Telecommun Syst (2013) 52:1719–1729
DOI 10.1007/s11235-011-9469-5

Vacant codes grouping and fast OVSF code assignment scheme for WCDMA networks

Vipin Balyan · Davinder S. Saini

Published online: 8 June 2011

Abstract Channelization codes used in WCDMA are Orthogonal Variable Spreading Factor (OVSF) codes. These codes suffer from code blocking limitation. Many designs are proposed to avoid this limitation but most of them do not consider number of codes searched, which affects call establishment delay prior to handling a call. We propose a fast OVSF code assignment design which aims to reduce number of codes searched with optimal/suboptimal code blocking. The code assignment scheme aims to use those vacant codes whose parents are already blocked. This leads to occurrence of more vacant codes in groups, which ultimately leads to less code blocking for higher rate calls. The number of codes searched increases linearly in our design compare to most of other novel proposed single code methods like crowded first assignment, where it increases exponentially with increase in user rates. Also the calculation of vacant codes at one layer will be sufficient to identify the vacant code adjacency for all the layers which reduces complexity. Simulation results are presented to verify the superiority of the design.

Keywords OVSF codes · Code blocking · Code searches · Code assignment · Crowded first assignment

V. Balyan · D.S. Saini (✉)
Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Waknaghat, Himachal Pradesh Distt. Solan, 173215, India
e-mail: davinder.saini@juit.ac.in

V. Balyan
e-mail: vipin.balyan@juit.ac.in

1 Introduction

Wideband code division multiple access (WCDMA) [1] uses OVSF codes to handle multimedia rates along with traditional voice rate services. The different type of services has different quality of service (QoS) requirements. While real time calls like speech and video conferencing require less call establishment delay and jitter, the non real time calls like video downloads and email transfers require higher transmission bandwidth and reliability. In OVSF based networks the time required in optimum vacant code selection has a significant impact on call establishment delay and jitter. Therefore, it is required to minimize the code selection time. The proposed design aims to minimize this delay while preserving the other performance benefits in WCDMA networks. The spreading factor (SF) of OVSF codes differ according to the application. The higher is the rate requirement of user, lesser is the SF. However, their product is always 3.84 Mcps. The maximum allowable value of SF is, $SF_{max} = 2^{L-1}$, if the OVSF system serve users of L distinct classes.

OVSF codes are generated from the binary code given in [2]. WCDMA uses 8 layer OVSF code tree. All codes in each layer are mutually orthogonal. The essential property of OVSF code tree is that once a code is assigned to a call, neither its children nor ancestors are eligible for code assignment of future calls. This is because two different codes can be used if they are orthogonal to each other and a code is not orthogonal to its ancestors or children. This limits the number of codes for new calls. It may happen that code tree is unable to handle new call despite of having sufficient capacity. This is called *code blocking* and is the major limitation of OVSF based 3G wireless networks. The code blocking can be explained in Fig. 1, where the maximum capacity of the system/tree is $32R$ (R is 7.5 kbps for WCDMA downlink). There are six different layers corresponding to six user

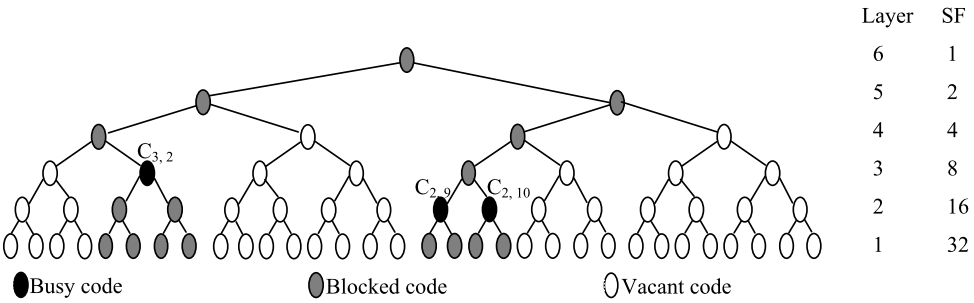

Fig. 1 A six layer OVSF code tree with maximum capacity of $32R$

classes. The SF in layer l, $1 \leq l \leq 6$ is 2^{6-l}. A particular code in layer l is denoted by C_{l,n_l}, $1 \leq n_l \leq 2^{6-l}$. The used capacity of the code tree due to all ongoing calls is $8R$ (sum of rates $4R$, $2R$, $2R$ due to used codes $C_{3,2}$, $C_{2,9}$, $C_{2,10}$ respectively). The remaining capacity of the system which can be utilized by future calls is $(32R-8R) = 24R$. If a new call with rate $16R$ arrives, the system does not support it because there is no vacant code with rate capacity $16R$ as both codes are blocked due to some of its busy children. This leads to code blocking.

Many schemes are proposed in the literature to reduce code blocking. Broadly speaking, there are two types of schemes namely single code and multi code schemes. Crowded first assignment (CFA) [3], leftmost code assignment (LCA) [3], fixed set partitioning (FSP) [4] and recursive fewer codes blocked (RFCB) scheme [5] are few single code assignment schemes. In CFA, the code assignment is carried out to serve higher rate calls better in future. It has two categories namely crowded first code (based upon number of busy children) and crowded first capacity (based upon children used capacity). In LCA, the code assignment is carried out from left side of the OVSF code tree. In the FSP, the code tree is divided into a number of sub trees according to the number of input traffic classes and their distribution. The RFCB design works on the top of CFA and the optimum code is the one which makes least number of higher rates codes blocked. The RFCB design resolves a tie by searching the best candidate recursively up to layer L. The adaptive code assignment (ADA) [6] scheme divide the tree into small portions according to arrival distribution, which reduces the number of codes searched for new calls, as the different rate calls fall in different portions. The NCPH design [7] assigns precedence number to the adjacent vacant codes of a newly assigned code. If there are multiple vacant codes, a new call is handled adjacent code with highest precedence number. The dynamic code assignment (DCA) scheme in [8] handles new call using code reassignments. This is the best single code scheme to reduce code blocking but the cost and complexity in reassignments is too high which limits its usage for low to medium traffic conditions. The traditional DCA is further improved by capacity and class partitioning [9] methods which reduce complexity and increase scalability. The number of code searched [10, 11] has direct impact on delay or speed of the code assignment and can have significant impact on delay prone services like speech transmission and video conferencing etc. The multi code multi rate assignment (MMCA) [12] scheme takes into consideration mobile devices with different multi code transmission capabilities and different QoS parameters. The multi code design scheme with code sharing [13] is suggested to reduce wastage capacity or code blocking. It combines scattered capacity (children codes of assigned codes) of already assigned codes to reduce code blocking problem. The multi code design [14] formulates the optimum number of codes/rakes required to handle new call. The multi code design given in [15] defines internal and external fragmentation limitations of the OVSF code tree and suggests various ways to reduce them. The internal fragmentation is due to quantized rate handling capability of the code tree and external fragmentation is due to scattered busy (or vacant) codes in the code tree. The integration scheme [16] provides the optimal code selection with the constraints of amount of allocated codes and maximal resource wastage ratio. It gives superior performance using two and three codes in a multi code with crowded group first strategy. The code utilization and blocking benefits are significant for resource wastage ratio of 40%. In the time based code allocation scheme [17], the impact of remaining time of calls on performances of OVSF code assignment and reassignment is investigated. The code assignment finds the optimum vacant code whose busy sibling has maximum remaining time. The quality based code assignment [18] analyzes an interference limited system. Three code assignment and reassignment designs are proposed. While most of the designs use the vacant code whose parents have highest number of busy codes, the quality based methods use the parents with least busy capacity. This is done to handle variable data rates better. The dynamic priority assignment algorithm in [19] provides QoS differentiation among different traffic classes based on delay sensitivity. It gives the deterministic delay bound for various classes under the constraint of reliable transmission. Additionally, the algorithm provides guaranteed rate for specified scheduling period.

