OVSF code allocation in CDMA based Wireless Ad Hoc Networks

Anurag Aggarwal
Department of Computer Science and Engineering,
Indian Institute of Technology,
Kanpur 208016
anuragag@cse.iitk.ac.in

Diwaker Gupta
Department of Computer Science and Engineering,
Indian Institute of Technology,
Kanpur 208016
gdiwaker@cse.iitk.ac.in

***Abstract*— Wireless ad hoc networks provide a new dimension to computing by allowing a group of portable devices to talk to each other without any additional infrastructure and have been the focus of research in the area of wireless networks. CDMA has been gaining popularity as the multiaccess technique and is part of many wireless standards now. In this project we present techniques to improve performance of the CDMA based wireless ad hoc systems and present results based on extensive simulations. The resource utilization of network in a CDMA based system depends largely on the code allocation scheme. Using a graph theoretic formulation of the problem, we explore solutions for code allocation problem under different requirements.**

***Index Terms*— Ad hoc networks, MANET, CDMA, OVSF codes, Graph Coloring, Range Sum**

I. Introduction

MOBILE ad hoc network (MANET) is an autonomous system of mobile nodes connected to each other through a fully mobile infrastructure. These networks are attractive due to the ease and speed of deployment. Although they haven't been really deployed but their use has been envisioned in wide variety of areas like military applications, conferences and rescue missions. Since these networks are in an incipient stage, there are no established standards for them. The IEEE802.11 series which has been gaining acceptance as the standard for wireless communication networks, also has provision of using Code Division Multiple Access(CDMA) for MANETs. The scheme proposed is a very simple one and does not try to maximize the use of available network resources. Also the support for MANETs is provided for the sake of completeness and is not truely based on an "infrastructure-less" network. In this project, we studied some of the issues in such CDMA based networks. We focused mainly on the OVSF code allocation schemes and looked at the performance of the algorithms suggested in the literature through simulations done partly in NS-2 and partly on our own simulator.

A. Organization of the report

We start this report by a brief introduction of CDMA in section II. This is accompanied by an overview of OVSF codes and the MC-CDMA system. In section III we discuss some issues concerning resource allocation in wireless ad-hoc network and motivate the need for optimal code allocation strategies, and hence this work. Section IV gives a formulation of the problem of code allocation to maximize aggregate throughput in terms of the concept of *Range Coloring* of a graph and some previous relevant results including the NP-Completeness of the problem. This is followed by a discussion on centralized code allocation schemes in section V. We start by presenting an approximation scheme proposed in an earlier work and the approximation bound for that algorithm. A more careful analysis of the algorithm gives a better approximation bound for the algorithm. We then present heuristics to improve the performance of the algorithm and a comparison of their performance based on simulations. Finally we conclude by presenting the conclusions in section VI.

II. Code Division Multiple Access

A. Overview

Code Division Multiple Access (CDMA) is a new and revolutionary concept in field of communications which allows multiple users to transmit at the same time in the same frequency band. Traditional multiple access techniques like TDMA and FDMA are based on the philosophy of letting no more than one transmitter occupy a given time-frequency slot. Whenever this condition is violated in random-access communication, the receiver is unable to recover any of the colliding transmissions. In CDMA we exploit the fact that reception free from interchannel interference is a consequence of the use of orthogonal signaling and it can be accomplished even by signals that overlap both in time and frequency. In a CDMA based system, users are assigned different "signature wave forms" or "codes". Each transmitter sends its data stream by modulating its own signature waveform as in a single-user digital communication system. The receiver does not need to concern itself with the fact that the signature waveforms overlap both in frequency and time, because their orthogonality ensures that they will be transparent to the output of the other user's correlator. CDMA derives advantage from the fact that the sharing of resources is inherently dynamic: reliability depends on the number of simultaneous users, rather than on the (usually much larger) number of potential users of the system.

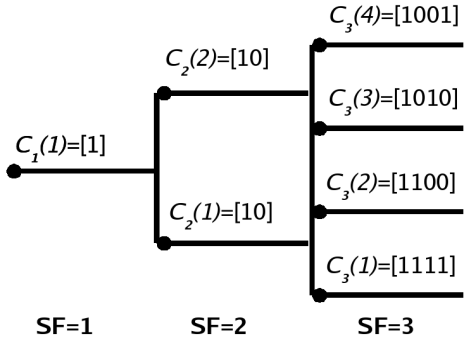

Fig. 1. Orthogonal Variable Spreading Tree Code

B. Spreading Codes

Spread spectrum communication uses much larger bandwidth than required by spreading original information signal using noise like sequences. A spread spectrum receiver then uses the synchronized replica of the noise like sequences to recover the original information. The spreading sequences form the basis of CDMA by providing resilience to interference at the cost of low bit rate. There are many well known spreading code generators and one of them is Walsh Codes

1) Walsh Codes: Two codes are said to be orthogonal if they have zero cross-correlation. Hadamard transform [2]is one of the best known techniques to generate orthogonal codes. Walsh codes are generated by applying Hadamard transform upon 1 repeatedly. Hadamard transform is given by

$$H_0 = [1]$$

$$H_n = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix}$$

Each row of the $2^n \times 2^n$ matrix H_n gives the 2^n bit code for a user.

