M.Tech.(Computer Science) Dissertation Series

Channel Assignment in Cellular Mobile Networks Using a Heuristic Algorithm

A dissertation submitted in partial fulfilment of the requirements for the M.Tech.(Computer Science) degree of the Indian Statistical Institute

By

Kalikinkar Mandal
Roll No : CS0706

Under the supervision of

Prof. Bhabani P. Sinha

Indian Statistical Institute
203, B.T. Road
Kolkata-700108

Certificate

This is to certify that the Dissertation titled "**Channel Assignment in Cellular Mobile Networks Using a Heuristic Algorithm**", done at Advanced Computing Microelectronics Unit, Indian Statistical Institute, Kolkata, is a bonafide work done by Kalikinkar Mandal, as a partial fulfillment for the award of the degree of Master of Technology in Computer Science under my guidance.

Prof. Bhabani P. Sinha
Date:

Countersigned
(External Examiner)
Date:

Acknowledgement

I am grateful to **Prof. Bhabani P. Sinha**, ACM Unit, Indian Statistical Institute, Kolkata, for providing me the opportunity to work at ACM Unit under his guidance. It was pleasure working under his supervision. He always available with new ideas and suggestions during the difficult phase of the study.

I would like to express my gratitude to all faculty members of ACM Unit and friends for their encouragement and help during the course of this dissertation.

Kalikinkar Mandal(cs0706)
M.Tech(CS), 2nd Year
ISI, Kolkata

Contents

Abstract

Our proposed technique deals with the channel assignment problem in a hexagonal cellular network with two-band buffering, where the channel interference does not extend beyond two cells. Here we introduced notion of simpler sub-network. We present an algorithm for solving the channel assignment problem. The proposed algorithm provides a near-optimal assignment for two benchmark problems and an optimal solution for remaining six benchmark problems but the computation time is less than 50 milisecond for all the problems on HPxw8400 Workstation compared to 10-20 second time taken by the algorithm in Coalesced cap approach (on an unloaded DEC Alpha station 200 4/233) for obtaining optimal solutions.

Index Terms− Simpler sub-network, benchmark problems, cellular networks, channel assignment, bandwidth.

Chapter 1

Introduction

1.1 Introduction

In recent years, the number of mobile users has grown up rapidly, while the communication bandwidth for providing services to them has remained more or less unchanged. Hence, the problem of using the radio spectrum efficiently to satisfy the customer demands has become a critical research issue. The geographical area under the service domain of a mobile cellular network is divided into a number of cells. Structure of each cell is considered as hexagonal in shape. Whenever a mobile cellular network is designed, each cell is assigned a set of frequency channels to provide services to the individual calls of that cell.

The **Channel Assignment Problem** is the task of assigning frequency channel to the cells satisfying some frequency separation constraints to avoid channel interference and using as small bandwidth as possible. we consider here the static model of CAP, where the demands of cells are known a priori.

For a network, the available radio spectrum is divided into non-overlapping frequency bands. We assume that the frequency bands are of equal length and are numbered as 0, 1, 2, ..., from the lower end. Each such frequency band is called as a channel. The terms channel assignment and frequency assignment will be used interchangeably in our discussion. The highest numbered channel required in an assignment problem is termed as *bandwidth*. Three types of interference [1] are generally taken into consideration in the form of constraints:

1. **co-site channel constraint:** any pair of channels assigned to the same cell must be separated by a certain number.

2. **adjacent channel constraint:** adjacent channels are not allowed to be assigned to certain pairs of cells simultaneously.

3. **co-channel constraint:** same channel is not allowed to be assigned to certain pairs of cells simultaneously.

We consider the Channel Assignment Problem in a hexagonal cellular network with 2-band buffering, where the interference does not extend beyond two cells. For various relative values of s_0, s_1, and s_2, the minimum frequency separations required to avoid interference for calls in the same cell, or in cells at distances one and two respectively.

The channel assignment problem is equivalent to a generalized graph-coloring problem, which is a well-known NP-complete problem. An exact search for the optimal solution is impractical for large-scale system due to its exponentially growing computation time. As a result, most of the investigations on this problem are based on heuristic approaches.

In this work we present a channel assignment algorithm for assigning channels to the different nodes to meet the total demand on each node in sucessive phases. In each phase we would first try to do the assignment only on linear chains of nodes such that any two nodes in two different chains are separated by a distance of at least two. However, if it is not possible to find such chains, we would then do the assignment with lines or triangles or quadrilateral superimposed on a chain as shown in Figure 3.1-4. The basic motivation behind this approach of sucessive multiphase assignments is to break the total assignments into assignments on simpler network structures (sparse chains or chains with few lines or chains with triangles or chains with quadrilateral(s)) on them with homogeneous demand on each node (only a portion demand on each node being met in every phase) and such an assignment can be done very quickly. The solution by this approach is near-optimal, requiring at most 5-6% more channels than the optimal solution. For all the philadelphia benchmark problems, but the execution time is significantly reduced. For example our proposed algorithm takes less than 50 milisecond execution time on HPxw8400 Workstation compared to 10-20 sec. time taken by the algorithm given in [2] for obtaining optimal solutions.

1.2 Preliminaries

In this section first we describe the general model of CAP for any arbitrary cellular network. Then we give some definition.

1.2.1 Model of the cellular network:

We consider here general model of CAP which is described in [3]. This model is described by the following components:

1. The number of distinct cells, say, n, with cell numbers as 0,1, ..., n-1.
2. A demand vector $D = (d_i)$ $(0 \leq i \leq n\text{-}1)$ where d_i represents the number of channels required for cell i.
3. A frequency separation matrix $C = (c_{ij})$, where c_{ij} represents the minimum frequency separation requirment between a call in cell i and a call in cell j,$(0 \leq i, j \leq n\text{-}1)$.

4. A frequency assignment matrix $\Phi = (\phi_{ij})$, where ϕ_{ij} represents the frequency assigned to call j in cell i $0 \leq i \leq n$-1, $0 \leq j \leq d_i$-1. The assigned frequencies ϕ_{ij}'s are assumed to be evenly spaced and can be represented integers ≥ 0.

5. A set of frequency separation constraints specified by the frequency separation matrix: $|\phi_{ik} - \phi_{jl}| \geq c_{ij}$ for all i, j, k, l(except when $i = j$, $k = l$).

The goal of the channel assignment problem is to assign frequencies to the cells satisfying the frequency separation constraints as specified above, in such a manner that the system bandwidth becomes optimal.

1.2.2 Definition

Definition 1 The cellular graph is a graph where each cell of the cellular network is represented by a node and two node have an edge between them if the corresponding cells are adjacent to each other.
Definition 2 The cellular network is said to belong to a k-band buffering system if it is assumed that the interference does not extend beyond k cells from the call originating cell.
Definition 3 Suppose $G = (V, E)$ is a cellular graph. A subgraph $G' = (V', E')$ of the graph $G = (V, E)$ is defined to be distance *k-clique*, if every pair of nodes in G' is connected in G be a path of length at most k.

Chapter 2

Related Work

In this chapter we mention some techniques to solve the *channel assignment problem* and give brief descriptions of the methods.
Since CAP is an NP-complete problem, researchers have tried to solve the problem by an approximation algorithms using neural networks, simulated annealing, tabu search [4] etc. Several authors proposed a number of techniques using genetic algorithms [5], introducing the concept of critical block [3], coalesced CAP Approach [2] etc.

2.1 Lower bounds on bandwidth

2.1.1 Homogeneous demand on hexagonal cellular network

Consider a hexagonal cellular network with 2-band buffering system. Let ω be the demand of each node. Since we have considered 2-band buffering system, there are three parameters s_0, s_1, s_2, which are to avoid three types of interferences. Lower bounds on the required bandwidth for the cellular network with homogenious demand ω is defined in terms of s_0, s_1 and s_2 [6].

2.1.2 Non-homogeneous demand on hexagonal cellular network

We assume that the calls in the same cell should be separated by at least s_0 and the calls in the cells those are distance one apart should be separated by at least s_1 and those are distance two apart should be separated by at least s_2.

Bandwidth Bounds:
Given a distance-2 clique G with non-homogeneous demand vector $D = (d_i)$, it is necessary to know the lower bounds on the minimum number of frequency needed to its assignment to check optimality of the solution. Let $d = \max_{1 \leq i \leq 7}(d_i)$. Then, a trivial lower bound on the bandwidth is $s_0(d-1)$ [3]. This lower bound is not always tight for all values of s_0, s_1, s_2

and D. The tight lower bound can be calculated in the following way:
Lemma A lower bound on minimum bandwidth [3] for G with demand vector $D = (d_i)$, where $d = max_{1 \leq i \leq 7}(d_i)$ is:

1. $max((d-1)s_0, (\sum_{1 \leq i \leq 7} d_i - 1)s_2 + (s_0 - s_2)(d_4 - 2) + 2(s_1 - s_2))$ for $s_1 \leq s_0 \leq (2s_1 - s_2)$ and

2. $max((d-1)s_0, (\sum_{1 \leq i \leq 7} d_i - 1)s_2 + 2(s_1 - s_2)(d_4 - 2) + 2(s_1 - s_2))$ for $s_0 \geq (2s_1 - s_2)$.

Let $W = (\omega_i)$ be the demand vector, where ω_i is the channel required for the node i. Let $\omega = \max_{1 \leq i \leq n}(\omega_i)$. Then the trivial lower bound on bandwidth is $(\omega - 1)s_0$ [3]. This lower bound is not always tight for all values of s_0, s_1 and s_2 and W for 2-band buffering system.

2.1.3 Non-hexagonl cellular network

Lower bound for the non-hexagonal cellular network can be found by reconstituting a subset of the network in a hierarchical way [2].

2.2 Channel Assignment Technique

2.2.1 Homogeneous demand

Genetic algorithm approach: Using the elitist model of genetic algorithm(EGA) we can solve Channel Assignment Problem(CAP). For hexagonal cellular networks with homogeneous demand of ω channels per cell, this approach essentially selects a small subset of cells of the network, on which we apply the EGA to find its assignment and next repeat the assignment for the whole network. For $\omega = 1$, this approach improves the bandwidth requirement by 25% at best over that in [5].

Cellular graph is a graph where each cell of the cellular network is represented by a node and two nodes have an edge between them if the corresponding cells are adjacent to each other. cellular graph simply represents the topology of the cellular structure, without any regard to the demand per cell, and is different from the CAP graph mentioned above. The cellular graph is of hexagonal structure with two-band buffering, i.e., the interference extends only up to two cells from the call originating cell.

GA for channel assignment technique:
Let Q be the set of all finite length string or chromosome. Each element of the string is an (rs), where is the cell number at which a call is generated and s is the call number to this cell r. A collection of M(finite) such strings or chromosomes is called a *population*. A simple genetic algorithm is composed of three basic operators: 1) reproduction or selection, 2) crossover,(we have taken here PMX) and 3) mutation. We will repeat the above three

steps up to certain number of times with some crossover probability(cp=0.95) and mutation probability (here we start with 0.5) [5]. Fitness function is bandwidth required for a string. The computation time for problem 3 and 7 are 0.5-1.0 sec. For problem 1 2-5 sec, problem 4 6-12 sec, problem 5 2-7 sec, problem 8 6-17 sec. For problems 2 and 6, computation time for the optimal assignment varied between 12 80 h for different runs on DEC Alpha station.