Table 1 Relationship between layer 'l' and maximum adjacency

Layer (l)	1	2	$\cdots$	l	$\cdots$	$L-1$	L
Maximum adjacency	$2^{L-1}-1$	$2^{L-2}-1$	$\cdots$	$2^{L-l}-1$	$\cdots$	1	0

The single code design described in this paper reduces code blocking with additional benefit of reduction in number of code searched prior to assignment of new call. For a particular higher layer code, all the children codes are either vacant or occupied. This provides the occurrence of vacant codes in groups, which can be easily located and the number of code searches are reduced. This further reduces decision time (or call establishment time), complexity and cost. Although, the ADA [6] design also provides the reduction in code searches, but the design discussed in this paper provides the additional benefit of reduction in code blocking.

The remainder of the paper is organized as follows. Section 2 gives the description of proposed design along with flowchart and examples. Simulation results are given in Sect. 3. The paper is concluded in Sect. 4.

2 Proposed design

Consider an L ($L = 8$ in WCDMA) layer OVSF code tree. If a new call with rate $2^{l-1}R$, $1 \le l \le L$ arrives, the proposed scheme (called as adjacent vacant code (AVC) scheme) lists all the vacant codes in layer l. Let layer l has N_l, $N_l \le 2^{L-l}$ vacant codes. The i^{th} vacant code in layer l is represented by C_{l,n_l^i}, $1 \le n_l^i \le 2^{L-l}$ and $1 \le i \le N_l$. For a code C_{l,n_l^i}, define *adjacency* (A_{l,n_l^i}) as the number of vacant codes adjacent to C_{l,n_l^i} whose parents in all layers are same as the parents of C_{l,n_l^i}. The adjacency of all these vacant codes is the same and it is sufficient to check the status of one of these codes. The design checks only left code from this group. For vacant code C_{l,n_l^i} whose left hand code is busy/blocked or its left hand vacant code has at least one same vacant parent as the parent of C_{l,n_l^i}, the AVC scheme checks the parents of code C_{l,n_l^i} in layers $l+1$ to L till the blocked parent in layer l' (where $l' > l$ and $l' = \min[l+1, L] | C_{l', \lfloor n_l^i / 2^{l'-l} \rfloor}$ is blocked) appears. The identifier $\lfloor x \rfloor$ represents smallest integer less than or equal to x. The adjacency of code C_{l,n_l^i} becomes $2^{l'-l-1}-1$. The vacant codes C_{l,n_l^i} to $C_{l,n_l^i+A_{l,n_l^i}}$ need not to be checked as all these have same value of adjacency. The values of A_{l,n_l^i} for code C_{l,n_l^i} is equal to $2^{l'-l-1}-1$ although there may be more than $2^{l'-l-1}-1$ consecutive vacant codes. The maximum possible value of A_{l,n_l^i} depends upon layer number l.

Table 2 Relationship between adjacency of a vacant code in layer l and its parents in layer l', $l' > l$

Layer number	Vacant codes	Adjacency
l	C_{l,n_l^i} to $C_{l,n_l^i+A_{l,n_l^i}}$	A_{l,n_l^i}
$l+1$	$C_{l+1,\lfloor n_l^i/2 \rfloor}$ to $C_{l+1,\lfloor n_l^i/2 \rfloor+\lfloor A_{l,n_l^i}/2 \rfloor}$	$\lfloor A_{l,n_l^i}/2 \rfloor$
$\cdots$	$\cdots$	$\cdots$
$l+\log_2[A_{l,n_l^i}+1]$	$C_{l+m,\lfloor n_l^i/2 \rfloor}$ to $C_{l+m,\lfloor n_l^i/2^m \rfloor+\lfloor A_{l,n_l^i}/2^m \rfloor}$	$\lfloor A_{l,n_l^i}/2^m \rfloor$

where $m = \log_2[A_{l,n_l^i}+1]$

The relationship between layer number l, vacant codes location and maximum adjacency is given in Table 1. The lesser is the value of adjacency A_{l,n_l^i} for a code C_{l,n_l^i}, lesser is the number of adjacent vacant codes for code C_{l,n_l^i}. For a specified adjacency A_{l,n_l^i} (say), the codes from layer l to $(l+\log_2[A_{l,n_l^i}+1])$ are vacant and can be used for new calls as given in Table 2.

Hence, if a vacant code C_{l,n_l^i} has higher value of adjacency, the bigger code portion around C_{l,n_l^i} is free, which leads to more higher rate parents free to handle new call(s). The design selects the vacant code whose adjacency is least. If tie occurs for two or more vacant codes for adjacency, the optimum selection can be done using following two ways.

- Pick any code randomly.
- Use a second level criterion like elapsed time, crowded first capacity, crowded first space, etc.

It assigns new call to most crowded portion of the tree. One of the significant benefits of the AVC design is to reduce the vacant code searches at the arrival of new call which makes the code assignment faster than single code designs like crowded first code and crowded first space schemes. Adjacency leads to less code searches for a vacant code C_{l,n_l^i} in two ways.

(a) Let adjacency is k for vacant code C_{l,n_l^i}, then for all codes between C_{l,n_l^i} and C_{l,n_l^i+k} has same adjacency k and only one code need to be searched. Our next vacant code search will start from C_{l,n_l^i+k+1}.