2) Orthogonal Variable Spreading Factor Codes: CDMA provides support for variable spreading factor codes. These codes have different lengths and it is required that these variable length codes be orthogonal to each other. These codes can be generated using an algorithm based on code tree as shown in figure 1. Since during spreading, each information bit is multiplied by an entire codeword, it means that longer codes are associated with lower bit rates. When two messages with different codewords and spreading factors are transmitted at the same time, the shorter codeword, modulated by its information message, will get repeated a number of times for each transmission of one longer codeword. This means that the longer code that is derivable from a smaller code is not orthogonal. In terms of the tree structure of the codes, this translates to the condition that a code is not orthogonal to all the codes below it in the code tree. The codes in the tree are generated recursively by generating two codes from one code at each level. If $C_{N-1}(i)$ is one of the codes at the $N-1$ level, then we generate two codes $C_N(2i) = C_{N-1}(i) \oplus \overline{C_{N-1}(i)}$ and $C_N(2i-1) = C_{N-1}(i) \oplus C_{N-1}(i)$, where $\oplus$ denotes the concatenation operation.

C. MC-CDMA System

There are various variants of CDMA like DS-CDMA, MT-CDMA, MC-CDMA [3] etc. In our study we use MC-CDMA based on the simulation results on BER shown in [1]. MC-CDMA is a digital modulation technique where a single data symbol is transmitted at multiple narrowband subcarriers where each subcarrier is encoded with a phase offset of 0 or π based on a spreading code. The narrowband subcarriers are separated by a frequency $1/T_b$ at baseband, where T_b is the symbol duration. This results in the subcarriers being orthogonal to each other at baseband. At the receiver, by multiplying with the particular frequency of interest and summing over a symbol duration, we can isolate the symbol component at that subcarrier [4].

1) Transmitter: The MC-CDMA transmitter replicates $d^j{}_i$, the ith bit of the jth user, into N copies, where N is the number of subcarriers. Each copy thus obtained is multiplied by $C^j{}_i$, the ith bit of the signature code assigned to the jth user. This operation can be seen as spreading in the frequency domain of the data stream on to the available bandwidth W. These copies, in turn, modulate the N subcarriers, where the subcarrier separation, Δf, is $1/T_b$. These components are then added together to obtain the transmitted signal, $s^j(t)$, for the jth user. The transmitted signal for the jth user is given by:

$$s^j(t) = \sum_{i=-\infty}^{+\infty} \sum_{m=0}^{N-1} C^j_m d^j_i e^{j2\pi m\Delta f(t-iT_b)} p_{T_b}(t-iT_b)$$

where $p_{T_b}(t)$ is the rectangular symbol pulse waveform.

2) Receiver: The received signal is

$$\begin{aligned} r(t) &= \sum_{j=1}^{J} \int_{-\infty}^{+\infty} s^j(t-\tau) \otimes h^j(\tau;t) d\tau + n(t) \\ &= \sum_{i=-\infty}^{+\infty} \sum_{m=0}^{N-1} \sum_{j=1}^{J} z^j_m(t) d^j_i C^j_m p_s(t-iT_b) e^{j2\pi m\Delta f t} + n(t) \end{aligned}$$

where $z^j_m(t)$ is the received complex envelope at the mth subcarrier. The despreading operation is the inverse of the spreading operation done at the transmitter. The individual components, contributing to the decision variable, can be obtained by demodulating with their respective carrier frequencies, followed by multiplication by the corresponding code bit and the gain at that subcarrier, G^j_m, and integrating over one symbol duration. The gain is used to compensate for the distortion, in the amplitude and phase, introduced by the channel at the mth subcarrier. Finally the decision variable is given by

$$v^j(t = iT_b) = \sum_{m=0}^{N-1} G^j_m y(m)$$

$$y(m) = \sum_{j=1}^{J} z^j_m(iT_b) d^j c^j_m + n_m(iT_b)$$

where $y(m)$ and $n_m(iT_b)$ are the complex baseband component of the received signal and the complex additive gaussian noise at the mth subcarrier at $t = iT_b$ respectively.

3) Combining Schemes: Various combining schemes have been proposed to determine the gain. These techniques are essentially heuristics designed to reduce the effect of fading and interference while not enhancing the effect of noise on the decision. Some of the schemes commonly used are EGC, MRC, ORC and MMSEC [4]. In our simulations we have used EGC due to its simplicity. In Equal Gain Combining (EGC) the gain factor is given by

$$G_m^j = c_m^j z_m^{j*} / |z_m^j|$$

which implies that the multiplication with the received signal's complex envelope will result in the amplitude being passed as it is.

III. Resource Management in Wireless Ad Hoc Networks

Designing medium access protocols which can allow optimal use of the available channel resources has been a challenging problem. Traditionally, the MAC protocols worked by dividing the channel in terms of frequency or time and allowing a particular user exclusive access to that resource. These protocols were plagued with the problem of low utilization of resources due to lack of proper coordination between the nodes. But as discussed earlier CDMA allows multiple users to communicate with each other at the same time,in the same spatial region and the available bandwidth is shared among the nodes.

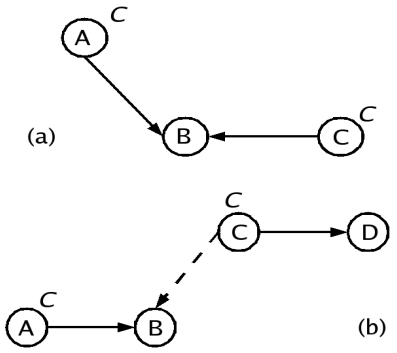

Fig. 2. Two scenarios involving secondary interference

A. Issues in CDMA based MAC

Code allocation is an important part of any CDMA based MAC protocol and it has severe impact on the performance of the protocol itself. In cellular systems, all the users are assigned codes by a central authority but in a MANET it has to be done in a distributed manner. The basic requirement of any code allocation scheme is that should prevent any interference between various transmitter-receiver pairs by allocating then orthogonal codes. Interference between nodes can be of two types:

1) *Primary Interference* : Interference due to two neighbors transmitting to each other at the same time using the same code
2) *Secondary Interference* : This type of interference occurs when two or more senders' transmission interferes at one receiver. There can be two possible scenarios in this type of interference as shown in figure 2. In the first case, both the transmitters were trying to communicate with the common receiver, thus resulting in an interference. In the second case, one or both the transmitters were not explicitly trying to communicate with the interfering receiver but the receiver happened to receive the signal as it was in the communicating range of the sender.