2.2.2 Non-homogeneous demand

Homogeneous demands can also be used to solve the channel assignment problem with non-homogeneous demand vector $W = (\omega_i)$. Given a network with nonhomogeneous demand vector $W = (\omega_i)$ with ω_i being the demand for cell i. The trivial minimum bandwidth requirement is given by $(\omega - 1)s_0$ [3]. we may obtain a solution to the problem with nonhomogeneous demand vector $W = (\omega_i)$ where $\omega = \max(\omega_i)$ keeping the bandwidth requirement very close to the optimal one.

Critical block approach:
Let there be n nodes in the cellular graph of k-band buffering with a demand vector $W = (\omega_i)$, $1 \leq i \leq n$.
Given a cellular graph G with a demand vector W, and the set of all possible distance k-cliques $\{G_j\}$, each with minimum bandwidth B_j, the critical block is that distance k-cliques, whose minimum bandwidth requirement is maximum for all B_j's.
For a network with a given demand vector and frequency separation constraints, we present an algorithm for finding its critical block. A novel idea of partitioning (through a linear integer programming (IP) formulation) the critical block into several smaller sub-networks with homogeneous demands has been introduced which provides an elegant way of assigning frequencies to the critical block with a very small execution time [3]. This idea of partitioning is then extended for assigning frequencies to the rest of the network. The proposed algorithm gives an optimal solution for all the eight benchmark instances, with minimum number of frequency channels. The running time of all the eight benchmarks except problems 2 and 6 takes a few seconds on an unloaded Sun Ultra 60 workstation. For the problems 2 and 6 algorithm needs around 60 seconds and 72 seconds of running time, respectively on the same workstation.

Coalesced CAP approach:
An elegant technique for solving the channel assignment problem which can be applied even to a cellular network with no regular hexagonal structure. This technique first maps a given problem P to a modified problem P' on a small subset of cells of the network, offering a much reduced search space. This helps solving the problem P' by applying approximate algorithms more efficiently [2]. This solution to P' is then used to derive the solution to the original problem P, based on the solution obtained for P', two possible situations may arise: 1) the solution to P derived from the solution to P' results in zero call blocking, i.e., it is an

admissible solution for P or
2) if all requirements for P are not satisfied by the solution to P', resulting in call blocking. An algorithm which is a modified version of the forced assignment with rearrangement(FAR) operation reported in [11]. Application of this Modified FAR operation to well-known benchmarks always generates optimal results for all of them. The running time of this algorithm is around 10 and 20 seconds on an unloaded DEC Alpha station 200 4/233, for the benchmark problems 2 and 6, respectively to get optimal solution.

Chapter 3

Channel Assignment Algorithm

Introduction

In this chapter, we present our proposed channel assignment algorithm for assigning channels to different nodes to meet the total demand on each node, in sucessive phases. First we break up the total demand of the network (which is non-homogeneous in general) in terms of homogeneous demands on different simpler network structure of the original network. The simpler network structures are sparse chains or perturbed chains having few lines or triangles or quadrilateral(s) superimposed on some nodes of such chains as shown in the Figure 3.1-4. This process of finding this sparse chains or sparse perturbed chains will be done by a node-finding algorithm. After that the actual assignment of frequencies with homogeneous demands taken together on appropriate simpler sub-networks of the given network is done by frequency assignment algorithm. We find the homogeneous demand on each such sub-network by a weight finding algorithm. Finally all these homogeneous assignment of the appropriate sub-networks of the given network together constitute the non-homogeneous assignment of the original netwok.

In critical block approach [3] the assignment is first done over the critical block. But here our approach is to break the whole network into several small sub-networks with almost linear (chain) structure having homogeneous demand of the nodes. Then we do the assignment of this sub-networks sucessively producing a very fast result with non-optimal solutions.

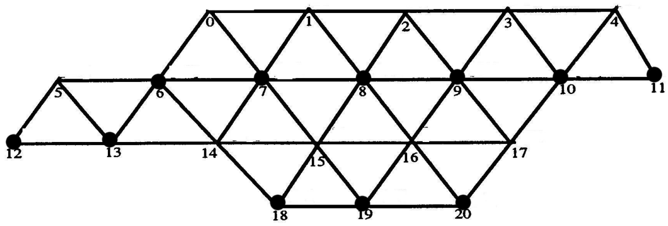


Figure 3.1: Sparse chain

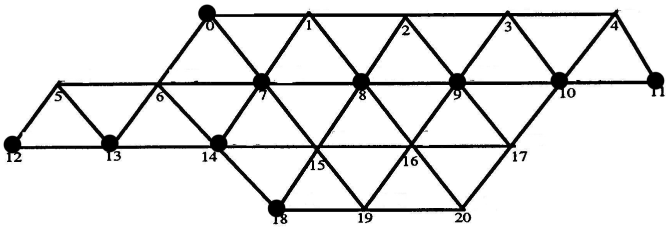


Figure 3.2: Chain with lines

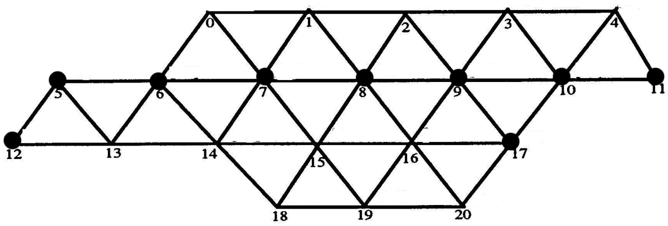


Figure 3.3: Triangle superimposed on a chain

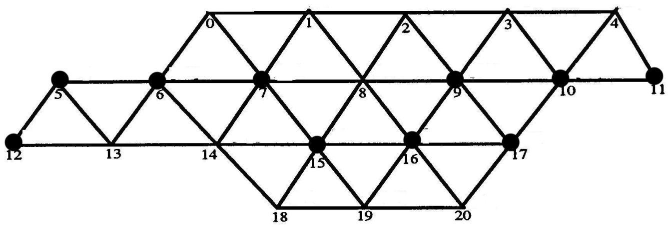
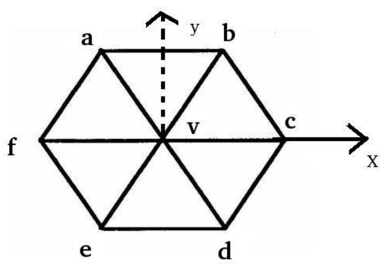


Figure 3.4: Quadrilateral superimposed on a chain

3.1 Construction of Simpler Network

We construct a sub-networks by choosing some nodes. We choose the nodes from the network based on the demand vector and some minimum bandwidth. The construction of the network is done in two stages. In the first stage, we try to find linear chain. If it is not possible to find linear chain then we find a chain with lines or triangle or quadrilateral i.e., linear chain with some perturbation. Then in the second stage, we find another linear chain if possible. Before that we introduce the notion of forward node, backward node, distance matrix and some definitions related to this.

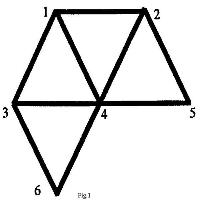
Consider the cellular graph given below. Take the node v. Node v has adjacent nodes a, b, c, d, e and f i.e node v has six neighbours. Let $N(v)$ be the neighbours of the node v, where



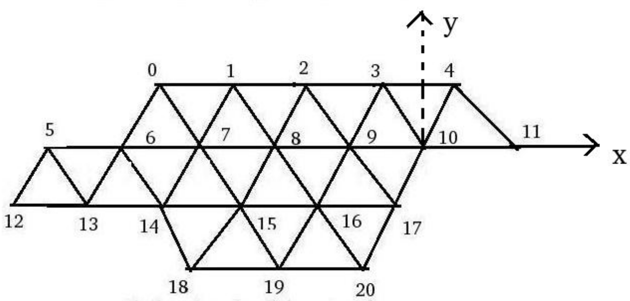
$N(v) = \{a, b, c, d, e, f\}$.
Now we partition the set $N(v)$ into two disjoint subsets as described below.
Let v be the origin. Let vX be the x-axis and vY be the y-axis. The nodes which are belonging to the angle range 90^0-270^0, put those node(s) into the backward neighbor set and the node(s) which belongs to the range 271^0-89^0, put those node(s) into the forward neighbor set.
We denote the forward neighbor of v as $N_f(v)$ and backward neighbor of v as $N_b(v)$.
$N_f(v) = \{b, c, d\}$.
$N_b(v) = \{a, f, e\}$.

3.1.1 Example

Consider the graph given below.



In the above network the forward neighbor and backward neighbor of node 4 are:
$N_f(4) = \{2, 5\}$
$N_b(4) = \{1, 3, 6\}$



The benchmark cellular network

In the above figure suppose there are imaginary axes x-axis and y-axis at each node to calculate forward neighbor and backward neighbor.
In the above network forward and backward neighbor of the nodes 3, 8, 18 are given below:
$N_f(3) = \{4, 10\}$ and $N_b(3) = \{2, 9\}$
$N_f(8) = \{2, 9, 16\}$ and $N_b(8) = \{1, 7, 15\}$
$N_f(18) = \{15, 19\}$ and $N_b(18) = \{14\}$
Similarly one can find the forward and backward neighbor of all the nodes in the network.
Notations:
Let D be the demand vector and let N be the number of nodes in the network.
Let ω =$(\omega_1, \omega_2, ..., \omega_\alpha)$ be the weight vector. We will detremine α later.
We denote N_b[i][j] is the j th backward neighbor node of node i. Similarly N_f[i][j] is the j th forward neighbor node of node i.
We denote $Nodes^f$[ ], the set of nodes is found using f frequency channels.

3.2 Definitions:

Chain: Let $v_1, v_2, ..., v_n$ be the sequence of nodes. The *chain* formed by these nodes are defined as $v_1e_1v_2e_2...e_{n-1}v_n$ where e_i = edge between v_i and v_{i+1}.

In this context the terms *chain* and *path* will be used interchangeably in our discussions.

Distance between two nodes: Let v_1 and v_2 be two nodes in the network. We denote $d(v_1, v_2)$ is the distance between the node v_1 to v_2 and is defined as length of the shortest path from v_1 to v_2.

Triangle: We call a chain contains a *triangle* if there exists a node v and two other nodes v_1 and v_2 which are different from v such that $d(v, v_i) = 1$ for $i = 1, 2$ and $d(v_1, v_2) = 1$. Where $d(v, u)$ is the distance between the node u and v.

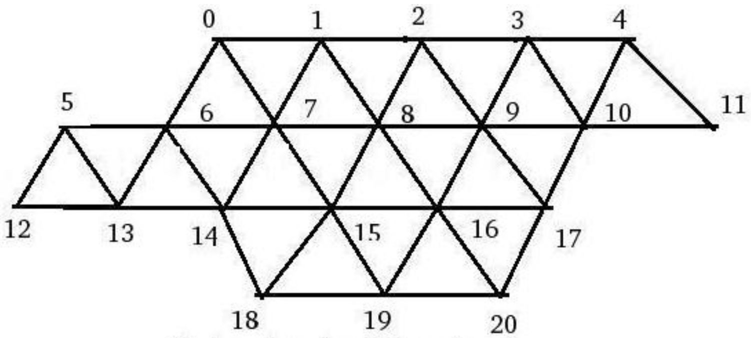
Weight Vector: Let $V = \{v_1, v_2, ..., v_n\}$ be the set of nodes. Let $A = \{a_1, a_2, ..., a_n\}$ be the assignment on the nodes. We denote $\omega = (\omega_i)$ as a weight vector and ω_i is how many times we take the same set V of nodes and the assignment A.