(b) For finding the adjacency of a code C_{l,n_l^i}, we go to its parent codes and check whether they are vacant or not. If we go up l'', $1 \le l'' \le L-l$ steps above then number of code

Table 3 Finding number of code searches in Fig. 1 at the arrival of $2R$ rate call

Code # in layer 2	Whether status checked?	Number of skips	Number of codes searched
$C_{2,1}$	Y	1	3
$C_{2,2}$	N	NA	NA
$C_{2,3}$	Y	0	1
$C_{2,4}$	Y	0	1
$C_{2,5}$	Y	3	4
$C_{2,6}$	N	NA	NA
$C_{2,7}$	N	NA	NA
$C_{2,8}$	N	NA	NA
$C_{2,9}$	Y	0	1
$C_{2,10}$	Y	0	1
$C_{2,11}$	Y	1	3
$C_{2,12}$	N	NA	NA
$C_{2,13}$	Y	3	4
$C_{2,14}$	N	NA	NA
$C_{2,15}$	N	NA	NA
$C_{2,16}$	N	NA	NA

NA: Not applicable, Y: Yes, N: No, the code search is skipped as one of its adjacent vacant codes is already considered for status check

searches will be $l'' = L - l$, (one for each layer) and adjacency or codes vacant preceding C_{l,n_l^i} checked is $2^{l''} - 1$. Checking of a parent code leads to search of two children status on code tree and while doing so, code searches will increase linearly (instead of exponentially). For a new $2R$ call arrival, the number of code searched in the AVC scheme is illustrated in Table 3 (corresponding to Fig. 1). As discussed earlier, all the vacant codes need not to be checked. For example, the 1st code $C_{2,1}$ is vacant and status of its parents are checked till the 1st blocked parent appears. The 1st blocked parent is $C_{4,1}$. The parent checking procedure stops and the adjacency of $C_{2,1}$ come out to be 1. Due to the adjacency value 1, code $C_{2,2}$ is not the candidate for status check as it is already considered in parent check procedure of $C_{2,1}$. The next vacant code whose parent should be checked is $C_{2,5}$. The procedure is repeated for the complete tree as given in Table 3. In Table 3, column 2 represents whether status of code in layer 2 is checked or not, column 3 represents number of skips before next vacant code to be searched and column 4 represents number of codes searched for finding adjacency. The flow chart of algorithm is given in Fig. 2. The general outline of the algorithm is given as follows.

1. Input parameters like arrival rate, call duration, code tree size (L) and number of user classes (say L') are entered.
2. Generate new call with rate $2^{l-1}R$.
3. For every new call capacity check is performed. If capacity is available go to step 4 otherwise the call is rejected.
4. (a) If at least one vacant code is available, list all the vacant codes (say N_l) and the subset z of these N_l vacant codes whose parent(s) status must be checked.
   (b) Go to the first vacant code from the left.
   (c) Find the adjacency and skip the number of vacant codes equal to adjacency.
   (d) If all z vacant codes (and their parents) are checked, find the code with the least value of adjacency and assign call to this optimum code. Otherwise go to the next vacant code and return to step 4(c).
   (e) Go to step 2.

Assuming layer l has N_l, $N_l \leq 2^{L-l}$ vacant codes and each vacant code in layer l can be denoted by C_{l,n_l^i}, $1 \leq n_l^i \leq 2^{L-l}$ and $1 \leq i \leq N_l$, the number of code searched in the AVC scheme can be compared with the codes searched in *CFA* scheme.

2.1 Code searches for AVC scheme

For a new call arrival with rate $2^{l-1}R$, let C_{l,n_l^1} is the first vacant code. The next vacant code whose status need to be checked is at the location $n_l^1 + 2^{l_1-1}$(l_1 is the steps/layers above l where first blocked parent code is found for a vacant code C_{l,n_l^1}). Let k^{th} vacant code whose status has to be checked (which may be different from C_{l,n_l^k}) is denoted by $C_{l,n_l^{x_k}}$, $x_k \in [1, N_l]$, $x_k \geq k$. Consider n_l^1 and l_1 is $n_l^{x_1}$ and l_{x_1} respectively. If t_{x_1} denotes number of busy or blocked

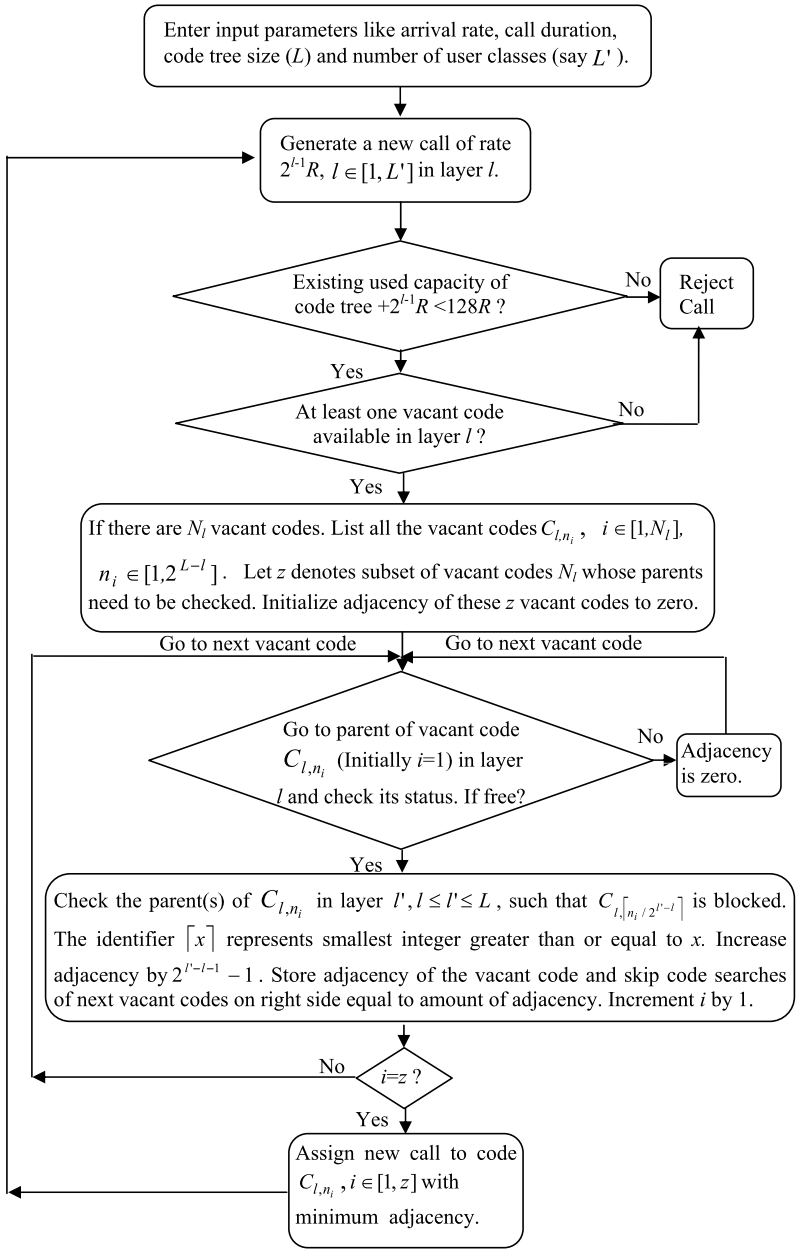

Fig. 2 Flowchart of proposed AVC scheme

codes checked, next vacant code $C_{l,n_l^{x_2}}$ searched will be at location $n_l^{x_1} + 2^{l_{x_1}-1} + t_{x_1}$. This result can be generalized to find location of k^{th} vacant code $C_{l,n_l^{x_k}}$ and is given by $n_l^{x_{k-1}} + 2^{l_{x_{k-1}}-1} + t_{x_{k-1}}$. The procedure is repeated z times, where z denotes subset of N_l vacant codes which need to be checked for adjacency and the last vacant code checked is $C_{l,n_l^{x_z}}$ as given in Table 4.