B. Different Approaches for formulation of problem

Due to above problems, different code allocation policies have been considered. Many approaches have been proposed for code assignment using fixed length codes but there are not many results for the variable length codes. In [5], authors have studied the allocation of variable length codes in schemes which solve only a subset of the problems that occur in MANETs. These schemes are :

1) *Receiver-based Code Assignment(RCA)* : Here each node is assigned a receiving code such that no two neighbors of any node are assigned the same code. In this scheme the receiver hears to only one code but there can be primary interference
2) *Transmitter-based Code Assignment(TCA)* : Here each node is assigned a transmitter code such that all neighbors of a node have different transmitter code. This scheme avoids primary interference but is prone to secondary interference
3) *Pairwise Code Assignment(PCA)* : Here each pair of transmitter-receiver is assigned a unique code such that no two edges in topology have conflicting codes. This scheme also does not completely alleviate the problem of secondary interference

As it is clear from the above schemes that they have been designed more for the ease of implementation rather than completely avoiding the problems occurring in MANETs. The formulation of the problem used in [1] is more general and we would be using it for the rest of the discussion in the report.

IV. Graph Theoretic Formulation of Problem

The problem of code allocation can be formulated as a graph theoretic problem. A MANET can be considered as an undirected graph $G_{topo} = (V, E)$, called the topology graph, where set V represents the set of mobile nodes and E is the set of edges corresponding to a link between adjacent mobile nodes in the MANET. We define a new graph $G_{comm} = (V', E')$, called the communication graph, to represent the communication going on in a MANET at an instant. Here

- $V' = \{ v_{(i,j)} \mid i \text{ and } j \text{ are a transmitter-receiver pair } \}$
- $E' = \{ (v_{(i,j)}, v_{(p,q)}) \mid i \text{ is within communicating range of } q \text{ or } p \text{ is within communicating range of } j \}$

So the problem of allocating codes in the MANET reduces to allocating codes to these nodes from the available set of OVSF codes so that no two neighbors are assigned non-orthogonal

codes. To complete the graph theoretic formulation of the problem, the concept of Range Sum of a graph was introduced.

A. Range Sum of a graph

The range sum of a graph is a variant of the chromatic sum [6] problem. The chromatic sum of a graph is the minimum sum of the color of vertices over all colorings of the graph with natural numbers. In range sum problem, we color the vertices with ranges which are obtained by recursively dividing the range $[0,1)$ into σ parts with the restriction that no two neighbors have ranges with non-zero intersection. Formally,

Definition Define the set $R_\sigma(k)$ with parameters σ and k as:

- $R_\sigma(0) = [0,1)$
- $R_\sigma(k+1) = \{[l(r) + (t-1) * \frac{|r|}{\sigma}, l(r) + t * \frac{|r|}{\sigma}) \mid t \leq \sigma, \forall r \in R_\sigma(k)\}$
  where $l(r)$ denotes the left end of the range r.

From now on $R_\sigma(k,i)$ would be used to refer to the ith element of $R_\sigma(k)$ and $\mathcal{R}_\sigma$ would be used to denote the set of all $R_\sigma(k)$'s i.e. $\mathcal{R}_\sigma = lim_{k\to\infty} \bigcup_{i=0}^{k} R_\sigma(k)$.

So now we can define a proper *range coloring* of a graph $G = (V,E)$ as a coloring of the nodes of the graph with ranges $r \in \mathcal{R}_\sigma$ such that no two neighbors have overlapping ranges i.e. $r(u) \cap r(v) = \phi$ when $(u,v) \in E$.

The *Range Sum* of the graph G, $\Gamma_\sigma(G)$ is defined to be maximum sum of the lengths of the ranges assigned to the nodes over all possible proper colorings of the graph.

B. Previous Results

In [1], the following important results were shown regarding the Range Sum of the graph:

1) **NP-Completeness of the Range Sum problem**: The decision problem regarding the existence of a range coloring with range sum $\geq k$ was shown to be NP-complete. So we cannot have a polynomial time algorithm for computing the range sum of a graph
2) **Equivalence of code allocation for throughput maximization and Range Sum problem**: It was shown that the problem of allocating OVSF codes to maximize the aggregate throughput of the network could be reduced to finding the range sum of the communication graph at an instant.

Subsequently both centralized and distributed approximation algorithms for finding the optimal code allocation were proposed. These would be discussed in more detail in the following sections

C. Some Observations

- Any clique of a graph cannot have range sum greater than 1 i.e. $\sum_{i=1}^{k} | r(v_i) | \leq 1$ if $v_1, v_2, \ldots, v_k$ form a clique.
- The optimal range sum of a bipartite connected graph with n nodes is n/σ. Since there are no edges connecting two vertices in one bipartite set, so the whole set can be colored by a range r having $| r | = 1/\sigma$. Similarly the other set can also be colored with a different range r_1 also having $| r_1 | = 1/\sigma$ as $\sigma \geq 2$. If there is a range assignment which can do better than this should have atleast one node with range r such that $| r | > 1/\sigma$ but the only such range is $r = [0,1)$. But this would mean that this node's neighbors don't get any code. Hence this is the range sum for the bipartite graph.
- The range coloring problem can also be modeled as the Graph Prefix Free Code Assignment Problem discussed in [7]. In this problem, we have to assign prefix free codes to the nodes such that for any edge (i,j) the codes C_i and C_j are not prefixes of each other. Prefix free codes have the property that no codeword is a prefix of another codeword.