Sparse chain: Let C_1 be a chain formed by the nodes $v_1, v_2, ..., v_n$ and C_2 be the chain formed by the nodes $u_1, u_2, ..., u_m$. We call C_1 and C_2 are sparse chain if for any node v_i from C_1 and for any node u_j from C_2, $d(v_i, u_j) \geq 2$ for $1 \leq i \leq n, 1 \leq j \leq m$. The distance between the chains C_1 and C_2 are at least two.

Quadrilateral: We call a chain contains a *quadrilateral* if there exists a node v and three other nodes v_1 , v_2 and v_3 which are different from v such that $d(v, v_i) = 1$ for $i = 1, 2, 3$ and $d(v_1, v_3) = 2$. Where $d(v, u)$ is the distance between the node u and v.

3.3 Distance Matrix

Distance matrix stores the information redarding the distance between two any nodes in the network. We require the distance between two nodes, when we assign the frequency channel to the sub-networks.
Consider the 21-node cellular network.



The benchmark cellular network

Let D_{matrix} be the distance matrix of the above network of order 21×21. For the node i and node j, (i, j)-th entry of D_{matrix} is distance between the node i and node j.
The distance matrix corresponding to the above network is defined as:

$$D_{matrix} = \begin{pmatrix}
0 & 1 & 2 & 3 & 4 & 2 & 1 & 1 & 2 & 3 & 4 & 5 & 3 & 2 & 2 & 2 & 3 & 4 & 3 & 3 & 4 \\
1 & 0 & 1 & 2 & 3 & 3 & 2 & 1 & 1 & 2 & 3 & 4 & 4 & 3 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\
2 & 1 & 0 & 1 & 2 & 4 & 3 & 2 & 1 & 1 & 2 & 3 & 5 & 4 & 3 & 2 & 2 & 2 & 3 & 3 & 3 \\
3 & 2 & 1 & 0 & 1 & 5 & 4 & 3 & 2 & 1 & 1 & 2 & 6 & 5 & 4 & 3 & 2 & 2 & 4 & 3 & 3 \\
4 & 3 & 2 & 1 & 0 & 6 & 5 & 4 & 3 & 2 & 1 & 1 & 7 & 6 & 5 & 4 & 3 & 2 & 5 & 4 & 3 \\
2 & 3 & 4 & 5 & 6 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 1 & 1 & 2 & 3 & 4 & 5 & 3 & 4 & 5 \\
1 & 2 & 3 & 4 & 5 & 1 & 0 & 1 & 2 & 3 & 4 & 5 & 2 & 1 & 1 & 2 & 3 & 4 & 2 & 3 & 4 \\
1 & 1 & 2 & 3 & 4 & 2 & 1 & 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 1 & 2 & 3 & 2 & 2 & 3 \\
2 & 1 & 1 & 2 & 3 & 3 & 2 & 1 & 0 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 1 & 2 & 2 & 2 & 2 \\
3 & 2 & 1 & 1 & 2 & 4 & 3 & 2 & 1 & 0 & 1 & 2 & 5 & 4 & 3 & 2 & 1 & 1 & 3 & 2 & 2 \\
4 & 3 & 2 & 1 & 1 & 5 & 4 & 3 & 2 & 1 & 0 & 1 & 6 & 5 & 4 & 3 & 2 & 1 & 4 & 3 & 2 \\
5 & 4 & 3 & 2 & 1 & 6 & 5 & 4 & 3 & 2 & 1 & 0 & 7 & 6 & 5 & 4 & 3 & 2 & 5 & 4 & 3 \\
3 & 4 & 5 & 6 & 7 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 & 4 & 5 & 3 & 4 & 5 \\
2 & 3 & 4 & 5 & 6 & 1 & 1 & 2 & 3 & 4 & 5 & 6 & 1 & 0 & 1 & 2 & 3 & 4 & 2 & 3 & 4 \\
2 & 2 & 3 & 4 & 5 & 2 & 1 & 1 & 2 & 3 & 4 & 5 & 2 & 1 & 0 & 1 & 2 & 3 & 1 & 2 & 3 \\
2 & 2 & 2 & 3 & 4 & 3 & 2 & 1 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 & 1 & 2 & 1 & 1 & 2 \\
3 & 2 & 2 & 2 & 3 & 4 & 3 & 2 & 1 & 1 & 2 & 3 & 4 & 3 & 2 & 1 & 0 & 1 & 2 & 1 & 1 \\
4 & 3 & 2 & 2 & 2 & 5 & 4 & 3 & 2 & 1 & 1 & 2 & 5 & 4 & 3 & 2 & 1 & 0 & 3 & 2 & 1 \\
3 & 3 & 3 & 4 & 5 & 3 & 2 & 2 & 2 & 3 & 4 & 5 & 3 & 2 & 1 & 1 & 2 & 3 & 0 & 1 & 2 \\
3 & 3 & 3 & 3 & 4 & 4 & 3 & 2 & 2 & 2 & 3 & 4 & 4 & 3 & 2 & 1 & 1 & 2 & 1 & 0 & 1 \\
4 & 3 & 3 & 3 & 3 & 5 & 4 & 3 & 2 & 2 & 2 & 3 & 5 & 4 & 3 & 2 & 1 & 1 & 2 & 1 & 0
\end{pmatrix}.$$

3.4 Assignment Scheme

We have shown frequency assignment scheme for a sequence of nodes. The chain formed by these sequence of nodes does not contains a triangle.

Scheme 1: Suppose we have a sequence of nodes $v_1, v_2, ..., v_{n-1}, v_n$ form a linear chain. Given s_0 channels. Can we assign frequency channel to these nodes.
If s_0 is odd with $s_0 \geq 5$ then $s_0 = 2k + 1$, $k \geq 2$. $s_1 = 2$ and $s_2 = 1$

node#	v_0	v_1	v_2	...	v_{i-1}	v_i	v_{i+1}	...	v_{2k-1}	v_{2k}	...
channel#	0	$k+1$	1	...	$k + \lceil \frac{i-1}{2} \rceil$	$\lfloor \frac{i}{2} \rfloor$	$k + \lceil \frac{i+1}{2} \rceil$	...	k	0	...

This is the frequency assignment of the nodes. For any three node v_{i-1}, v_i and v_{i+1}, the channel difference between v_i and v_{i+1} is $k + \lceil \frac{i+1}{2} \rceil - \lfloor \frac{i}{2} \rfloor$ which is equal to k+1 when i is even and channel difference between v_i and v_{i-1} is $k + \lceil \frac{i-1}{2} \rceil - \lfloor \frac{i}{2} \rfloor$ which is equal to k when i is even. When i is odd then first difference is k and second difference is $k + 1$. So there is no channel interference.

If s_0 is even then $s_0 = 2k$

node#	v_0	v_1	v_2	...	v_{2i-1}	v_{2i}	v_{2i+1}	...	v_{2k-2}	v_{2k-1}	v_{2k}	...
channel#	0	k	1	...	$2k-i+1$	i	$2k-i$	...	$k-1$	$k+1$	0	...

Similarly for any three nodes v_{2i-1}, v_{2i} and v_{2i+1}, the channel difference between the node v_{2i-1}, v_{2i} is $2k-2i+1$ and channel difference between the node v_{2i+1}, v_{2i} is $2k-2i$. Maximum value of i is $k-1$, so the minimum difference is equal to $2k-2(k-1) = 2 = s_1$. Hence there is no interference.

For example take $s_0 = 5$, $s_1 = 2$ and $s_2 = 1$. In Fig.1.1, we see using s_0 frequency channel, we can assign frequency to the nodes on a linear chain of length at least s_0.

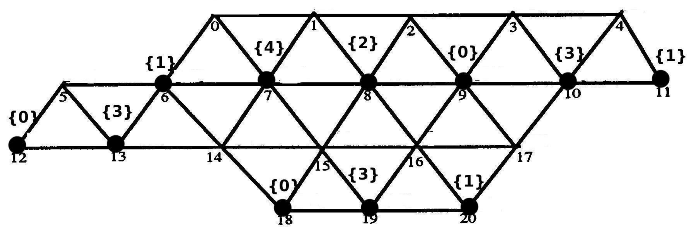


Fig.1.1

In Fig.1.1 each node has a label of the form $\{x\}$ where frequency channel x assigned to that node.

Take $s_0 = 7$, $s_1 = 2$ and $s_2 = 1$.

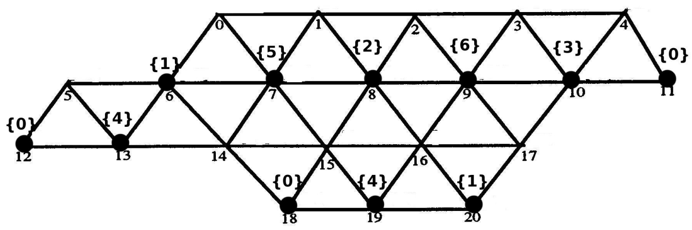


Fig.1.2

In the above Fig.1.2 we have assigned frequency to the nodes on a linear chain of length at least s_0.
So using s_0 frequency channel we can find an frequency assignment of the nodes on a linear

chain of length at least s_0. This is not true for all values of s_0, s_1, s_2.

Cosider a chain with triangle. Take $s_0 = 5$, $s_1 = 2$ and $s_2 = 1$.

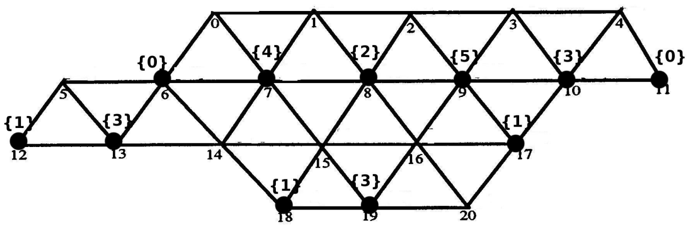


Fig.1.3

In Fig.1.3, each node has a label of the form $\{x\}$ where frequency channel x assigned to that node. Here number of frequency channel required to assign frequencies on the nodes is equal to $s_0 + 1$.
So when a chain contains a triangle then we require $s_0 + 1$ frequency channel, this is not true for all values of s_0, s_1, s_2.

Cosider a chain with triangle. Take $s_0 = 5$, $s_1 = 2$ and $s_2 = 1$.

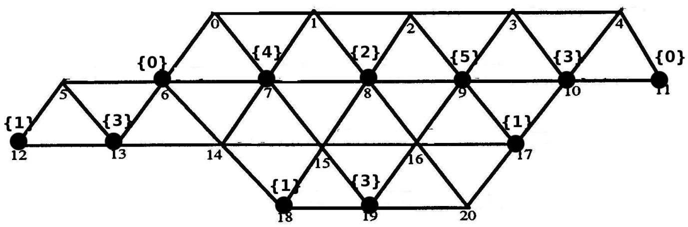


Fig.1.3

A chain having a quadrilateral superimposed on it shown in Fig.1.3. The assignment on the nodes is shown in Fig.1.3. Number of frequency channel required to assign frequencies on the nodes is equal to $s_0 + 2$.