For the first code C_{l,n_l^1} or $C_{l,n_l^{x_1}}$ number of code searches is $l_{x_1} + 1$. For k^{th} vacant code $C_{l,n_l^{x_k}}$, number of code searches is $l_{x_k} + 1 + t_{x_{k-1}}$. The procedure is repeated till the last code $C_{l,n_l^{x_z}}$. Number of code searched for a vacant code and total number of code searched till that vacant code is also shown in Table 4.

Total number of code searched in AVC design is given below:

$$N_{AVC} = n_l^{x_1} - 1 + l_{x_1} + 1 + l_{x_2} + 1 + t_{x_1} + l_{x_3} + 1 + t_{x_2} + \cdots + l_{x_z} + 1 + t_{x_{z-1}} + t_{x_z} \quad (1)$$

$$= n_l^{x_1} - 1 + \sum_{i=1}^{z} (l_{x_i} + 1 + t_{x_i}) \quad (2)$$

$$= n_l^{x_1} - 1 + z + \sum_{i=1}^{z} (l_{x_i} + t_{x_i}) \quad (3)$$

Table 4 Total number of codes searched in layer 'l' at the arrival of $2^{l-1}R$ call

Vacant code	Adjacency (A_{l,n_l^k})	Number of skips	Next location to be searched	Number of code searched for $C_{l,n_l^{x_k}}$	Total number of code searched (say $NA_{l,n_l^{x_k}}$)
C_{l,n_l^1} or $C_{l,n_l^{x_1}}$	$2^{l_{x_1}-1}-1$	$2^{l_{x_1}-1}-1$	$n_l^{x_1}+2^{l_{x_1}-1}+t_{x_1}$	$l_{x_1}+1$	$NA_{l,n_l^{x_1}}=n_l^{x_1}-1+l_{x_1}+1=n_l^{x_1}+l_{x_1}$
$C_{l,n_l^{x_2}}$	$2^{l_{x_2}-1}-1$	$2^{l_{x_2}-1}-1$	$n_l^{x_2}+2^{l_{x_2}-1}+t_{x_2}$	$l_{x_2}+1+t_{x_1}$	$NA_{l,n_l^{x_2}}=NA_{l,n_l^{x_1}}+l_{x_2}+1+t_{x_1}$
...	...	...	...	...	...
$C_{l,n_l^{x_k}}$	$2^{l_{x_k}-1}-1$	$2^{l_{x_k}-1}-1$	$n_l^{x_1}+2^{l_{x_k}-1}+t_{x_k}$	$l_{x_k}+1+t_{x_{k-1}}$	$NA_{l,n_l^{x_k}}=NA_{l,n_l^{x_{k-1}}}+l_{x_k}+1+t_{x_{k-1}}$
...	...	...	...	...	...
$C_{l,n_l^{x_z}}$	$2^{l_{x_z}-1}-1$	$2^{l_{x_z}-1}-1$	$n_l^{x_z}+2^{l_{x_z}-1}$	$l_{x_z}+1+t_{x_z}$	$NA_{l,n_l^{x_z}}=NA_{l,n_l^{x_{z-1}}}+l_{x_z}+1+t_{x_{z-1}}$

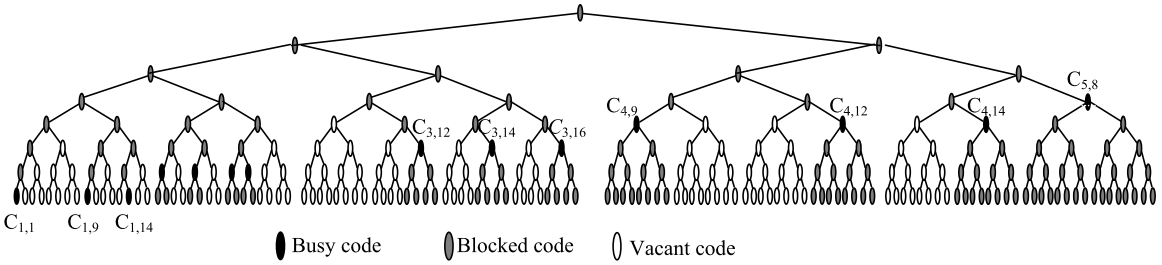

Fig. 3 Illustration of code searches in AVC and CFA schemes

2.2 Code searches for CFA scheme

Considering the definition of $C_{l,n_l^{x_1}}$, the number of codes searched for $C_{l,n_l^{x_1}}$ using CFA [3] scheme is $NC_{l,n_l^{x_1}} = l_{x_1}+2^{l+l_{x_1}-1}-1$. For i^{th} vacant code, code searched can be written as $NC_{l,n_l^{x_i}} = l_{x_i}+2^{l+l_{x_i}-1}-1$. For all N_l, $N_l \le 2^{L-l}$ vacant codes, the total number of codes searched becomes

$$N_{CFA} = 2^{L-l}+l_{x_1}+(2^{l+l_{x_1}-1}-1)+l_{x_2}+(2^{l+l_{x_2}-1}-1) + \cdots + l_{x_{N_l}}+(2^{l+l_{x_{N_l}}-1}-1) \quad (4)$$

$$= 2^{L-l}+\sum_{i=1}^{N_l}(l_{x_i}+2^{l+l_{x_i}-1}-1) \quad (5)$$

$$= 2^{L-l}-N_l+\sum_{i=1}^{N_l}(l_{x_i}+2^{l+l_{x_i}-1}) \quad (6)$$

Our proposed scheme always needs less number of code searches as compare to CFA [3], which is given in the Appendix.

In comparison to CFA, where code search is increasing exponentially with every parent search, the AVC design leads to a linear code searches which decreases the number of code searched as we go in upper layers from l to $L-l$. For illustration, consider example in Fig. 3. Consider a new call arrival with rate $2R$. CFA scheme search most crowded portion of the tree. For that it searches the parent codes and their children. It selects a vacant code whose parent/parents have maximum number of busy codes. The comparison of CFA and AVC scheme is illustrated in Table 5 in context with Fig. 3. The adjacencies of all the vacant codes in layer 2 (corresponding to $2R$ arrival) is listed in column 4. The higher value of adjacency (number of skips) corresponds to large free area. The code with the minimum adjacency is the optimum code and hence one of the codes with adjacency 0 is used for $2R$ call.