V. Centralized Code Allocation Schemes

Since the range sum problem for arbitrary graphs was shown to be NP-complete, we look at some heuristics for allocation of code.

A. Previous Results

In [1], a greedy range coloring algorithm for sparse graphs is described. The algorithm refers to ith range in the lexicographic ordering on the index, (k,t), of the ranges in $\mathcal{R}_\sigma$ as $Range_\sigma(i)$. The algorithm is as follows:

Algorithm **Greedy-Range-Color(G,σ):**
Input: An undirected graph $G = (V,E)$ and integer parameter σ.
Output: A proper range coloring $r : V \to \mathcal{R}_\sigma$ of G
For each node $v \in V$ maintain:

- $r(v)$, the range(color) assigned to v, initialized to ϕ
- $Unused(v)$, the subset of [0,1) not being used to color any node u adjacent to v; or more formally

$$Unused(v) = [0,1) - \bigcup_{\forall u;(u,v)\in E} r(u)$$

Algorithm:
Step 1: Determine an ordering of vertices $v \in V$, $v_1, v_2, \ldots, v_n$
Step 2: Consider the vertices in the order determined above. Let the current vertex be v_r. Find the least value of t such that

- $Range_\sigma(t) \cap Unused(v_r) = Range_\sigma(t)$ and
- $Unused(u) - Range_\sigma(t) \neq \phi$, $\forall u$ satisfying $(u,v_r) \in E$ and $r_(u) = \phi$

Color v_r with $Range(t)$, i.e. $r(v_r) = Range_\sigma(t)$ and update $Unused(u) = Unused(u) - r(v_r)$, $\forall u, (u,v_r) \in E$. Repeat until there is some uncolored vertex.

The above algorithm was shown to find a $\sigma^{\frac{\sigma+\overline{d}}{\sigma-1}}$-approximation to the range sum of graph G where $\overline{d}$ is the average degree of the graph G. This bound is good when the graph is sparse but not in a dense graph. In the next section, we provide a bound for dense graphs.

B. Approximation Bound of the Greedy-Range-Color algorithm on Dense Graphs

Theorem 5.1: For each G with number of nodes n and maximum degree Δ, algorithm Greedy-Range-Color is a $n * O(1)$ approximation to $\Gamma_\sigma(G)$.

Proof: For an ordering of the vertices of a graph, *lower degree*, l_i, of the vertex i is the number of lower indexed neighbors of i. Similarly, *higher degree*, h_i, is the number of higher indexed neighbors of i. Note that $l_1 = 0$, $l_2 \leq 1$, $l_3 \leq 2$ and so on. In general $l_i \leq \Delta$ where Δ is the maximum degree of any vertex in the graph. From the definition it is clear that $\sum l_i = \sum h_i = e$, the number of edges in the graph. Let $n = |V|$ be the order of the graph G and $v_1, v_2, \ldots, v_n$ be the vertices of G in the order in which they are chosen by the algorithm. At most l_i colors and their induced subranges are forbidden colors for the vertex v_i. Therefore v_i is colored with a range that contributes at least $\frac{1}{\sigma^{\lceil \frac{1+l_i}{\sigma-1} \rceil}}$. Another thing to note is that no node will be colored with range less than $\frac{1}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}}$ because if a node v_i has $l_i < \Delta$, then by the bound given above the results holds. If a node v_i has $l_i = \Delta$, it would mean that all its neighbors have been colored already and hence this node doesn't need to leave any range and it can take up range $r(v_i)$ with $|r(v_i)| = \frac{1}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}}$

So the sum of the ranges allocated by the Greedy-Range-Color, $\Gamma_\sigma^{greedy}(G)$, is given by

$$\begin{aligned}
\Gamma_\sigma^{greedy}(G) &= \sum_{v \in V} |r(v)| \\
&\geq \sum_{v \in V} \frac{1}{\sigma^{\lceil \frac{1+l_i}{\sigma-1} \rceil}} \\
&\geq \left(\frac{1}{\sigma^{\lceil \frac{1}{\sigma-1} \rceil}} + \frac{1}{\sigma^{\lceil \frac{2}{\sigma-1} \rceil}} + \ldots + \frac{1}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}}\right) + \frac{n-\Delta}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}}
\end{aligned}$$

If $\Delta \leq (\sigma - 1)$, then

$$\Gamma_\sigma^{greedy}(G) = \sum_{v \in V} \frac{1}{\sigma} = \frac{n}{\sigma}$$

This is the optimal as all nodes have been allocated codes from the first level of the code tree itself.

