographical data of an actual 24 km×21 km area around Helsinki, Finland, as studied by Kunz. However, we have used only KUNZ1 and KUNZ2 in this paper and have neglected the occurrence of adjacent-channel interference due to our simplified model.

Each of these problems is specified in the literature in terms of the number of cells in the network (Nc), the number of channels in the pool (Ncb) and a demand vector D which is a vector whose ith element denotes the traffic demand in cell i, i = 1,2,…, Nc. The descriptive details of each problem are tabulated in Table I.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Nc</th>
<th>Ncb</th>
<th>Demand vector D</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX1</td>
<td>4</td>
<td>11</td>
<td>1,1,1,3</td>
</tr>
<tr>
<td>EX2</td>
<td>5</td>
<td>17</td>
<td>2,2,2,4,3</td>
</tr>
<tr>
<td>HEX1</td>
<td>21</td>
<td>37</td>
<td>2,6,2,2,2,4,13,19,7,4,4,7,4,9,14,7,2,2,4,2</td>
</tr>
<tr>
<td>HEX2</td>
<td>21</td>
<td>21</td>
<td>1,1,1,3,6,7,10,10,11,5,6,4,4,7,5,5,5,6</td>
</tr>
<tr>
<td>KUNZ1</td>
<td>10</td>
<td>30</td>
<td>10,11,9,5,9,4,5,7,4,8</td>
</tr>
<tr>
<td>KUNZ2</td>
<td>15</td>
<td>44</td>
<td>10,11,9,5,9,4,5,7,4,8,8,9,10,7,7</td>
</tr>
</tbody>
</table>

However, in order to make each problem compatible with our model, we made the following modifications:

1. As we have assumed that all cells are arranged in the form of a parallelogram, we express the given Nc of each problem in the form r × c, where r, c are integers, and hence determine the configuration of the cellular network by setting the number of rows to r and the number of all columns to c. We maintain r ≤ c and try to choose r and c in such a way as to keep the rows and columns balanced as far as possible.

2. We arbitrarily select a cell k and assume that, just before a call demand arrives in this cell at time t, Dk calls were already in progress in the ith cell, i = 1,2,…, Nce, i ≠ k, and (Dk−1) calls in the kth cell, where Dj denotes the jth entry of the demand vector D, j = 1,2,…,Nce. Accordingly, we have manually constructed for each problem an Nc×Ncb allocation matrix, avoiding co-channel interference, which describes the status of ongoing calls in each cell before the new call arrival and thus embodies the initial condition. e.g. in EX1, we assume that, when a fresh call arrives at cell k = 4 at time t, 1 call in each of cells 1 to 4 and 2 calls in cell 4 were going on with channels 1, 5, 9 assigned to cells 1, 2, 3 respectively and channels 2, 7 being used by cell 4. Similarly, in EX2, when a new call arrives at cell 4, 2 calls in each of cells 1 to 3 and 3 calls in each of the remaining cells were being serviced, with channels 1, 6 assigned to the calls in cell 1, channels 10, 17 to those in cell 2, channels 4, 14 in cell 3, channels 7, 12, 16 in cell 4, and channels 4 (reused), 9, 14 (reused) in cell 5.

The relevant details are provided in Tables II, III, IV and V:
VI. SIMULATION

We wrote a program in C for minimising the fitness function $F(X)$ using PSO. For each problem, we performed at least 100 simulations under the same initial conditions (i.e. for all the simulations of a particular problem, we assumed that a call demand arrives in the same cell $k$ when the network has the same allocation matrix). Each simulation is identical to taking the average of 10 runs of the PSO algorithm. If, on applying the update formulas, any component of position or velocity exceeds the highest permissible value or becomes smaller than the lowest allowed value, it is fixed at the corresponding highest or lowest value respectively. However, the final result is a vector of real numbers between 1 and $N_{ch}$ but only integral values (channel numbers) are admissible. So, we round off each real-valued component of the solution vector by its “floor” or “ceiling” according as its fractional part is smaller or larger than 0.5. Each time that a simulation results in a solution, which violates CCC, the call is rejected. As a parameter for determining the effectiveness of our method, we define the call rejection probability (CRP) as

$$\text{CRP} = \frac{N_{\text{rejected}}}{N_{\text{total}}}$$

where

$N_{\text{rejected}}$ = number of simulations in which the incoming call is rejected in the host cell considered;

$N_{\text{total}}$ = total number of simulations.

Thus, CRP is the cumulative proportion of simulations, for which the call is rejected, in the long run.

It is worthwhile to mention here that this parameter CRP is based on but different from the call blocking probability used in the [1], [2]; the former characterizes a particular cell under a given initial condition while the latter characterizes the cellular network as a whole.

The simulation results for each of the problems solved are presented in Table VI. Figures 2 and 3 show the convergence of CRP for HEX1 (which has the largest size among the benchmarks considered) and KUNZ2 (which is based on a real-life situation). Figure 4 shows the computed values of CRP in the selected host cells for the different benchmarks.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Host cell $k$</th>
<th>Number of simulations</th>
<th>Number of times the call is rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX1</td>
<td>4</td>
<td>112</td>
<td>32</td>
</tr>
<tr>
<td>EX2</td>
<td>4</td>
<td>100</td>
<td>28</td>
</tr>
<tr>
<td>HEX1</td>
<td>6</td>
<td>104</td>
<td>21</td>
</tr>
<tr>
<td>HEX3</td>
<td>5</td>
<td>105</td>
<td>32</td>
</tr>
<tr>
<td>KUNZ1</td>
<td>7</td>
<td>110</td>
<td>39</td>
</tr>
<tr>
<td>KUNZ2</td>
<td>7</td>
<td>100</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 2. Plot of the cumulative proportion of simulations for which call is rejected in cell 6 vs. the number of simulations in HEX1

Figure 3. Plot of the cumulative proportion of simulations for which call is rejected in cell 7 vs. the number of simulations in KUNZ2
VII. CONCLUSION

In this paper, we have applied PSO, which is known to perform better than many other optimization algorithms (such as Genetic Algorithm), to the CAP and have obtained reasonably good results in terms of the CRP, which is a new parameter introduced by us.

However, a real cellular network is more complex and the calls arrive in different cells in a more random manner, following a certain distribution. We are currently investigating how this algorithm will perform under such dynamically changing traffic load in a more realistic situation.

REFERENCES