Now the idea is if we find such a chain in the network then the above kind of assignment is possible. Our aim is we try to find such a linear chain and do the assignment. If it is not possible to find such chain then we find chain with triangle or chain with two consecutive triangle.

3.5 Algorithms

In this section we present five type of algorithms. The algorithms are given in the following manner:
1) find a set of node using frequency channel f
2) assign frequency channel to these nodes
3) if frequency reuse is possible in the network then find the nodes and its assignments
4) find the weight at the i-th step
4) iteration for f
5) modify the demand vector according to the set of nodes and its weights.

3.5.1 Node Finding Algorithms

In this section we select a set of nodes to construct the simpler network. We take an amount of bandwidth, say f and choose the nodes by the below algorithm. Our nodes selection method is like that:
1) while we are taking bandwidth s_0, we find a set of nodes such that the path or chain formed by these nodes of length at least s_0, the path does not contain any triangle.
2) while we are using bandwidth $s_0 + 1$, we select a set of nodes such that the chain formed by these nodes of length at least s_0, the path does not contain two consecutive triangle or a quadrilateral.

We are starting to find nodes from maximum demand node. Note that if we have more than one maximum then we take that one whose sum of the neighboring node demand is maximum. Each time we find nodes from maximum demand node.
Algorithm : $Find\_Nodes\_Using\_f\_Frequency$
Input: Max_Demand_Node , maximum demand node and number of frequency channel f.
Output: a sequence of nodes and n, number of nodes.

begin
Step 1: if ($N_f(Max\_Demand\_Node) == null$)

call *procedure* : $Backward\_Search($ Max_Demand_Node , f $)$
Let n_1 be the number of nodes and $Nodes_b[\ ]$ be the sequence of nodes returned by the procedure $Backward\_Search$.

Step 2: if ($N_b(Max\_Demand\_Node) == null$)

call *procedure* : $Forward\_Search($ Max_Demand_Node , f $)$
Let n_2 be the number of nodes and $Nodes_f[\ ]$ be the sequence of nodes returned by the procedure $Forward\_Search$.

Step 3: if ($N_b(Max\_Demand\_Node) \neq null$ & $N_b(Max\_Demand\_Node) \neq null$)

3.1 let max_f and $smax_f$ be the maximum and second maximum of the forward neighbor of Max_Demand_Node , max_b and $smax_b$ be the maximum and second maximum of the backward neighbor of Max_Demand_Node.

3.2 if ($max_b \geq max_f$)

3.2.1 call *procedure* : *Backward_Search*(Max_Demand_Node , f)

3.2.2 if the node corresponding to the value max_f(let the node be N_{max}) and first two nodes selected by the procedure *Backward_Search* form a triangle then take the node corresponding to the value max_f. Call the procedure $Forward\_Search$(N_{max} , f)

3.3 else

3.3.1 call procedure $Forward\_Search$(Max_Demand_Node , f)

3.3.2 choose the backward neighbor node in such a way such that the first two nodes selected by $Forward\_Search$ and this node does not form a triangle while we are using frequency $f = s_0$. Call the procedure *Backward_Search*.

3.4 Let n_1 be the number of nodes and $Nodes_b[\ ]$ be the sequence of nodes returned by the procedure *Backward_Search*. Let n_2 be the number of nodes and $Nodes_f[\ ]$ be the sequence of nodes returned by the procedure $Forward\_Search$. $n = n_1 + n_2$, total number of nodes, and $Nodes_b[\ ] \cup Nodes_f\ [\ ]$ are the sequence of nodes.

end

Remark: If $f = s_0 + 1$, then in the neighborhood of maximum demand node we choose the nodes in such a way that the chain formed by the nodes does not contain two consecutive triangles or a quadrilateral. Again if $f = s_0 + 2$ then we don't need to follow this strategy, we just simply take the maximum demand node.

Algorithm : *Backward_Search*

Input: f , number of frequency and Max_Demand_Node maximum demand node.

Output: $Nodes_b[\ ]$ contains sequence of nodes and n number of nodes in the set $Nodes_b[\ ]$.

begin

Step 1: Set $i = 0$ and $Nodes_b[i] = Max\_Demand\_Node.$

Step 2: if ($f = s_0$)

while($N_b[\ Nodes_b[i]\ ][1] \neq null$)

{

2.1 Find the first maximum and second maximum demand nodes of backward neighbor of $Nodes_b[i]$.

2.2 $i = i + 1$.

2.3 Put the maximum demand node into the array $Nodes_b[\,]$, provided the chain formed by the nodes does not contains a triangle. Otherwise put second maximum demand node. If any node has three neighbor and first maximum and second maximum are iequal and maximum demand of both backward neighbor of maximum and second maximum demand node are same then take third maximum demand node of $Nodes_b$[i-1].

}

Step 3: if ($f = s_0 + 1$)
while($N_b[\ Nodes_b[i]\ ][1] \neq null$)
{

3.1 Find the first maximum and second maximum demand nodes of backward neighbor of $Nodes_b[i]$.

3.2 $i = i + 1$.

3.3 Put the maximum demand node into the array $Nodes_b[\,]$, provided the path formed by the nodes does not contains a quadrilateral or two consecutive triangle. Otherwise put the second maximum demand node.

}

Step 4: if ($f > s_0 + 1$)
while ($N_b[\ Nodes_b[i]\ ][1] \neq null$)
{

4.1 Find the first maximum and second maximum demand nodes of backward neighbor of $Nodes_b[i]$.

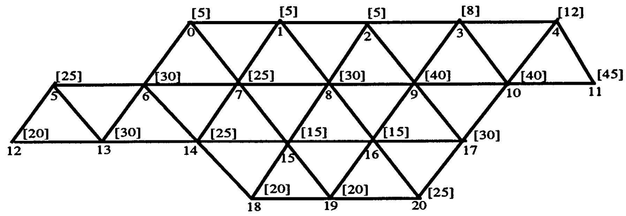
4.2 $i = i + 1$.

4.3 Put the maximum demand node into the array $Nodes_b[\,]$.

}

Step 5: return the set of nodes $Nodes_b[\,]$ and $(i + 1)$ is the number of nodes.
end

Algorithm : *Forward_Search*
This algorithm is same as *Backward_search*. Instead of backward neighbor $N_b(v)$ of v we take here forward neighbor of $N_f(v)$ of v.



The benchmark cellular network.

Example 1: In the above network, each node has a label of the form $[x]$, where x is the demand of that node. Node 11 is the maximum demand node and $N_f(11) = null$ so we call the procedure *Backward_Search*. Nodes selected by *Find_Nodes_Using_f_Frequency* are {11, 10, 9, 8, 7, 6, 13, 12}.

3.5.2 Assignment Algorithm

Once we find the set of nodes, the following algorithm is used to assign the channels to the nodes.
Algorithm: *Assignment_On_The_Nodes*
Input: Set of nodes $Nodes^f[\ ]$ and n is the number of nodes and f minimum bandwidth. Frequency separation constraints s_0, s_1, s_2.
Output: A conflict free assignment of the set of nodes. *assignment*[i] is the frequency assignment on the i th node.
begin
Step 1:
$if\ (\ f = s_0$ and $s_0 = 2k + 1)$
{

for $i = 0$ to $n - 1$

$assignment$[i] $= \lfloor \frac{i}{2} \rfloor$, if i is even and s_0 is odd

$assignment$[i] $= \lfloor \frac{s_0}{2} \rfloor + \lceil \frac{i}{2} \rceil$, if i is odd

}

$if\ (\ f = s_0$ and $s_0 = 2k\ )$
{

for $i = 0$ to $\frac{n}{2}$

$assignment$[2i] $= i$

$assignment$[2i+1] $= 2k - i$

}

Step 2:
else
{

2.1 Take first three nodes from the array $Nodes^f$[ ] , check do they form triangle or not.

2.2 Set $i = 0$ and $assignment$[i] $= 0$. Find an appropriate frequency assignment of the first three nodes. Frequency assignment should be admissible.

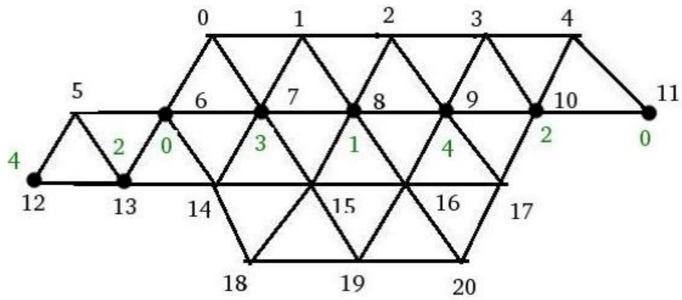
2.3 For $i \geq 3$ follow the following assignment rule:

2.3.1 Take the node $Nodes^f$[i], take the frequency j, find all nodes where the frequency j is used. If all these nodes are 3 distance($k + 1$ distance for k-band buffering system , here $k = 2$) apart from node $Nodes^f$[i] then goto the next step. If all nodes are not three distance apart then then reject the frequency j, try with another one ($0 \leq j \leq f - 1$).

2.3.2 Find all the neighboring node(s) of $Nodes^f$[i], where the frequency is assigned. Let p be the number of nodes and $freq_l$ ($0 \leq l \leq p - 1$) is assigned on the l th node. If | j - $freq_l$ | $\geq s_1$ and | j - $freq_l$ | % ($f - 1$)$\neq 0$ for all l then goto the next step. Otherwise try with another frequency.

2.3.3 Find all node(s) which are distance two apart from the node $Nodes^f$[i] where the frequency is assigned. Let p_1 be the number of nodes and $freq_{l_1}$ ($0 \leq l_1 \leq p_1 - 1$) is assigned on the l_1 th node. If | j - $freq_{l_1}$ | $\geq s_2$ for all l_1, then assign the frequency j on node $Nodes^f$[i]. Otherwise try with another frequency. If assignment of a node is done then assign the frequency to the next node.

}
end
Remark: In the above algorithm in step 2.2 while we are using frequency $f = s_0$ or more, different types of frequency assignment is possible. We take such an assignment that we can assign frequency on the maximum number of nodes and nodes selected by the procedure $Find\_Nodes\_Using\_f\_Frequency$ in the network. When $s_1 = 2s_2$, we don't take step 2.3.3.



Example 2: In Example 1 the nodes selected by $Find\_Nodes\_Using\_f\_Frequency$ are {11, 10, 9, 8, 7, 6, 13, 12}. One possible confilct free assignment is shown in the above figure. Here we have used $f = s_0$ frequency channels. 0-th frequency channel is assigned to the node 11, 2-nd frequency channel is assigned to the node 10, 4-th frequency channel is assigned to the node 9, 1-st frequency channel is assigned to the node 8, 3-rd frequency channel is assigned to the node 7. Then at the node 6, 13 and 12 we are reusing 0-th, 2-nd and 4-th frequency channels respectively, since we have considered the network is a 2-band buffering system. {0, 3, 1, 4, 2, 0, 3, 1} this is also one possible confilct free assignment.