If $\Delta > (\sigma - 1)$, then

$$\begin{aligned}
\Gamma_\sigma^{greedy}(G) &\geq (\sigma-1)\left(\frac{1}{\sigma} + \frac{1}{\sigma^2} + \ldots + \frac{1}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil - 1}}\right) + \frac{n-\Delta}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}} \\
&= \frac{(\sigma-1)}{\sigma}\left(\frac{1 - \frac{1}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil - 1}}}{1 - \frac{1}{\sigma}}\right) + \frac{n-\Delta}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}} \\
&= \frac{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil} - \sigma}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}} + \frac{n-\Delta}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}} \\
&= \frac{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil} - \sigma + n - \Delta}{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}} \\
&= n \times \frac{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil} - \sigma + n - \Delta}{n\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}} \\
&\geq n \times \frac{\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil - 1}}{n\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil}} \\
&= \frac{n}{n\sigma} \geq \frac{\Gamma_\sigma(G)}{n\sigma}
\end{aligned}$$

In the first step we drop some of the terms in the sum and use the formula for summing up a Geometric Progression. In the second last step, we used the fact that $n > \Delta$ and $\sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil} - \sigma > \sigma^{\lceil \frac{\Delta}{\sigma-1} \rceil - 1}$. The last inequality follows from the trivial $n \geq \Gamma_\sigma(G)$ bound. Since σ is a constant for a given set of graphs, so the Greedy-Range-Color finds a $n * O(1)$-approximation to the range sum of G. ■

C. New Heuristics

In the above algorithm it is important to observe that the order in which vertices are chosen can greatly effect the performance of the algorithm. We tried out various ordering of the vertices and compared their performance. Before listing out the heuristics used, we would like to define some terms which would be used in describing the heuristics.

Definition The *saturation degree* of a node during the coloring procedure of a graph is defined to be the number of different colors that are adjacent to it (neighbors which have already been assigned colors) [8]

Definition The *saturation range* of a node during the range coloring procedure of a graph is defined to be the union of the ranges already assigned to the neighbors.

The various ordering heuristics used were:

- **Random Order** In this heuristic, the algorithm chooses the vertices in a random order. This corresponds to the proposal in the original algorithm.
- **Increasing Degree Order** In this heuristic, the algorithm chooses the vertices in increasing order of their degree.If two vertices have the same degree, any one of them is chosen
- **Decreasing Degree Order** In this heuristic, the algorithm chooses the vertices in decreasing order of their degree. If two vertices have the same degree, any one of them is chosen
- **Increasing Saturation Degree Order** In this heuristic, the algorithm chooses the vertices in increasing order of their saturation degree. The colors used for maintaining the saturation degree are maintained independent of the range colors. The first vertex chosen by the algorithm is the one having minimum degree. In case two vertices have the same saturation degree, the vertex with minimum degree in the uncolored subgraph is chosen.
- **Decreasing Saturation Degree Order** In this heuristic, the algorithm chooses the vertices in decreasing order of their saturation degree.The colors used for maintaining the saturation degree are maintained independent of the range colors. The first vertex chosen by the algorithm is the one having minimum degree. In case two vertices have the same saturation degree, the vertex with minimum degree in the uncolored subgraph is chosen.
- **Increasing Saturation Range Order** In this heuristic, the algorithm chooses the vertices in increasing order of their saturation range.The first vertex chosen by the algorithm is the one having minimum degree. In case two vertices have the same saturation degree, the vertex with minimum degree in the uncolored subgraph is chosen.

- **Decreasing Saturation Range Order** In this heuristic, the algorithm chooses the vertices in decreasing order of their saturation range.The first vertex chosen by the algorithm is the one having minimum degree. In case two vertices have the same saturation degree, the vertex with minimum degree in the uncolored subgraph is chosen.

After ordering the vertices, we allocate the codes in the greedy way as described in the original algorithm.

1) Properties of the ordering algorithms: The heuristics stated in the previous section are adapted from the sequential graph coloring approximation algorithms.

Lemma 5.2: The Decreasing Saturation Degree Order and Decreasing Saturation Range Order combined with the greedy algorithm give the optimal range sum for the bipartite graphs [8].

Proof: The Decreasing Saturation Degree Order generates a bipartite coloring of the graph as proved in [8]. This implies that any node which is chosen never has a saturation degree of more than 1 during the coloring. So it can always be assigned a range r with $\mid r \mid = 1/\sigma$ as $\sigma \geq 2$.

The Decreasing Saturation Range Order will also generate a range coloring for the bipartite graph which gives the range sum of the graph. The first node chosen by the algorithm will be assigned a range $r = [0, 1/\sigma)$. The next vertex that would be chosen would be from the other bipartite set. It would be assigned a range $r = [1/\sigma, 2/\sigma)$. Now any vertex that would be chosen would have either $[0, 1/\sigma)$ or $[1/\sigma, 2/\sigma)$ as the saturation range. If a node has a saturation range that which is a union of 2 or more ranges, then it must have two neighbors with different ranges. Starting from these neighbors, we can construct two chains. Since G is finite, there must be some common vertex y between the two chains. Either this cycle should have even length or the graph is not bipartite. If the cycle has even length, then the two neighbors must have same range. ■

2) A heuristic based on coloring: An algorithm based on graph coloring was also designed for the range allocation problem. The algorithm is described below:

Algorithm **Greedy-Frequency**

Step 1: Color the graph using some approximation algorithm for optimal graph coloring. In our case we used the saturation degree based approximation algorithm.

Step 2: Find the *maximal* coloring corresponding to the coloring discovered in previous step. The maximal coloring with respect to a coloring can be defined in the following way: Suppose that in the original coloring of a graph $G = (V, E)$ with $|V| = n$, the colors $c_1, c_2, \ldots, c_m$ are assigned to $k_1, k_2, \ldots, k_m$ nodes, respectively, in the graph where $k_1 \geq k_2 \geq \ldots \geq k_m$ and $\sum_{i=1}^{m} k_i = n$. To find a maximal coloring corresponding to this coloring, we consider the colors c_i in order and try to increase the number of nodes colored by c_i (k_i) by swapping color $c_j, j > i$ with c_i where this swapping still results in a proper coloring.