Periodic code searches can be avoided to a great extent if the current code status remains valid for next k calls. Adjacencies of codes of layer l provide information of adjacency of children codes (layers below l) and parent codes (layers above l). Let the vacant code C_{l,n_l^i} has adjacency P_i. Its children codes have adjacency given by
Layer $l-1$, codes $C_{l-1,2n_l^i}$ to $C_{l-1,2n_l^i+1}$ has adjacency

$$A_1 = 2P_i+1 \quad (7)$$

Layer $l-2$, codes $C_{l-2,4n_l^i}$ to $C_{l-2,4n_l^i+2^2-1}$ has adjacency $A_2 = 2A_1+1$. Similarly, for vacant codes in l_1 steps below layer l i.e. layer $l-l_1$ codes $C_{l-l_1,2^{l_1}n_l^i}$ to $C_{l-l_1,2^{l_1}n_l^i+2^{l_1}-1}$ has adjacency

$$A_{l_1} = 2A_{l_1-1}+1 \quad (8)$$

For vacant code C_{l,n_l^i}, parent codes have adjacency given by
Layer $l+1$, code $C_{l+1,\lfloor n_l^i/2 \rfloor}$ has adjacency

$$P_1 = \lfloor P_i/2 \rfloor \quad (9)$$

Table 5 Comparison of number of code searches in AVC and CFA scheme for tree in Fig. 2

Status with Adjacency	Codes #	Code searches in AVC scheme	Number of skips in AVC AVC scheme	Code search in CFA scheme
Busy/Blocked Adjacency:0	$C_{2,1}$, $C_{2,5}$, $C_{2,7}$, $C_{2,9}$, $C_{2,11}$, $C_{2,13}$, $C_{2,14}$, $C_{2,23}$, $C_{2,24}$, $C_{2,27}$, $C_{2,28}$, $C_{2,31}$ to $C_{2,36}$, $C_{2,45}$ to $C_{2,48}$, $C_{2,53}$ to $C_{2,64}$	1 for each code = 31	0	1each = 31
Vacant Adjacency:0	$C_{2,2}$, $C_{2,6}$, $C_{2,8}$, $C_{2,10}$, $C_{2,12}$	2 each = 10	0	5each = 5 × 5 = 25
Vacant Adjacency:1	$C_{2,3}$, $C_{2,4}$,	3 (for $C_{2,3}$ only)	1	10each = 20
	$C_{2,15}$, $C_{2,16}$	3 (for $C_{2,15}$ only)	1	10each = 20
	$C_{2,21}$, $C_{2,22}$	3 (for $C_{2,21}$ only)	1	10each = 20
	$C_{2,25}$, $C_{2,26}$	3 (for $C_{2,25}$ only)	1	10each = 20
	$C_{2,29}$, $C_{2,30}$	3 (for $C_{2,29}$ only)	1	10each = 20
Vacant Adjacency:3	$C_{2,17}$ to $C_{2,20}$	4 (for $C_{2,17}$ only)	3	19each = 76
	$C_{2,37}$ to $C_{2,40}$	4 (for $C_{2,37}$ only)	3	19each = 76
	$C_{2,41}$ to $C_{2,44}$	4 (for $C_{2,41}$ only)	3	19each = 76
	$C_{2,49}$,$C_{2,52}$	4 (for $C_{2,49}$ only)	3	19each = 76
	Total number of searches	72		460

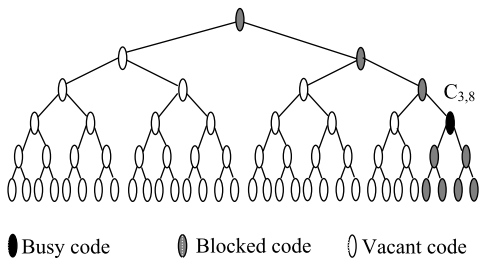

Fig. 4 Illustration of parent and children code adjacency

Layer $l+2$, code $C_{l+1,\lfloor n_l^i/2^2 \rfloor}$ has adjacency $P_2 = \lfloor P_i/2^2 \rfloor$. In a similar way, for a vacant code in l_2 steps above layer l i.e. layer $l+l_2$ code has adjacency

$$P_{l_2} = \lfloor P_i/2^{l_2} \rfloor \tag{10}$$

For L^{th} layer adjacency will be $P_L = \lfloor P_i/2^{L-l} \rfloor$.

For illustration consider the Fig. 4. For a call of rate $4R$, vacant codes are $C_{3,1}$ to $C_{3,7}$. Adjacency for these codes is 3 from $C_{3,1}$ to $C_{3,4}$, 1 for $C_{3,5}$, $C_{3,6}$ and 0 for $C_{3,7}$ respectively. The new call will be assigned to $C_{3,7}$ as adjacency is minimum for $C_{3,7}$. As we know the adjacency of code in layer 3, apparently we know adjacency of their parent and children codes also. From Fig. 4, it is clear that adjacency of $C_{2,1}$ to $C_{2,8}$ has adjacency 7 i.e. $(2\times3+1)$. $C_{2,9}$ to $C_{2,12}$ has adjacency 3 i.e. $(2\times1+1)$. In a similar way, we can find adjacency of parent codes of a vacant code for e.g. $C_{2,1}$ has adjacency 7, its parent code $C_{3,1}$ has adjacency $\lfloor 7/2 \rfloor = 3$.

3 Simulation results

3.1 Simulation parameters

- Arrival rate 'λ' is Poisson distributed with mean value 0–4 calls/min.
- Call duration '$1/\mu$' (μ is the service rate) is exponentially distributed with mean value of 3 minutes.
- The maximum capacity of the code tree is $128R$ (R is 7.5 kbps).
- There are 5 classes of users with rates R, $2R$, $4R$, $8R$, $16R$.
- Simulation is done for 5000 users and result is average of 10 simulations.