3.5.3 Frequency Reuse Algorithm

When the frequency assignment on the set of nodes return by the algorithm $Find\_Nodes\_Using\_f\_Frequ$ is done, next we find a set of nodes(if possible) in the network where we can reuse the frequency channels and find its assignments. It is not always possible that we can find a set of nodes(where frequency reuse is possible), this selection of nodes depends on the previous selection of nodes.

Algorithm: $Frequency\_Reuse$

Input: The nodes $Nodes^f[\,]$ return by the algorithm $Forward\_Search$ or $Backward\_Search$ and the frequency assignment$assignment[\,]$ on that nodes returned the algorithm $Assignment\_On\_the\_N$ and n number of nodes. Frequency separation constraints s_0, s_1, s_2.

Output: Node(s) $reuseNodes^f[\,]$ and their conflict free frequency assignment.

begin

Step 1: Take the middle most node from the set $Node^f[\,]$ and v be that node. Find all nodes which are d distance apart from the node v. When $f = s_0$, we take $d = 2$ and when $f > s_0$ we take $d = 1$. Let A_d be the set of nodes which are d distance apart from the node v.

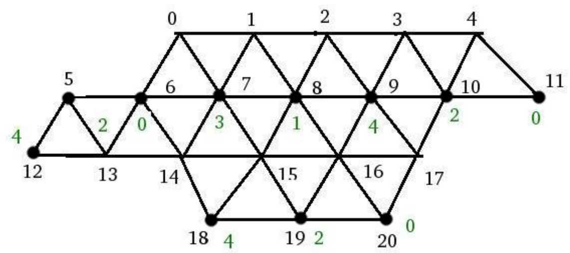
Step 2: For each node v_1 in the set A_d do the following:

Take the frequency j and check can we assign the frequency j on the node v_1 ($0 \leq j \leq f - 1$). If the frequency j is not possible to assign on the node v_1 then try with another frequency.

If frequency assignment is not possible on v_1 then goto the next node in A_d.

end

Remark: In the above algorithm in step 1 we took $d = 2$ because if we take the nodes which are one distance apart from the node v then they may form a triangle there. But we know that while we are using frequency $f = s_0$ then the path formed by the nodes does not contain any triangle. Thus to reduce the number of checking (the checking is whether j^{th} frequency assignment on i^{th} node is valid or not), we have taken $d = 2$.



Example 3: In Example 2 we have seen, {0, 2, 4, 1, 3, 0, 2, 4} is the frequency assignment on the nodes {11, 10, 9, 8, 7, 6, 13, 12}. Now using *Frequency_Reuse* algorithm we found the nodes {18, 19, 20} where we can reuse the frequencies if possible. And {4, 2, 0} are the frequency assignments. {18, 19, 20} these nodes form a chain. The first chain formed by the nodes {11, 10, 9, 8, 7, 6, 13, 12}. The distance between first and second chain is two. This chains are called sparse chains.

At the end of *Frequency_Reuse* algorithm we have constructed the simpler sub-network with its node assignments. In the above network the sub-network is formed by {11, 10, 9, 8, 7, 6, 13, 12, 4, 2, 0} nodes.

Analysis: For some demand vector D, {11, 10, 17, 9, 8, 15, 14, 13, 12} be the set of nodes returned by the algorithm $Find\_Nodes\_Using\_f\_Frequency$ when $f = s_0$. If we assign 0-th frequency channel at node 11, 2-nd frequency channel at node 10 and 4-th frequency channel at node 17 then we can not assign any frequency channel to the node 9. Rest of the frequency assignment on the nodes are shown in Fig.1. Hence we do not prefer this type of assignment.

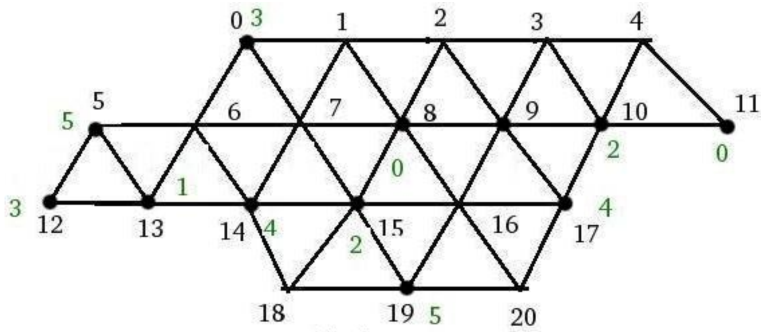


Fig.1

If we assign 0-th frequency channel at node 11, 3-rd frequency channel at node 10 and 1-st frequency channel at node 17 then {0, 3, 1, 5, 2, 0, 3, 1, 4} is the frequency assignment on the linear chain. Now applying algorithm *Frequency_Reuse*, we can assign the frequency channel 4 to the nodes 0 and 19. So we take such an assignment that we can assign frequency channel to the maximum number of nodes.

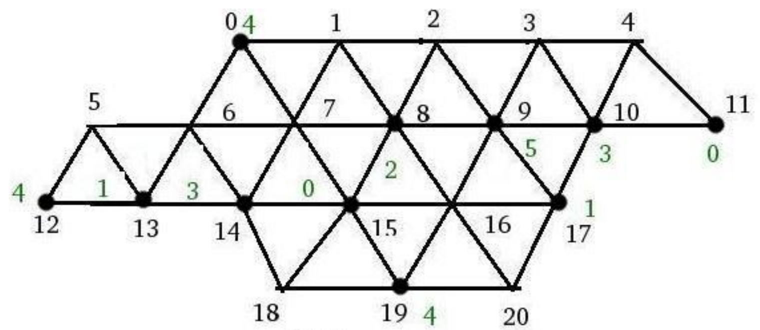


Fig.2

3.5.4 Weight Finding Algorithm

Once we have found the set of nodes and its assignments, we should find the weight corresponding to the set of nodes i.e. how many times we take the same set of nodes and its assignments. Using the below algorithm we find the weight(ω_i) at i^{th} step.

Algoithm : *Find_Weight_At_ith_Stage*
Input: $Nodes[\,]$, the set of nodes returned by the algorithm *Find_Nodes_Using_f_Frequency* and n is number of nodes. *Demand vector* at i th step.
Output: $weight(\omega_i)$, is the weight at the i th step.

begin
Step 1:

$for(\ i = 0\ ;\ i < n-1\ ;\ i++\ )$
{

1.1 Let d_1 and d_2 are the first maximum and second maximum demand respectively of the neighbors of $Nodes[i]$.
If any node has only one neighbor then put $d_2 = 0$.

1.2 Calculate $d = d_1 - d_2$, and $W[\ i\ ]= d$.

}

Step 2: Find the minimum number in the array $W[\ ]$ other than zero and let it be N_{min}.

Step 3: return (N_{min}+1) is the weight at the i th step.
end

Example 4: In Example 1 the nodes selected by $Find\_Nodes\_Using\_f\_Frequency$ are {11, 10, 9, 8, 7, 6, 13, 12}. For each node in the set we find *maximum* and *second maximum* of the *backward* neighbor of that node and calculate their absolute difference. The absolute difference set is {28, 10, 15, 10, 5, 5, 20}. Now 5 is the minimum number in this set. So the $weight(\omega_i{}^{(s_0)}) = 5 + 1 = 6$. *weight* is weight corresponding to the set of nodes.

3.5.5 Iteration finding Algorithms:

Upto this stage we have done the followings:

1. found a set of nodes using f frequency channel.
2. assiged the frequencies to the nodes.
3. checked the further frequency reuse is possible or not.
4. found wight i.e., once we have found a set of nodes and their assignments then how many times we can repeat the assignment on the same set of nodes.

Let V be the set of nodes of the simple sub-network. Let v_1 and v_2 be two nodes where the maximum and minimum frequency channel is assigned. If the demand is more than one then the channel assignment in the second round and onwards, starting from node v_1 again and following the same order as it was in first round.
In the first step what could be the minimum value of f. Minimum value of $f = s_0$ for avoiding co-site channel interferences.

One trivial question is that how many times we can choose $f = s_0$. Similarly how many times we can choose $f = s_0 + 1$ and so on.
Using the algorithm $Iteration\_For\_s_0$ we find how any times we take $f = s_0$.

Must Node: Must node is a node in the network, while we are constructing a linear chain in the node-finding algorithm, the first node chosen by the node-finding algorithm the chain contains a triangle.

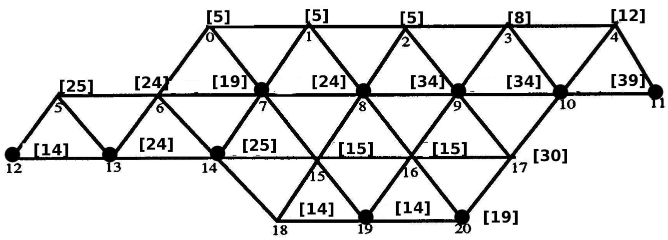
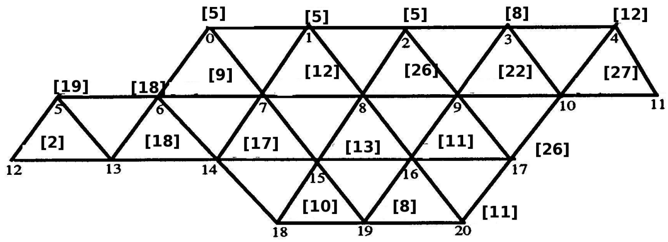


Fig.2

In the above figure node 11 is the maximum demand node. We have choosen the paths from node 11. At node 17 first form the triangle. If a triangle is in the chain of length s_0 then minimum bandwidth s_0+1. We take $f = s_0$ until demand of node 17 and node 11 are equal. If we do not take these two nodes together then one node demand will be more that another. So we need one more iteration to assign the frequency to maximum demand node for that we have to use minimum s_0 frequency channel. When demand of node 11 and 17 are equal then we have to use frequency channel $f = s_0 + 1$. And each step we have to include node 17.

Algorithm: $Iteration\_For\_s_0$
Input: Demand vector D, must node $must\_node$ and maximum demand node Max_Demand_Node.
Output: α_1, number of times $f = s_0$.
begin
Step 1: Find the maximum demand node in the neighbor of $must\_node$. If Max_Demand_Node belonging to the neighbor of must node, do not take that node. Let n_{max} be the maximum demand node and d_{max} be its demand.
Step 2: Calculate difference between must node demand and d_{max} i.e $diff = d_{max}$ - $D[$ $must\_node$ $]$. Again calculate difference between maximum demand (corresponding to maximum demand node) and d_{max} i.e $k = D[$ Max_Demand_Node $]$ - d_{max}.
Step 3: Calculate $T = \frac{diff}{2} + int(\frac{k}{2}) \times 2$, where $int(x)$ is the integer part of x.
Step 4: Return T.
end



After α_1 iteration the structure of the demand vector is shown in the above figure. At this stage we are using $f = s_0 + 1$. When we are choosing the nodes, if node 4 is include then node 11, 4, 17, 9 formed two consecutive triangle. Then we have to take $f = s_0 + 2$. Then by below algorithm we find how many times we take $f = s_0 + 1$.

Algorithm: $Iteration\_For\_(f_2)$

Input: Demand vector D' after α_1 iteration and must node $must\_node$.