Step 3: Now consider the problem of code allocation to be that of assigning prefix free codes $C_1, C_2, \ldots, C_n$ to colors having usage frequencies as $k_1, k_2, \ldots, k_m$ with the aim of maximizing the sum $S = \sum_{i=1}^{m} k_i/\sigma^{l_i}$. Here σ^{l_i} is the length of the prefix free code assigned to color c_i. The optimal solution for this problem is based on a greedy algorithm which considers the colors in decreasing order of frequencies k_i and assigns the shortest prefix free code which does not cause any conflicts with the already assigned colors. So the greedy algorithm assigns prefix free code C_i with length σ^{l_i} to color c_i and then every node in the graph having color c_i is assigned a range with length $1/\sigma^{l_i}$. Note that there is one to one correspondence between the prefix free codes and ranges similar to one we had between OVSF codes and ranges. So there should be no problem in assigning a unique range corresponding to a prefix free code.

The reduced problem of assigning orthogonal codes to the colors according to the frequency of the color is solved optimally by the greedy method. This can be shown through the following lemmas.

Lemma 5.3: Let C be a color alphabet in which each color $c \in C$ has frequency $f[c]$. Let x and y be two colors in C having the lowest frequencies. Then there exists an optimal prefix code for C in which the codewords for x and y have the same length and differ only in the last bit.

Proof: In this proof we take the tree T representing an arbitrary optimal prefix code and modify it to make a tree representing another optimal prefix code such that the colors x and y appear as sibling leaves of maximum depth in the new tree. Let b and c be two colors that are sibling leaves of maximum depth in T. Without loss of generality, we assume that $f[b] \leq f[c]$ and $f[x] \leq f[y]$. Since $f[x]$ and $f[y]$ are two lowest leaf frequencies, in order, and $f[b]$ and $f[c]$ are two arbitrary frequencies, in order, we have $f[x] \leq f[b]$ and $f[y] \leq f[c]$. As shown in figure 3, we exchange the positions in T of b and x to produce a tree T', and then we exchange the positions in T' of c and y to produce a tree T''. The difference in cost between T and T' is

$$\begin{aligned} B(T') - B(T) &= \sum_{c \in C} \frac{f[c]}{l_{T'}(c)} - \sum_{c \in C} \frac{f[c]}{l_T(c)} \\ &= \frac{f[x]}{l_T(x)} + \frac{f[b]}{l_T(b)} - \frac{f[x]}{l_{T'}(x)} - \frac{f[b]}{l_{T'}(b)} \\ &= \frac{f[x]}{l_T(x)} + \frac{f[b]}{l_T(b)} - \frac{f[x]}{l_T(b)} - \frac{f[b]}{l_T(x)} \\ &= (f[b] - f[x])(1/l_T(b) - 1/l_T(x)) \\ &\geq 0 \end{aligned}$$

because both $f[b] - f[x]$ and $1/l_T(b) - 1/l_T(x)$ are nonnegative. More specifically. $f[b] - f[x]$ is nonnegative because x is a minimum-frequency leaf, and $1/l_T(b) - 1/l_T(x)$ is nonnegative because b is a leaf of maximum depth in T. Similarly, because exchanging y and c does not decrease the color-frequency sum, $B(T'') - B(T)$ is nonnegative. Therefore, $B(T'') \geq B(T)$, and since T is optimal, $B(T) \geq B(T'')$, which implies $B(T'') = B(T)$. Thus, T'' is an optimal tree in which x and y appear as sibling leaves of maximum depth, from which the lemma follows.

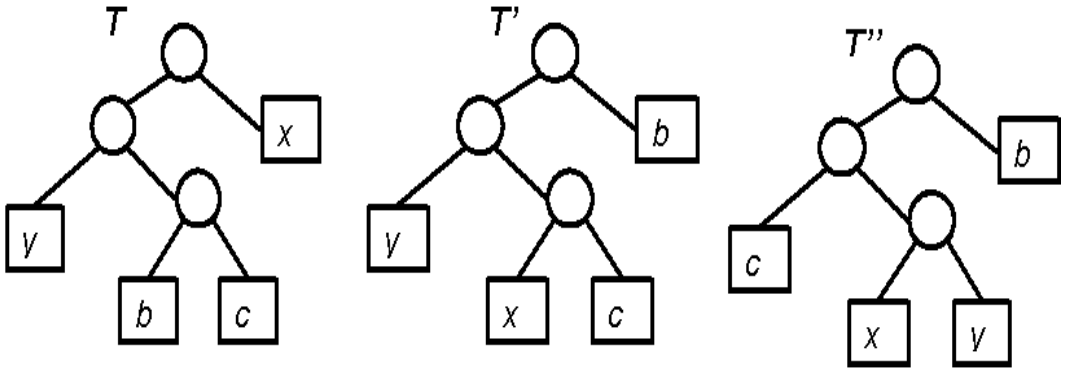

Fig. 3. Illustration of key step in proof of optimality for color based heuristic

■

Lemma 5.4: Let T be a full binary tree representing an optimal prefix code over a color alphabet C, where frequency $f[c]$ is defined for each color $c \in C$. Consider any two colors x and y that appear as sibling leaves in T, let z be their parent. Then, considering z as a color with frequency $f[z] = (f[x] + f[y])/2$, the tree $T' = T - x,y$ represents an optimal prefix code for the alphabet $C' = C - x,y \cup z$.