3.2 Results

We considered event driven simulation where each code in a layer is a server. The number of servers available for layer l is 2^{L-l} which is equal to number of codes in layer l. Let $\lambda_i, i \in [1,5]$ is the arrival rate and $\mu_i, i \in [1,5]$ is service rate for i^{th} class users. Define $\rho_i = \lambda_{i/\mu_i}$ as traffic load of the i^{th} class users. Also, for 5 class system the average arrival rate and average traffic load is $\lambda = \sum_{i=1}^{5} \lambda_i$ and $\rho = \sum_{i=1}^{5} (\lambda_i/\mu_i)$ respectively. In our simulation we consider call duration of all the calls equal i.e. $1/\mu = 1/\mu_i$. Therefore the average traffic load is $\rho = (1/\mu) \times \sum_{i=1}^{5} \lambda_i = \lambda/\mu$. If we define $[P_1, P_2, P_3, P_4, P_5]$ as probability distribution matrix, where $P_i, i \in [1,5]$ is the capacity portion used by

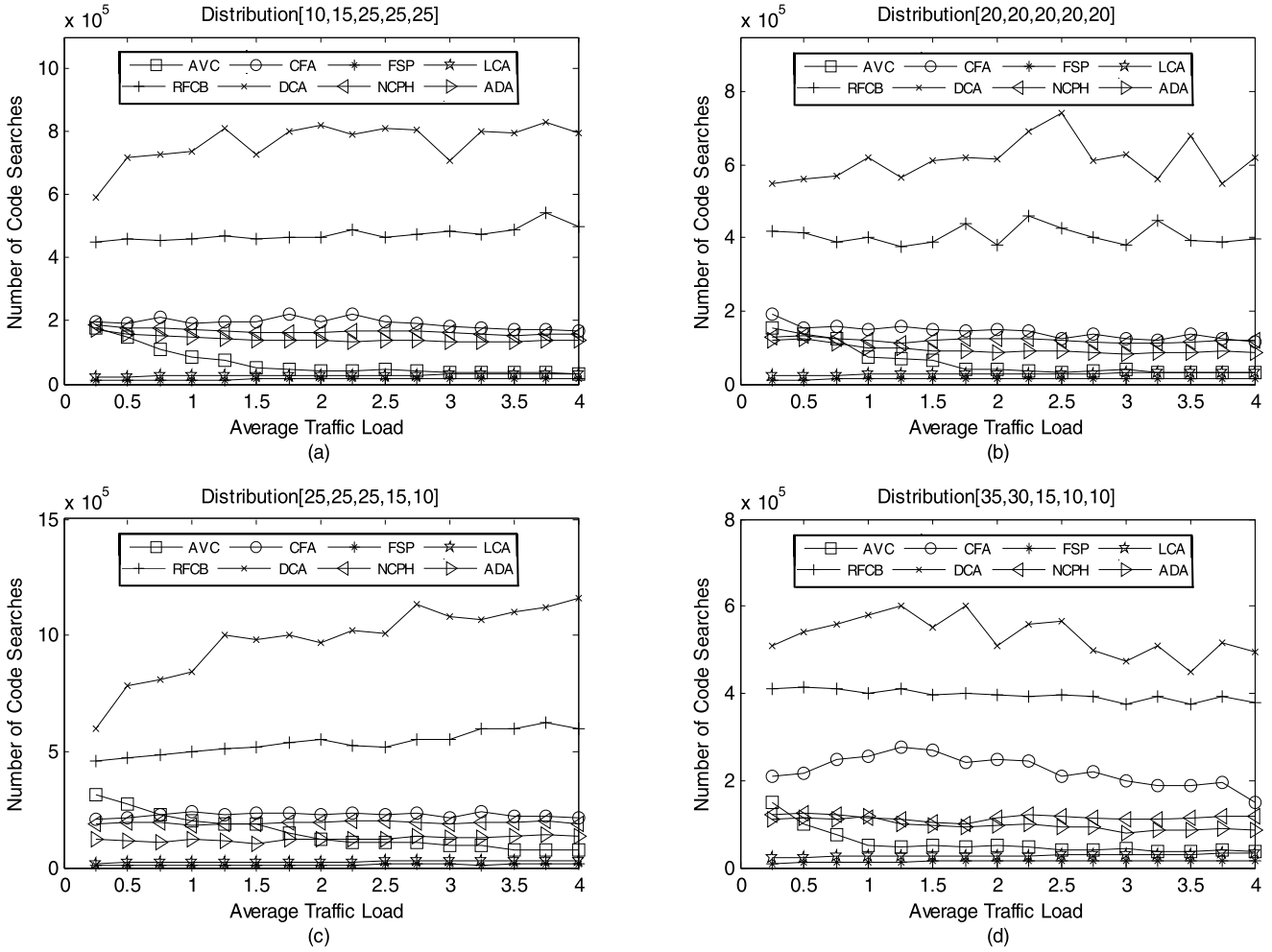

Fig. 5 Comparison of number code searches for distribution (**a**) [10, 15, 25, 25, 25], (**b**) [20, 20, 20, 20, 20], (**c**) [25, 25, 25, 15, 10], (**d**) [35, 30, 15, 10, 10]

the i^{th} class users, four traffic distributions are used for performance results given by

- [10, 15, 25, 25, 25], high rates dominating scenario.
- [20, 20, 20, 20, 20], uniform distribution scenario.
- [25, 25, 25, 15, 10], low rates dominating scenario I.
- [35, 30, 15, 10, 10], low rates dominating scenario II.

Two performance metrics namely number of code searches and blocking probability of the proposed AVC design is compared with fixed set partitioning (FSP) [4], crowded first assignment (CFA) [3], left code assignment (LCA) [3], recursive fewer code blocking (RFCB) [5], adaptive code assignment [6], next code precedence high [7] and dynamic code algorithm (DCA) [8] schemes discussed earlier.

3.3 Code searches

The comparison of the code searches is shown in Fig. 5 for four arrival rate distributions. If N_i denotes the number of codes searches required for i^{th} class, the total number of codes searched is given by

$$N = \sum_{i=1}^{5} N_i \tag{11}$$

where N_i is the total number of codes searched for i^{th} class of user. Results shows that AVC design leads to less number of code searches compared to CFA, RFCB, ADA, NCPH and DCA schemes. Although, CFA and DCA are two most used single code assignment designs, the AVC design requires less code searches compared to both of them which can be very beneficial especially for real time calls. On the other hand, the number of code searches required by AVC design are comparable to LCA and more than FSP schemes but the large code blocking probability in LCA and FSP schemes (as discussed in the next subsection) make these schemes less attractive. The reduction