Output: α_2, how many times $f = s_0 + 1$.

begin

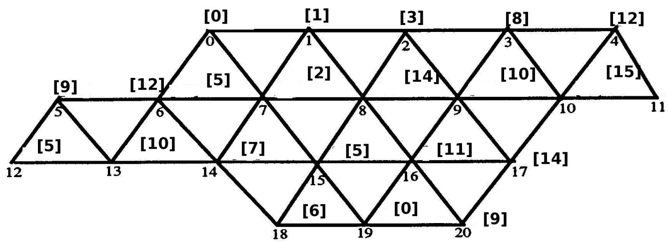
Step 1: Find first maximum and second maximum demand node in the neighbor of *must node* such that these three nodes forms a triangle. Let these three nodes be v_1, v_2, v_3.

Step 2: Let w_1, w_2,...w_k be the neighboring nodes of the triangle. Let $d_{max} = \max_{1 \leq i \leq k} D'[w_i]$.

Step 3: Let $d'_{max} = \min_{1 \leq i \leq 3} D'[v_i]$.

Step 4: Return $\text{int}(\frac{d'_{max} - d_{max}}{2}) + 1$.

end



After α_1+α_2 iteration the structure of the network is shown in the above figure. We take the frequency $f = s_0 + 1$ until the demand of maximum demand node becomes zero.

Algorithm: *Iteration_For_(f_3)*
Input: Demand vector D'' after α_2 iteration and must node *must_node*.
Output: α_3, how many times $f = s_0 + 2$.
begin
Step 1: Return int($\frac{D''[must\_node]}{2}$).
end

According to our initial strategy we will find a set of nodes using frequency f and frequency assignment on the nodes. Once we find the set of nodes and its frequency assignments then we can find how many times we repeat this assignments on the nodes. By weight finding algorithm, we can find the weight corresponding to the particular set of nodes. We have solved the problem how many times we use frequency $f = f_j (1 \leq j \leq 3)$, $f_j = s_0 + j - 1$. By the algorithm *Iteration_For_(f_j)*, we can find number of times we use frequency f.
Let α_j be the number of times we use frequency $f = f_j$. Let ${n_i}^{f_j}$ be the number nodes selected at i-th step and $V_i^{(f_j)}$ be the node set.
$V_i^{(f_j)} = \{V_i^{(f_j)}(k_1), V_i^{(f_j)}(k_2), ..., V_i^{(f_j)}(k_{{n_i}^{f_j}})\}$.
For a particular i and j, ${\omega_i}^{f_j}$ be the weight at the i-th step corresponding to the node set $V_i^{(f_j)}$.

Algorithm: *Preprocessing*
Input: Demand vector D, distance matrix D_{matrix} and forward and backward neighbor of each node. Frequency separation constraints s_0, s_1, s_2.
Output: A set of nodes $V_i^{(f_j)}$ and their frequency assignments ${A_i}^{(f_j)}$, $1 \leq i \leq \alpha_j$ for $1 \leq j \leq 3$. $V_i^{(f_j)}(k_{f_j})$ is the k_{f_j} th node selected at i-th step while we are using frquency $f = f_j$ and ${A_i}^{(f_j)}(k_{f_j})$ is the frequency assignment on that node ($1 \leq k_{f_j} \leq {n_i}^{(f_j)}$).
begin

Step 1: if ($s_1 = s_2$)
{

1. *while* ($D[Max\_Demand\_Node] \neq 0$) {

1.1 Set $f = s_0$.

1.2 call the procedure *Find_Nodes_Using_f_Frequency*.

1.2.1 let $V_i^{(s_0)}$ be the set of nodes and ${n_i}^f$ be the number of nodes.

1.2.2 call the procedure *Assignment_On_the_Nodes* and ${A_i}^{(f)}$ be the frequency assignment on the nodes.

1.2.3 call the procedure *Frequency_Reuse*, let ${R_i}^{(f)}$ be the set of nodes, ${RA_i}^{(f)}$ be the corresponding assignment of the nodes and ${m_i}^{(f)}$ be the number of nodes.

1.2.4 call the procedure *Find_Weight_At_ith_Stage* to find weight. Let ${\omega_i}^f$ be the weight.

1.2.5 decrease the demands by 1 ${\omega_i}^f$ times, for all nodes in the set $V_i^{(f)} \cup {R_i}^{(f)}$.

}

}

Step 2: if ($s_2 \leq s_1 \leq 2s_2$ or $s_1 \geq 2s_2$) {

1. call the procedure *Find_Nodes_Using_f_Frequency* with $f = s_0$.

   1.1 let ${V_1}^{(s_0)}$ be the set of nodes and ${n_1}^{s_0}$ be the number of nodes.

   1.2 call the procedure *Assignment_On_the_Nodes* and ${A_1}^{(s_0)}$ be the frequency assignment on the nodes.

   1.3 call the procedure *Frequency_Reuse*, let ${R_1}^{(s_0)}$ be the set of nodes, ${RA_1}^{(s_0)}$ be the corresponding assignment of the nodes and ${m_1}^{(s_0)}$ be the number of nodes.

   1.4 call the procedure *Find_Weight_At_ith_Stage* to find weight. Let ${\omega_1}^1$ be the weight.

   1.5 decrease the demands by 1 ${\omega_1}^0$ times, for all nodes in the set ${V_1}^{(s_0)} \cup {R_1}^{(s_0)}$.

2. call the procedure *Iteration_For_*(f_1) , let α_1 be the number of times $f_1 = s_0$.

3. For i = 2 to α_1+1 do the following:

   3.1 call the procedure *Find_Nodes_Using_f_Frequency*, let ${n_i}^{f_1}$ be the number of nodes and $V_i^{(f_1)}$ be the set of nodes.

   3.2 do the frequency assignment by the algorithm *Assignment_On_the_Nodes* and ${A_i}^{(f_1)}$ be the frequency assignment on the nodes.

   3.3 call the procedure *Frequency_Reuse*, let ${R_i}^{(f_1)}$ be the set of nodes, ${RA_i}^{(f_1)}$ be the corresponding assignment of the nodes and ${m_i}^{(f_1)}$. be the number of nodes.

   3.4 call the procedure *Find_Weight_At_ith_Stage*, let ${\omega_i}^1$ be the weight.

   3.5 decrease the demands by 1 ${\omega_i}^{f_1}$ times, for all the nodes in the set $V_i^{(f_1)} \cup {R_i}^{(f_1)}$.

4. call the procedure *Iteration_For_*(f_2), let α_2 be the number of times $f_2 = s_0 + 1$.

5. For i = 1 to α_2 do the following

   5.1 call the procedure *Find_Nodes_Using_f_Frequency*, let ${n_i}^{f_2}$ be the number of nodes and $V_i^{(f_2)}$ be the set of nodes.

   5.2 do the frequency assignment by the algorithm *Assignment_On_the_Nodes* and ${A_i}^{(f_2)}$ be the frequency assignment on the nodes.

5.3 call the procedure $Frequency\_Reuse$, let ${R_i}^{(f_2)}$ be the set of nodes, ${RA_i}^{(f_2)}$ be the corresponding assignment of the nodes and ${m_i}^{(f_2)}$ be the number of nodes.

5.4 call the procedure $Find\_Weight\_At\_ith\_Stage$, let ${\omega_i}^2$ be the weight.

5.5 decrease the demands by 1 ${\omega_i}^2$ times, for all the nodes in the set $V_i^{(f_2)} \cup {R_i}^{(f_2)}$.

6. call the procedure $Iteration\_For\_(f_3)$, let α_3 be the number of times $f_3 = s_0 + 2$.

7. For i = 1 to α_3 do the following

7.1 call the procedure $Find\_Nodes\_Using\_f\_Frequency$, let ${n_i}^{f_3}$ be the number of nodes and $V_i^{(f_3)}$ be the set of nodes.

7.2 do the frequency assignment by the algorithm $Assignment\_On\_the\_Nodes$ and ${A_i}^{(f_3)}$ be the frequency assignment on the nodes.

7.3 call the procedure $Frequency\_Reuse$, let ${R_i}^{(f_3)}$ be the set of nodes, ${RA_i}^{(f_3)}$ be the corresponding assignment of the nodes and ${m_i}^{(f_3)}$ be the number of nodes.

7.4 call the procedure $Find\_Weight\_At\_ith\_Stage$, let ${omega_i}^3$ be the weight.

7.5 decrease the demands by 1 ${\omega_i}^{f_3}$ times, for all the nodes in the set $V_i^{(f_3)} \cup {R_i}^{(f_3)}$.

8. return total number of iteration $1 + \sum_{1 \le j \le 3} \alpha_j$

}
end

Once we have done proeprocessing then these results are available:
1) the set of nodes ${A_i}^{(f_j)} \cup {R_i}^{(f_j)}$ and number of nodes ${n_i}^{f_j} + {m_i}^{f_j}$ for $1 \le j \le 3$;
2) their frequency assignments ${A_i}^{(f_j)} \cup {RA_i}^{(f_j)}$;
3) weight vector ω.
Now we will do the final assignment of the nodes. In this stage we take the nodes in this manner: the nodes which we have found in preprocessing stage at the last step we take those set of nodes in the first stage and we find is there any other node(s) where the frequency assignment is possible. So we start with maximum value of f that we have used at preprocessing stage and then we decrease the value of f by 1 until $f = s_0$.

After $1 + \sum_{1 \le j \le 3} \alpha_j$ iterations, if there is any node with non-zero demand, we call that nodes are isolated node set(INS).
Algorithm: *Final_Assignment_On_Nodes*
Input: $V_i^{(f_j)}$ set of nodes, ${A_i}^{(f_j)}$ ($1 \le j \le 3$)are the assignments on the nodes and ${n_i}^{f_j}$ number of nodes in each set i, $1 \le i \le \alpha$ $1 \le j \le 3$. Weight vector ω and total number of iteration α. D initial demand vector.

Output: $FS_i^{(f_j)}$ set of nodes and $\eta_i^{f_j}$ number of nodes. $FA_i^{(f_j)}$ be the corresponding assignment on the nodes ($1 \leq i \leq \alpha$ $1 \leq j \leq 3$). $INS_{j_1}{}^f$ isolated nodes and $INA_{j_1}{}^f$ their assignments, $\xi_{j_1}{}^f$ be the number of nodes ($1 \leq j_1 \leq d_{max}$).
begin
Step 1: For each j ($3 \geq j \geq 1$) do the following:

1. For each $i = \alpha_j$ to 1, do the following:

   1.1 call the procedure *Frequency_Reuse* with input $V_i^{(f_j)}$, $A_i^{(f_j)}$ and $f = f_j$.

   1.2 let $RS_i^{f_j}$ be the set of nodes and $RSA_i^{f_j}$ be their assignments. $\beta_i^{f_j}$ be the number of nodes.

   1.3 $FS_i^{(f_j)} = V_i^{(f_j)} \cup RS_i^{f_j}$.is

   1.4 $FA_i^{(f_j)} = A_i^{(f_j)} \cup RSA_i^{f_j}$ and $\eta_i^{f_j} = n_i^{f_j} + \beta_i^{f_j}$.

   1.5 decrease the demand by 1 $\omega_i^{f_j}$ times, for all the nodes in the set $FS_i^{(f_j)}$.

Step 2: repeat the steps from 1.1 to 1.5 for the set of nodes $V_1^{(s_0)}$ with weight $\omega_1^{s_0}$.
Step 3: If $D[i] \neq 0$ $\forall i$, then take non-zero demand nodes.