Proof: We first show that the cost $B(T)$ of tree T can be expressed in terms of the cost $B(T')$ of tree T' by considering the component costs. For each $c \in C - x,y$, we have $l_T(c) = l_{T'}(c)$, and hence $\frac{f[c]}{l_T(c)} = \frac{f[c]}{l'_T(c)}$. Since $l_T(x) = l_T(y) = l'_T(z)$, we have

$$\frac{f[x]}{l_T(x)} + \frac{f[y]}{l_T(y)} = \frac{(f[x] + f[y])}{l_{T'}(z)} = \frac{f[z]}{l_{T'}(z)}$$

from which we conclude that $B(T) = B(T')$.

If T' represents a nonoptimal prefix code for the alphabet C', then there exists a tree T'' whose leaves are colors in C' such that $B(T'') > B(T')$. Since z is treated as a color in C', it appears as a leaf in T''. If we add x and y as children of z in T'', then we obtain a prefix code for C with cost $B(T'') > B(T)$, contradicting the optimality of T. Thus, T' must be optimal for the alphabet C'.

■

The assignment of codes suggested basically reduces to the greedy assignment depending upon the frequency of the colors as the color z formed by replacing the two least frequency nodes,x and y, has the least frequency among all the colors in the new alphabet C'. So now the color z will be one of the leaves. This way we get a greedy assignment of codes

Although the reduced problem described in last step of the algorithm is solved optimally but this does not give an optimal solution to the original problem even if we could find an optimal coloring of the given graph. This can be understood by the fact that by reducing the problem to that given in last step, we are assuming that all other colors are conflicting with a given color assignment at every node. This may not be true in general.

D. Simulation Results

For the comparison of these heuristics, a simple simulator consisting of two components – graph generator and algorithm simulator – was designed. The graph generator generates a series of communication graphs according to the following parameters:

- Number of graphs to be generated
- σ to be used for calculation of range sum
- The number of nodes in the topology graph
- The range of the nodes in terms of radius of transmission sphere as a fraction of the length of the square in which all nodes are located.
- The number of connections established

The graph generator first generates a topological graph with the number of nodes specified. The neighbors of any node are computed based on the range specified. Using this topological graph, the communication graph G_{comm} is constructed by generating random connections between the neighbors in topological graph. The algorithm simulator then runs the specified algorithm on the communication graph provided and calculates the approximate range sum for the σ specified. We implemented all the heuristics mentioned in previous section and compared their performance by doing simulations. For all the simulations done, σ was kept equal to 2. The simulations were done by varying two parameters:

- **Range of the graph**: This compares the performance of the algorithms by increasing the range of nodes in topology graph and thereby increasing the density of the graph. For the whole set of simulations in this case, the number of nodes was 60 and number of connections was 15. As seen by the results of simulation the range sum computed by the algorithms initially increases because