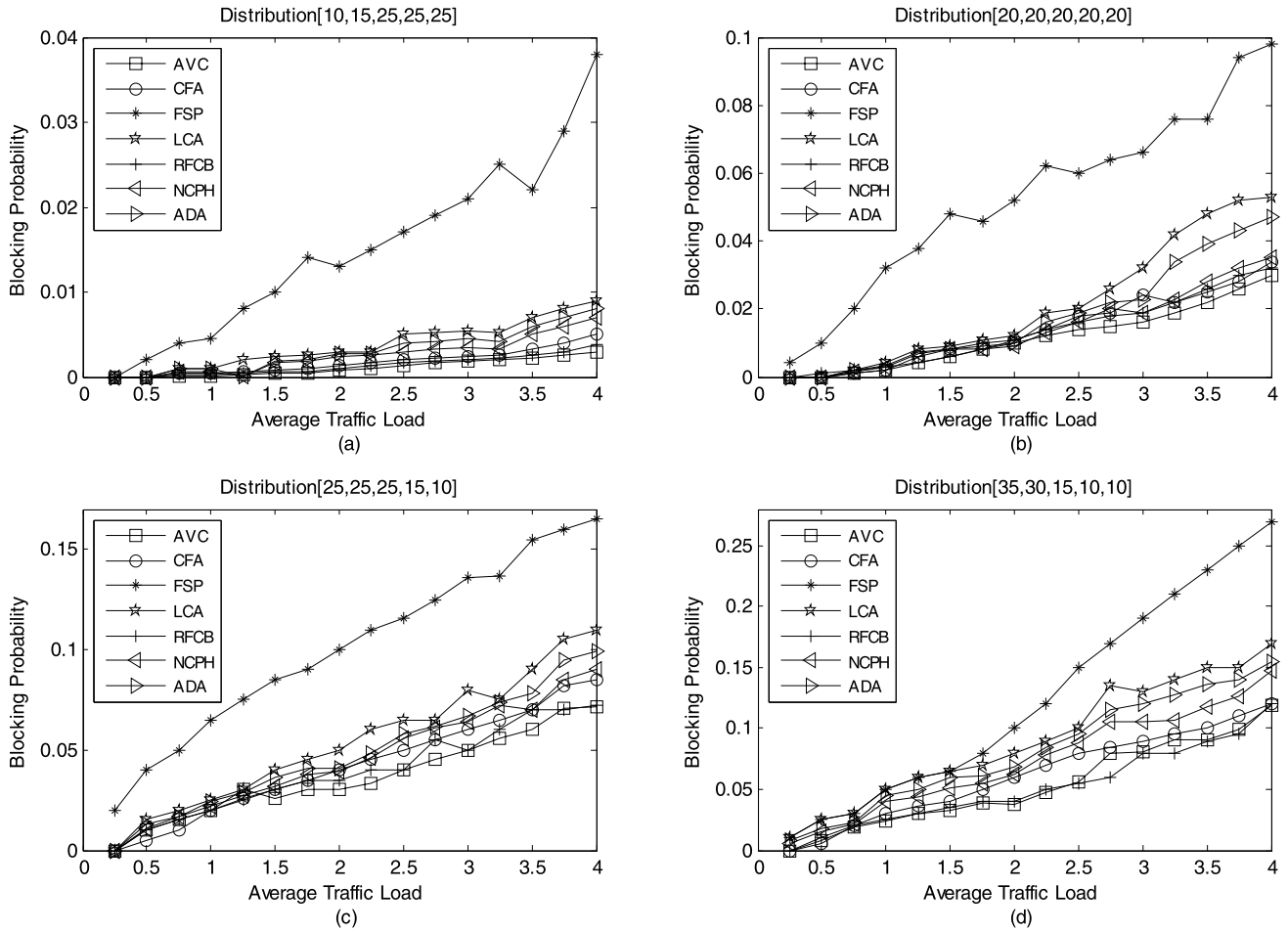

Fig. 6 Comparison of code blocking probability for distribution (**a**) [10, 15, 25, 25, 25], (**b**) [20, 20, 20, 20, 20], (**c**) [25, 25, 25, 15, 10], (**d**) [35, 30, 15, 10, 10]

in code searches in the AVC design is further reduced for distributions favoring higher rates as depicted in Figs. 5(c) and 5(d).

3.4 Code blocking

As mentioned earlier, the code blocking is the major limiting factor in OVSF based networks. The average code blocking for a 5 class system is defined as

$$P_B = \sum_{i=1}^{5} (\lambda_i P_{B_i} / \lambda) \tag{12}$$

where P_{B_i} is the code blocking of i^{th} class and is given by

$$P_{B_i} = \frac{\rho_i^{G_i} / G_i!}{\sum_{n=1}^{G_i} \rho_i^n / n!} \tag{13}$$

where $\rho_i = \lambda_i / \mu_i$ is the traffic load for i^{th} class.

The code blocking comparison is given in Fig. 6 for all 4 distributions. The results show that code blocking performance in the proposed AVC design is comparable to CFA, RFCB and is superior to FSP, LCA, ADA and NCPH schemes. The DCA scheme gives almost zero code blocking (hence not plotted) at the expense of higher reassignments which lead to increased complexity. Though the blocking probability is comparable to CFA and RFCB but reduction in code searches (as discussed in the previous subsection) makes the proposed design faster, less complex and cheaper.

From above results it is clear that the AVC design gives better aggregate results in terms of number of code searches and blocking probability compared to other novel designs. As mentioned earlier, the number of code searches is less due to linear increase in code checking with respect to layer number. In all other designs, this increase is exponential. Therefore, the AVC design can be useful especially for real time calls.

4 Conclusion

3G and beyond wireless networks are designed to handle multimedia rates better. Call processing delay and jitter are the significant QoS parameters for most of the real time calls. The paper proposed a fast, simple code assignment design. The design can be even better when the system favors real time calls. The number of codes searched in the proposed design is significantly lesser than popular crowded first scheme. The online calculation of codes searched reduces the cost, complexity and buffer size at the transmitter. The code blocking is also comparable or superior to existing single code designs. Work can be done to extend the single code description towards multi code assignment design. Also, the proposed design can be combined with existing FSP to increase speed further.

Appendix

The proposed scheme is better if

Number of code searches for AVC $\leq$ Number of code searches for CFA, i.e.

$$z+\sum_{i=1}^{z} l_{x_i}+\sum_{i=1}^{z+1} t_{x_i}-1 \leq 2^{L-l}-N_l+\sum_{i=1}^{N_l}(l_{x_i}+2^{l+l_{x_i}-1}) \tag{A.1}$$

$$z+\sum_{i=1}^{z} l_{x_i}+\sum_{i=1}^{z+1} t_{x_i}-1 \leq 2^{L-l}-N_l+\sum_{i=1}^{N_l} l_{x_i}+\sum_{i=1}^{N_l} 2^{l+l_{x_i}-1} \tag{A.2}$$

We can represent $\sum_{i=1}^{N_l} l_{x_i}$ as

$$\sum_{i=1}^{N_l} l_{x_i}=\sum_{i=1}^{z} l_{x_i}+\sum_{i=z+1}^{N_l} l_{x_i} \tag{A.3}$$

using (A.3) in (A.2)

$$z+\sum_{i=1}^{z} l_{x_i}+\sum_{i=1}^{z+1} t_{x_i}-1 \leq 2^{L-l}-N_l+\sum_{i=1}^{z} l_{x_i}+\sum_{i=z+1}^{N_l} l_{x_i}+\sum_{i=1}^{N_l} 2^{l+l_{x_i}-1} \tag{A.4}$$

$$z+\sum_{i=1}^{z+1} t_{x_i}-1 \leq 2^{L-l}-N_l+\sum_{i=z+1}^{N_l} l_{x_i}+\sum_{i=1}^{N_l} 2^{l+l_{x_i}-1} \tag{A.5}$$

$$z+\sum_{i=1}^{z+1} t_{x_i}+N_l-1 \leq 2^{L-l}+\sum_{i=z+1}^{N_l} l_{x_i}+\sum_{i=1}^{N_l} 2^{l+l_{x_i}-1} \tag{A.6}$$

It can be easily found out that

$$z \leq \sum_{i=1}^{N_l} 2^{l+l_{x_i}-1} \tag{A.7}$$

where z is number of vacant code searched for adjacency and N_l is total number of vacant codes.