1. find maximum demand among all non-zero demand nodes and let d_{max} be the maximum demand.

2. for $j = 1$ to d_{max} do the following:

   2.1 let $INS_{j_1}{}^f$ be the nodes. Check the chain formed by the nodes contains any triangle or not.

   2.2 if chain formed by the nodes contains any triangle then use frequency $f = s_0$. If the chain contains two consecutive triangle or a quadrilateral then use frequency $f = s_0 + 1$.

   2.3 call the procedure *Assignment_On_The_Nodes* with input $INS_{j_1}{}^f$. Let $INA_{j_1}{}^f$ be the assignments.

   2.4 call the procedure *Frequency_Reuse* with input $INS_{j_1}{}^f$, $INA_{j_1}{}^f$ and f. Let $S_{j_1}{}^f$ be the nodes and $A_{j_1}{}^f$ be their assignments returned by the algorithm *Frequency_Reuse*.Let $\xi_{j_1}{}^f$ be the number of nodes.

   2.5 $INS_{j_1}{}^f = INS_{j_1}{}^f \cup S_{j_1}{}^f$.

   2.6 $INA_{j_1}{}^f = INA_{j_1}{}^f \cup A_{j_1}{}^f$.

end

Algorithm: *Arrangement_Of_Frequencies*
Input: { $FS_i^{(f_j)}$ }, { $FA_i^{(f_j)}$} and $\eta_i^{f_j}$ ($1 \leq i \leq \alpha$ $1 \leq j \leq 3$). $INS_{j_1}{}^f$ isolated nodes,

$INA_{j_1}{}^f$ their assignments ($1 \leq j_1 \leq d_{max}$) and $\xi_{j_1}{}^f$ number of nodes.
Output: A conflict free assignment of the network and required bandwidth B.
begin
Step 1: Set $B = 0$.
Step 2: For each j ($3 \geq l \geq 1$) do the following:

For each $i = \alpha_j$ to 1, do the following:

1 assign the frequency channel $FA_i{}^{(f_j)}(t)$+B to the node $FS_i{}^{(f_j)}(t)$ until its demand exceeds, for $0 \leq t \leq \eta_i{}^{f_j} - 1$.

2 repeat the above step $\omega_i{}^{f_j}$ times and each time change $B = B + f_j$.

Step 3: For each $j_1 = 1$ to d_{max} do the following:

1. assign the frequency channel $INA_{j_1}{}^f(k)$+B to the node $INS_{j_1}{}^f(k)$ for $0 \leq k \leq \xi_i{}^f - 1$.
2. $B = B + f$.

Step 4: return B , number of frequency channel.
end

Chapter 4

Simulation Results

In this chapter we will take eight benchmark instances and show the results returned by our work. We take one most difficult problem and explain it step by step. Finally we would do the comparison between our approach and existing methods.

A set of benchmark problems has been defined on a hexagonal cellular network of 21 cells [3], [2].

D_1	8	25	8	8	8	15	18	52	77	28	13	15	31	15	36	57	25	8	10	13	8
D_2	5	5	5	8	12	25	30	25	30	40	40	45	20	30	25	15	15	30	20	20	25

In the above table D_1, D_2 are two demand vectors of the hexagonal cellular network of 21 cells. Eight problems (problems 1-8) are in terms of the specific values of s_0, s_1, s_2 and for the two-band buffering system and the corresponding demand vector used for each of them.

Problem	1	2	3	4	5	6	7	8
s_0	5	5	7	7	5	5	7	7
s_1	1	2	1	2	1	2	1	2
s_2	1	1	1	1	1	1	1	1
Demand	D_1	D_1	D_1	D_1	D_2	D_2	D_2	D_2

Problems 2 and 6 are most difficult ones. Now we describe the step by step solution to the problem 6 and present the complete result with 269 frequency channels.

4.1 Simulation Result for Problem 6:

In problem 6 $s_0 = 5$, $s_1 = 2$, $s_2 = 1$ and D_2 is the demand vector.
In Fig.1 each node has a label of the form [x], where x is the demand of that node. These

are the initial demand of each node. Now using above algorithms we show the results step by step.

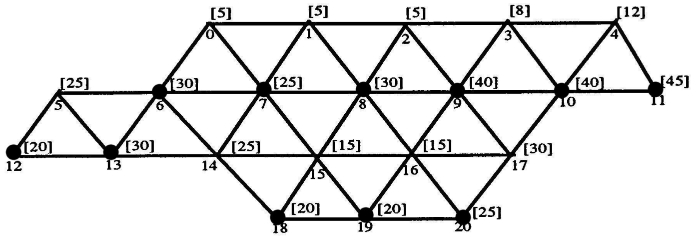


Fig.1

In the above network node 11 is the maximum demand node. $\{11, 10, 9, 8, 7, 6, 13, 12\}$ these nodes are found by the algorithm $Find\_Nodes\_Using\_f\_Frequency$ with $f = s_0 = 5$. The frequency assignment on the selected nodes are shown in the below table. Algorithm $Frequency\_Reuse$ selects $\{18, 19, 20\}$ these nodes along with their assignments. $\omega_1 = 6$ return by weight finding algorithm. Decrease the demands by ω_1 to the selected set of nodes.

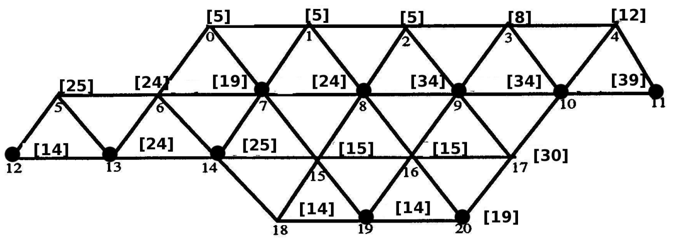


Fig.2

Fig.2 shows the structure of the network after one iteration. Similarly $\{11, 10, 9, 8, 7, 14, 13, 12\}$ these nodes are selected by the algorithm $Find\_Nodes\_Using\_f\_Frequency$ with $f = 5$. Algorithm $Frequency\_Reuse$ select nodes $\{19, 20\}$ along with their assignments. $\omega_2 = 2$. Decrease the demands by ω_2 to the selected set of nodes.

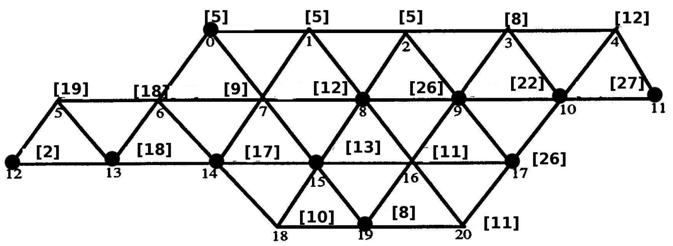


Fig.3

Fig.3 shows structure of the network after six iterations. Here $f = 6$, in this stage the path formed by the nodes contains a triangle. Node 10, 17, 9 forms a triangle.

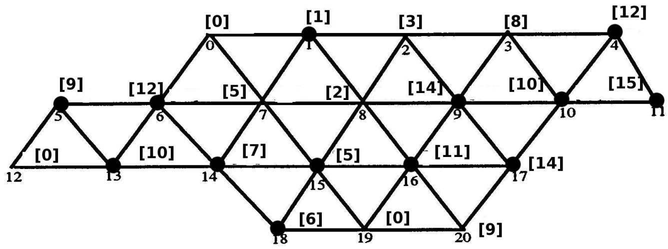


Fig.4

After four iterations, the above figure Fig.4 shows the demand of each node. Here $f = 7$.

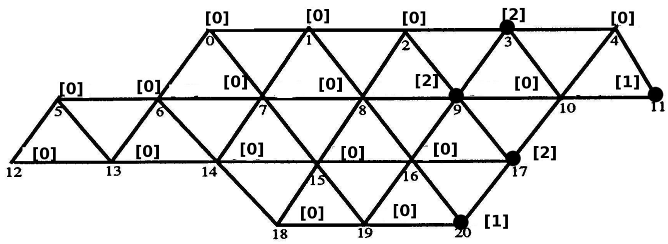


Fig.5

Fig.5 shows the stracture of the network after seven iteration. These nodes $\{3, 9, 11, 17, 20\}$ are isolated nodes. We do the assignments until all nodes demand equal to zero.

In the above table each row represents the frequency assignment on the selected nodes and ω_i represent the weight at the i-th step.
Each rows the nodes are selected by the algorithms $Find\_Nodes\_Using\_f\_Frequency$ and $Frequency\_Reuse$. The assignment is done by the algorithms $Assignment\_On\_The\_Nodes$ and $Frequency\_Reuse$.

Table 4.1: Frequency Assignment Table at Preprocessing Stage

Nodes	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	ω_i
$FA_1{}^{(s_0)}$							0	2	4	1	3	0	1	3					1	3	0	6
$FA_2{}^{(s_0)}$								2	4	1	3	0	1	3	0					3	0	2
$FA_3{}^{(s_0)}$						3	0	2	4	1	3	0	1						1	3	0	2
$FA_4{}^{(s_0)}$								2	4	1	3	0	1	3	0					3	0	2
$FA_5{}^{(s_0)}$						1	3	0	2		3	0	4				4	1	1			2
$FA_6{}^{(s_0)}$						1	3		8	1	3	0	4		0	2					0	2
$FA_7{}^{(s_0)}$						1	3	0	2		3	0	4	1	3		4	1				2
$FA_8{}^{(s_0+1)}$	4								2	5	3	0	4	1	3	0		1		4		2
$FA_9{}^{(s_0+1)}$	5					4	1		2	5	3	0			3	0		1		4		2
$FA_{10}{}^{(s_0+1)}$	4					5			2	5	3	0		1	3	0		1		4		4
$FA_{11}{}^{(s_0+1)}$		0	2			4	1	3		5	3	0			5			1	2		4	2
$FA_{12}{}^{(s_0+1)}$		4					3	0	2	5	3	0		1				1	4			2
$FA_{13}{}^{(s_0+2)}$		1			2	1				6	4	0		6	2	0	3	1	5			2
$FA_{14}{}^{(s_0+2)}$					2	1	4			6	4	0		6	2	0	3	1	5			2
$FA_{15}{}^{(s_0+2)}$				1	3	6	1	3	0	4		0		4		5	2	6	0			2
$FA_{16}{}^{(s_0+2)}$			0	5		1	4				3	0		6	2		3	1	0			2
$FA_{17}{}^{(s_0+2)}$			0		4	6				5	2	0		4	0		3				0	2
$FA_{18}{}^{(s_0+2)}$				4	2		0	2		1		0					3	5			0	2
$FA_{19}{}^{(s_0+2)}$					5			0		6	2	0						4			0	2
$FA_{20}{}^{(s_0)}$				0						2		1						4			1	1
$FA_{21}{}^{(4)}$				0						3								1				1

4.2 Comparison Of Results

In order to evaluate the performance of the algorithms for channel assignment problem, we should compare the lower bounds and execution time of the algorithms. Problems 2 and 6 are most difficult in literature. An efficient heuristic algorithm has been proposed in [7], which also produced non-optimal result for problems 2 and 6 with 463 and 273 channels, respectively. An efficient channel assignment algorithm has been proposed in [3], which produced optimal solution for problems 2 and 6 with execution time 60 sec and 72 sec respectively on an unloaded Sun Ultra 60 workstation. For problems 2 and 6, the number of the frequency channel required by our algorithm is at most 5-6% more than the optimal solution and for other six benchmark instances procuced optimal solutions. The exact number of channel required by our algorithm for problems 2 and 6 are 440 and 269 respectively. The execution time of our algorithm is at most 50 milisecond on an HPxw8400 workstation.