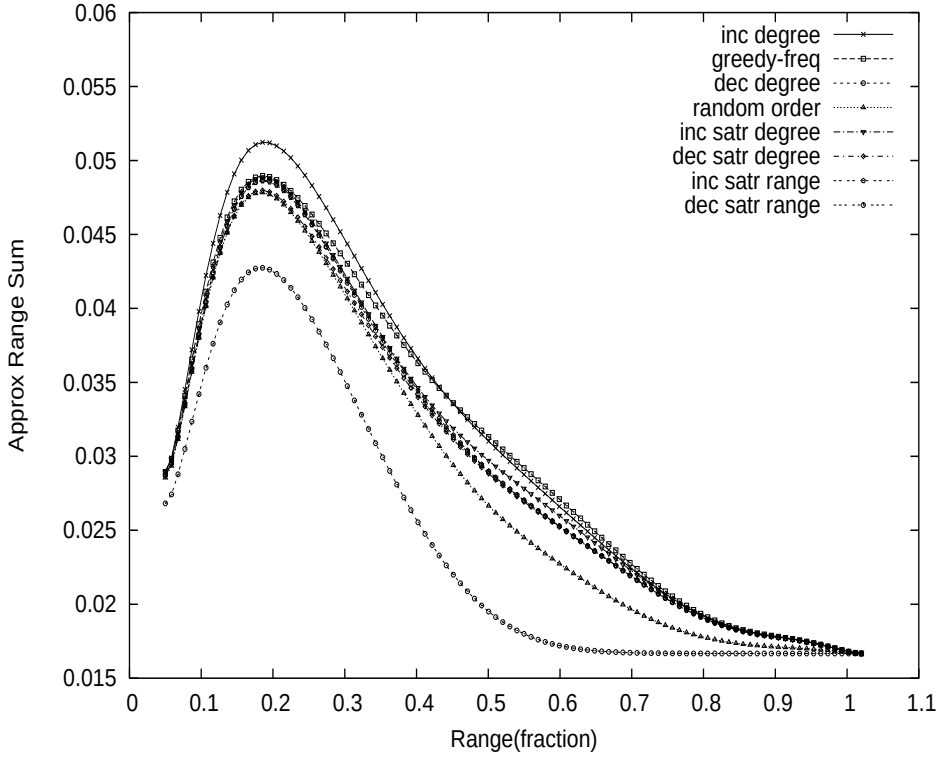

Fig. 4. Approx. Range Sum vs Range

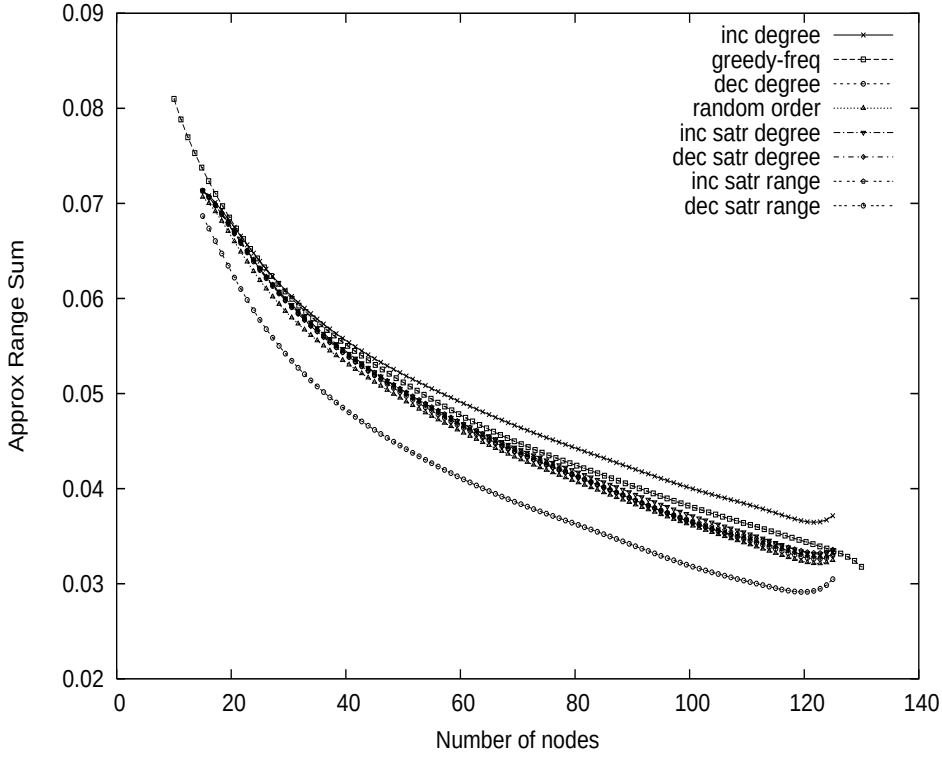

Fig. 5. Approx. Range Sum vs Number of Nodes

with very less density, the topology graph is very sparse and not many links can be formed. So the number of communications that can take place in the system are very less. After reaching a peak value, the approximate range sum starts decreasing with increasing range because increasing density implies that in the communication graph a node will have more neighbors and hence the range that would be allocated to it would be smaller. Among the heuristics, the one based on choosing vertices in increasing degree and greedy-frequency based algorithm seem to be doing better than others.

- **Number of nodes in the graph**: This compares the performance of algorithms on increasing the number of nodes in the topological graph keeping the range constant. For this set of simulation, range was kept constant at 0.2 and the number of connections was increased in proportion to the number of nodes to maintain the same theoretical throughput level. In this case the approximate range sum of the graph per node (which gives a measure of the throughput per node) decreases. This is due to the fact that by increasing the number of nodes, the approximation is applied for more nodes and hence the average falls. Here also the increasing degree based heuristic and greedy-frequency based algorithm seem to perform best.

From the simulation results stated above it is clear that the heuristics based on increasing degree and greedy-frequency seem to performing better than others. In practical situation, the algorithm based on increasing degree might be better as it is easier to implement and is faster than the greedy-frequency algorithm.

VI. Conclusion

The heuristics presented above improve the performance of the greedy code allocation algorithm. The heuristic based choosing the nodes in increasing degree order and then using the greedy scheme peforms the best. The coloring based heuristic also gives comparable performance.

VII. Future Work

To actually implement the code allocation schemes in practice, we need to have distributed algorithms based on the centralized code allocation schemes. So a possible extension of this work could be to design distributed algorithms which would do the same work as centralized algorithms at any instant. In dynamic schemes, we also need to consider other factors like the message passing cost between neighbors. So an efficient distributed version of the centralized algorithms needs to be designed carefully.

Acknowledgments

We are grateful to Dr. Dheeraj Sanghi for his guidance throughout this work. We would also like to thank Dr. Ajit Kr. Chaturvedi and Saurabh Srivastava for their help and cooperation.

References

[1] D. S. S. Srivastava, S. Tripathi and A. Chaturvedi, “Resource Optimization in CDMA based Wireless Ad Hoc Networks,” BTP Report, April 2002.

[2] R. Peterson, R. Ziemer, and D. Borth, *Introduction to Spread Spectrum Communications*. Prentice Hall International Editions, 1995.

[3] S. Hara and R. Prasad, “Overview of Multicarrier CDMA,” *IEEE Communications Magazine*, December 1997.

[4] J. L. N.Yee and G. Fettweis, “Multi-Carrier CDMA in indoor wireless networks,” in *PIMRC ’93*, Yokohama, September 1993, pp. 109–113.

[5] L. Hu, “Distributed code assignments for CDMA Packet Radio Network,” *IEEE/ACM Transactions on Networking (TON)*, vol. 1, no. 6, pp. 668–677, 1993.

[6] E. Kubicka and A. J. Schwenk, “An introduction to chromatic sums,” in *Proceedings of the seventeenth annual ACM conference on Computer science : Computing trends in the 1990’s*. ACM Press, 1989, pp. 39–45.

[7] N. Narayanaswamy and C. V. Madhavan, “On Assigning Prefix free codes to the vertices of a graph,” in *Computing and Combinatorics*, J. Wang, Ed. Springer, August 2001, pp. 328–337.

[8] D. Brlaz, “New methods to color the vertices of a graph,” *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.