Using (A.7) in (A.6), we get

$$\sum_{i=1}^{z+1} t_{x_i}+N_l-1 \leq 2^{L-l}+\sum_{i=z+1}^{N_l} l_{x_i} \tag{A.8}$$

Also,

$$\sum_{i=1}^{z+1} t_{x_i}+N_l-1 \leq 2^{L-l} \tag{A.9}$$

As 2^{L-l} is total number of codes in layer l, we can write

$$0 \leq \sum_{i=z+1}^{N_l} l_{x_i} \tag{A.10}$$

which is always true.

References

1. Adachi, F., Sawahashi, M., & Suda, H. (1998). Wideband DS-CDMA for Next-Generation Mobile Communications Systems. *IEEE Communications Magazine*, *36*, 56–69.
2. Adachi, F., Sawahashi, M., & Okawa, K. (1997). Tree structured generation of orthogonal spreading codes with different lengths for forward link of DS-CDMA mobile radio. *IEE Electronics Letters*, *33*, 27–28.
3. Tseng, Y. C., Chao, C. M., & Wu, S. L. (2001). Code placement and replacement strategies for wideband CDMA OVSF code tree management. In *Proc. of IEEE GLOBECOM* (Vol. 1, pp. 562–566).
4. Park, J. S., & Lee, D. C. (2003). Enhanced fixed and dynamic code assignment policies for OVSF-CDMA systems. In *Proc. of ICWN*, Las Vegas (pp. 620–625).
5. Rouskas, A. N., & Skoutas, D. N. (2005). Management of channelization codes at the forward link of WCDMA. *IEEE Communications Letters*, *9*(8), 679–681.
6. Saini, D. S., & Bhooshan, S. V. (2006). Adaptive assignment scheme for OVSF codes in WCDMA. In *Proc. of IEEE ICWMC*, Bucharest, July 2006 (pp. 65).
7. Saini, D. S., Hasthir, V., & Sood, M. (2009). Vacant code precedence and call blocking reduction in WCDMA systems. In *Proc. of IEEE IACC*, Patiala, Mar. 2009 (pp. 812–815).
8. Minn, T., & Siu, K. Y. (1998). Dynamic Assignment of orthogonal variable spreading factor codes in W-CDMA. *IEEE Journal on Selected Areas in Communications*, *18*(8), 1429–1440.

9. Park, J. S., Huang, L., & Kuo, C. C. J. (2008). Computational efficient dynamic code assignment schemes with call admission control (DCA-CAC) for OVSF-CDMA systems. *IEEE Transactions on Vehicular Technology*, *57*(1), 286–296.
10. Askari, M., Saadat, R., & Nakhkash, M. (2008). Comparison of various code assignment schemes in wideband CDMA. In *Proc. of IEEE on computer and comm. engineering*, Kuala Lumpur, May 2008 (pp. 956–959).
11. Saini, D. S., Kanwar, S., Parikh, P., & Kumar, U. (2009). Vacant code searching and selection of optimum code assignment scheme in WCDMA wireless networks. In *Proc. of IEEE conf. NGMAST*, Wales, UK, Sept. 2009 (pp. 224–229).
12. Yang, Y., & Yum, T. S. P. (2005). Multicode multirate compact assignment of OVSF codes for Qos differentiated terminals. *IEEE Transactions on Vehicular Technology*, *54*(6).
13. Saini, D. S., & Bhooshan, S. V. (2009). OVSF code sharing and reducing the code wastage capacity in WCDMA. *Wireless Personal Communications*, *48*, 521–529.
14. Saini, D. S., & Upadhyay, M. (2009). Multiple rake combiners and performance improvement in WCDMA systems. *IEEE Transactions on Vehicular Technology*, *58*(7), 3361–3370.
15. Chao, C. M., Tseng, Y. C., & Wang, L. C. (2005). Reducing internal and external fragmentation of OVSF Codes in WCDMA systems with multiple Codes. *IEEE Transactions on Wireless Communications*, *4*, 1516–1526.
16. Chen, M. X. (2008). Efficient integration OVSF code management architecture in UMTS. *Computer Communications*, *31*(9), 3103–3112.
17. Cheng, S. T., & Hsieh, M. T. (2005). Design and analysis of time-based code allocation schemes in W-CDMA Systems. *IEEE Transactions on Mobile Computing*, *4*(6), 604–615.
18. Tsai, Y. R., & Lin, L. C. (2009). Quality-based OVSF code assignment and reassignment strategies for WCDMA systems. *IEEE Transactions on Vehicular Technology*, *58*(2), 1027–1031.
19. Skoutas, D. N., & Rouskas, A. N. (2009). A Scheduling algorithm with dynamic priority assignment for WCDMA systems. *IEEE Transactions on Mobile Computing*, *8*(1), 126–138.

Vipin Balyan received B.E. degree in Electronics & Communication with honors from U.P. Technical University, Lucknow, Uttar Pradesh, India in 2003 and the M.Tech. degree in Electronics & Networking from LaTrobe University, Bundoora, Melbourne, Australia in 2006. He has been with RKGIT, Ghaziabad affiliated to U.P. Technical University, Lucknow, Uttar Pradesh, India as a Lecturer for 2 years. He is currently working as a Senior Lecturer in Jaypee University, Wakhnaghat, India and pursuing his Ph.D in "Efficient Single Code Assignment in OVSF based WCDMA Wireless Networks".

Davinder S. Saini was born in Nalagarh, India in January 1976. He received B.E. degree in electronics and telecommunication engineering from College of Engineering Osmanabad, India in 1998. He received M. Tech. degree in communication systems from Indian Institute of Technology (IIT) Roorkee, India in 2001. He received Ph.D. degree in electronics and communication from Jaypee University of Information Technology Waknaghat, India in 2008.

He is with Jaypee University of Information Technology Waknaghat since June 2002. Currently, working as an Assistant Professor in electronics and communication department. His research areas include Channelization (OVSF) codes and optimization in WCDMA, routing algorithms and security issues in MANETs.