Problem	1	2	3	4	5	6	7	8
Time:(msec)	21	45	31	31	8	12	8	8

Table 4.2: Performance Comparison

Problem	1	2	3	4	5	6	7	8
Lower bounds	**381**	**427**	**533**	**533**	**221**	**253**	**309**	**309**
Our approach	**381**	**440**	**533**	**533**	**221**	**269**	**309**	**309**
(2003)[3]	381	427	533	533	221	253	309	309
(2001)[7]	381	463	533	533	221	273	309	309
(2001)[8]	381	427	533	533	221	254	309	309
(2000)[9]	381	433	533	533	-	260	-	309
(1998)[10]	381	427	533	533	221	253	309	309
(1998)[11]	-	-	-	-	221	268	-	309
(1997)[12]	381	-	533	533	221	-	309	309
(1997)[1]	381	436	533	533	-	268	-	309
(1996)[13]	381	-	533	533	-	-	-	-
(1996)[14]	381	433	533	533	221	263	309	309
(1994)[15]	381	464	533	536	-	293		310
(1992)[16]	381	-	533	533	221	-	309	309
(1989)[17]	381	447	533	533	-	270	-	310

4.3 Conclusion

We have done the assignment of the network in few steps. First we break the whole network into simpler sub-networks. Next we have assigned frequency channel to these sub-networks with homogeneous demand. Finally all these homogeneous assignment of the appropriate sub-networks of the given network together constitute the non-homogeneous assignment of the original netwok.

The proposed algorithm is able to achive the near-optimal solution for the problems 2 and 6 and optimal solutions for the remaining of six benchmark instances with less than 50 milisecond. Hence this algorithm will be very useful to cope up with rapid fluctuation in demands on the nodes of a network due to handoff or heavy traffic situations, which could otherwise be different with the existing algorithms providing optimal solutions but requiring large execution time(20-30 seconds). Thus, while the existing optimal algorithms can be used for long-term channel assignment, our proposed algorithm may be used for short-term assignment to satisfying the channel demand very quickly, altough in a near-optimal way.

Table 4.3: Complete Channel Assignment for Problem 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	58	1	3	0	1	5	0	6	57	5	2	3	62	218	4	2	61	1	72	0
9	65	8	10	7	8	12	7	13	64	12	9	10	69	223	11	9	68	8	79	7
16	72	15	17	14	15	19	14	20	71	19	16	17	76	238	18	16	75	15	86	14
23	79	22	24	21	22	26	21	27	78	26	23	24	83	243	25	23	82	22	93	21
30	86	29	31	28	29	33	28	34	85	33	30	31	90	256	32	30	89	29	100	28
37	93	36	38	35	36	40	35	41	92	40	37	38	97	261	39	37	96	36	107	35
44	100	43	45	42	43	47	42	48	99	47	44	45	104	269	46	44	101	43	113	42
51	107	50	52	49	50	54	49	55	106	54	51	52	111	274	53	51	108	50	120	49
	117				57	59	56	62	118	59	56	60	117	277	60	116		57	127	
	124				64	66	63	69	125	66	63	67	124	282	67	123		64	134	
	131				71	73	70	76	132	73	70	74	128	290	74	130			141	
	138				78	80	77	83	139	80	77	81	135	295	81	137			148	
	145				85	87	84	90	144	87	84	88	142	297	88	146			155	
	152				92	94	91	97	151		91	95	149	302	95	153				
	159				99	101	98	104	158		98	102	156	310	102	160				
	166					108	105	111	165			109		315	109	167				
	170					115	112	114	171			112		317	116	221				
	176					122	119	121	177			119		322	123	226				
	182						126	128	183			126		330	130	241				
	188						133	135	189			133		335	137	246				
	194						140	142	195			140		337	144	259				
	200						147	149	201			147		342	151	264				
	206						154	156	207			154		350	158	276				
	212						161	163	213			161		355	165	281				
	436						168	174	249			168		357	172	296				
							174	180	254			174		362	178	301				
							180	186	268			180		370	184	316				
							186	192	273			186		375	190	321				
							192	198				192		377	196					
							198	204				198		382	202					
							204	210				204		390	208					
							210	216						395	214					
							216	219						397	229					
							221	224						402	234					
							236	227						407	249					
							241	232						412	254					
							259	239							267					
							264	244							272					
							280	247							288					
							285	252							293					
							300	257							308					
							305	262							313					
							320	270							328					
							325	275							333					
							340	278							348					
							345	283							353					
							360	286							368					
							365	291							373					
							380	298							388					
							385	303							393					
							400	306							410					
							405	311							415					
								318							418					
								323							423					
								326							428					
								331							433					
								338							437					
								343												
								346												
								351												
								358												
								363												
								366												
								371												
								378												
								383												
								386												
								391												
								398												
								403												
								408												
								413												
								416												
								421												
								426												
								431												
								439												

Table 4.4: Complete Channel Assignment for Problem 6

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
119	94	153	221	155	43	0	2	4	1	3	0	1	3	30	72	64	61	43	0	2
125	100	160	228	162	48	5	7	9	6	8	5	6	8	35	77	69	66	48	5	7
142	130	167	235	169	61	10	12	14	11	13	10	11	13	50	90	84	81	61	10	12
148	136	174	242	176	66	15	17	19	16	18	15	16	18	55	96	89	86	66	15	17
164	150	195	249	183	71	20	22	24	21	23	20	21	23	70	102	151	91	105	20	22
			255	190	76	25	27	29	26	28	25	26	28	75	108	158	97	111	25	27
			261	197	94	40	32	34	31	33	30	31	33	83	114	165	103	131	40	32
				204	100	45	37	39	36	38	35	36	38	88	120	172	109	137	45	37
				211	105	63	42	44	41	43	40	41	53	93	138	179	115	150	94	42
				218	111	68	47	49	46	48	45	46	58	99	144	186	121	157	100	47
				237	118	73	52	54	51	53	50	51	81	117	155	193	127	164	118	52
				244	124	78	57	59	56	58	55	56	86	123	162	200	133	171	124	57
					131	103	65	67	71	63	60	64	91	141	169	207	139	184	142	70
					137	109	80	74	76	68	65	69	97	147	176	214	145	191	148	75
					150	129	85	79	95	73	70	74	115	152	197	220	154	199	153	106
					157	135	106	87	101	78	75	79	121	159			161	211	160	112
					165	145	112	92	107	83	80	84	127	166			168	218	167	130
					172	154	126	98	113	88	85	89	133	173			175	228	174	136
					184	161	132	104	119	93	90	102	143	194			182	234	195	224
					191	168	178	110	125	99	96	108	149	201			189	241	202	231
					200	175	185	116	131	105	102		170	208			196			237
					207	180	206	122	137	111	108		177	215			203			244
					214	187	213	128	143	117	114		182	223			210			251
					222	196	240	134	149	123	120		189	230			217			258
					229	203	247	140	156	129	126		198	236			240			266
						210		146	163	135	132		205				247			
						217		181	170	141	138		212				254			
						227		188	177	147	144		219				260			
						238		209	184	152	150		225				268			
						245		216	191	159	157		232							
									198	166	164									
									205	173	171									
									212	180	178									
									219	187	185									
									225	194	192									
									232	201	199									
									238	208	206									
									245	215	213									
									252	223	220									
									263	230	227									
											234									
											241									
											248									
											257									
											262									

Bibliography

[1] 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*, vol. 46, pp. 676–685, 1997.

[2] B. P. S. S. C. Ghosh and N. Das, “Coalesced cap:an improved technique for frequency assignment in cellular networks,,” *IEEE Trans. Veh. Technol.*, vol. vol. 55, no. 2, pp. 640–653, March. 2006.

[3] ——, “A new approach to efficient channel assignment for hexagonal cellular networks,” *Int. J. Foundations Comp. Sci.*, vol. vol.14, pp. 439–463, June 2003.

[4] D. Kunz, “Channel assignment for cellular radio using neural networks,” *IEEE Transactions on Vehicular Technology*, vol. 40, pp. 188–193, 1991.

[5] B. P. S. S. C. Ghosh and N. Das, “Channel assignment using genetic algorithm based on geometric symmetry,,” *IEEE Trans. Veh. Technol.*, vol. vol. 52, no. 4, pp. 860–875, Jul. 2003.

[6] ——, “Optimal channel assignment in cellular networks with non-homogeneous demands,,” *Technical Report CCSD/ACMU/03-2001.*

[7] G. Chakraborty, “An efficient heuristic algorithm for channel assignment problem in cellular radio networks,” *Vehicular Technology, IEEE Transactions on*, vol. 50, no. 6, pp. 1528–1539, Nov 2001.

[8] R. Battiti, A. Bertossi, and D. Cavallaro, “A randomized saturation degree heuristic for channel assignment in cellular radio networks,” *IEEE Transactions on Vehicular Technology*, vol. 50, pp. 364–374, 2001.

[9] D.-W. Tcha, J.-H. Kwon, T.-J. Choi, and S.-H. Oh, “Perturbation-minimizing frequency assignment in a changing tdma/fdma cellular environment,” *IEEE Transactions on Vehicular Technology*, vol. 49, no. 4, pp. 390–396, Mar. 2000.

[10] D. Beckmann and U. Killat, "A new strategy for the application of genetic algorithms to the channel-assignment problem," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 4, pp. 1261–1269, Jul. 1999.

[11] C. Y. Ngo and V. O. K. Li, "Fixed channel assignment in cellular radio networks using a modified genetic algorithm," *IEEE Transactions on Vehicular Technology*, vol. 47, pp. 163–171, 1998.

[12] J.-S. Kim, S. Park, P. Dowd, and N. Nasrabadi, "Cellular radio channel assignment using a modified hopfield network," *IEEE transactions on vehicular technology*, vol. 46, no. 4, pp. 957–967, 1996.

[13] P. W. D. J. S. Kim, S. H. Park and N. M. Nasrabadi, "Channel assignment in cellular radio using genetic algorithm," *Wireless Personal Commun.*, vol. vol.3, no. 3, pp. 273–286, Aug. 1996.

[14] W.Wang and C. K. Rushforth, "An adaptive local-search algorithm for the channel-assignment problem," *IEEE Trans. Vehicular Technology*, vol. vol. 45, no. 3, pp. 495–466, Aug. 1996.

[15] T.-M. ko, "A frequency selective insertion strategy for fixed channel assignment," *in Proc. 5th IEEE Int. Symp.*, pp. 311–314, Sept. 1994.

[16] N. Funabiki and Y. Takefuji, "A neural network parallel algorithm for channel assignment in cellular radio network," *IEEE Trans. Vehicular Technology*, vol. 41, pp. 430–437, Nov. 1992.

[17] R. J. K. N. Sivarajan and J. Ketchum, "Channel assignment in cellular radio," *Proc. 39th IEEE Vehicular Technology Conf.*, pp. 846–850, May 1989.