Code Placement and Replacement Schemes for WCDMA Rotated-OVSF Code Tree Management

Yuh-Shyan Chen and Ting-Lung Lin

Department of Computer Science and Information Engineering
National Chung Cheng University
Chiayi 621, Taiwan, R.O.C.
yschen@cs.ccu.edu.tw

Abstract. In this paper, we introduce a new code tree structure, namely an ROVSF (rotated-orthogonal variable spreading factor) code tree. This work addresses both code placement and replacement schemes in a ROVSF code tree system. Some new properties of our developed ROVSF code tree are presented to develop code placement/replacement schemes with less code blocking probability and less reassignment cost. The main contribution of our scheme is to identify *linear-code chains (LCCs)* and *nonlinear-code chains (NCCs)* in the ROVSF code tree. Our code placement/replacement schemes initially attempt to allocate request codes on *linear-code chains*, and then try to allocate then to *nonlinear-code chains*. Finally, the simulation results illustrate that our code placement/replacement results based on the ROVSF code tree can actually improve the code blocking probability and the code reassignment cost.

1 Introduction

Wireless communication technology has been recently and widely investigated to significantly increase subscribers, traffic, and new bandwidth-consuming applications, such as mobile learning, mobile gaming, music down-loading and wireless video streaming. The capacity demand is supplied by provision of new spectrum technology, namely wideband CDMA or WCDMA (Wideband Code Division Multiple Access), in order to create a global standard for real-time multimedia services and support international roaming. In the 3G technical specifications, *orthogonal variable spreading factor* (OVSF) codes are usually selected to be the channelization codes. In 3GPP specifications [4], orthogonal codes (known as channelization codes) are used to preserve the orthogonality between users' physical channels. The spreading factor [1] indicates how many bits of those codes are used in the connection. They are basically Walsh codes of different lengths that are able to preserve orthogonality between channels even when they are operating at different data rates. The OVSF codes are arranged in a tree structure for code allocation, and the OVSF channelization codes are widely used to provide variable data rates for supporting different bandwidth requirements

in WCDMA systems. Much work in the literature [2][3][5][6] has intensively investigated *code placement* and *replacement* schemes with the objective of less call-blocking probability and less reassignment cost in the OVSF code tree.

In this paper, we initially introduce a new code tree structure, namely the ROVSF (rotated-orthogonal variable spreading factor) code tree, whose code capacity is the same as that of the traditional OVSF code tree. This work addresses both code placement and replacement schemes in a ROVSF code tree system, where ROVSF codes are used for up-link and down-link channelization codes. Some new properties of our developed ROVSF code tree are presented to develop code placement and replacement schemes with less code blocking probability and less reassignment cost. The main idea of our scheme is to identify *linear-code chains (LCCs)* and *nonlinear-code chains (NCCs)* in the ROVSF code tree. Our code placement and replacement schemes initially attempt to allocate request codes on *linear-code chains,* and then try to allocate them to *nonlinear-code chains*. Finally, the simulation results illustrate that our code placement and replacement results based on the ROVSF code tree can actually improve the code blocking probability and reduce the code reassignment cost.

The rest of this paper is organized as follows. Section 2 introduces the ROVSF code tree. The code placement and replacement schemes based on the ROVSF code tree are presented in Section 3. In Section 4, we discuss the simulation results to demonstrate the improvement in performance. Section 5 concludes this paper.

2 Rotated-OVSF Code Tree

In this section, we initially define a new code tree, namely ROVSF (Rotated OVSF) code tree, and then propose a code placement/replacement algorithms on the ROVSF code tree. Channelization codes in the ROVSF code tree have a unique description as $RC_{SF,k}$, where SF is the spreading factor of the code and k is the code number, $1 \leq k \leq SF$. Each level in the code tree defines channelization codes of length SF, corresponding to a spreading factor of SF. The code $RC_{SF,k}$ is denoted as a ROVSF code, observe that if any ROVSF code $RC_{X,Y}$ is orthogonal to its two children codes, $RC_{2X,2Y-1}$ and $RC_{2X,2Y}$. The ROVSF code tree is formally defined as follows.

Definition 1. ***ROVSF Code Tree:*** *The ROVSF code of root node of ROVSF code tree is assumed as 1, and two children codes of the root node are initially set to be* $(-1,-1)$ *and* $(-1,1)$*, respectively. Consider a neighboring ROVSF codes* $RC_{i,j} = (A)$ *and* $RC_{i,j-1=}(B)$ *at k-th level,* $i = 2^k$*, of code tree, where A and B denote as the ROVSF codes of* $RC_{i,j}$ *and* $RC_{i,j-1}$*, respectively. Two children codes of* $RC_{i,j}$ *at* $(k+1)$*-th level of ROVSF code tree are* $RC_{2i,2j-1} = (-B,-B)$ *and* $RC_{2i,2j} = (B,-B)$*. Similarly, two children codes of* $RC_{i,j-1}$ *are* $RC_{2i,2j-2} = (-A,-A)$ *and* $RC_{2i,2j-3} = (A,-A)$*. Two codes* (P,Q) *and* (R,S) *are said as brother codes if* $Q = S$ *and* P *is the complement of* R*, i.e.,* $P = -R$*.*

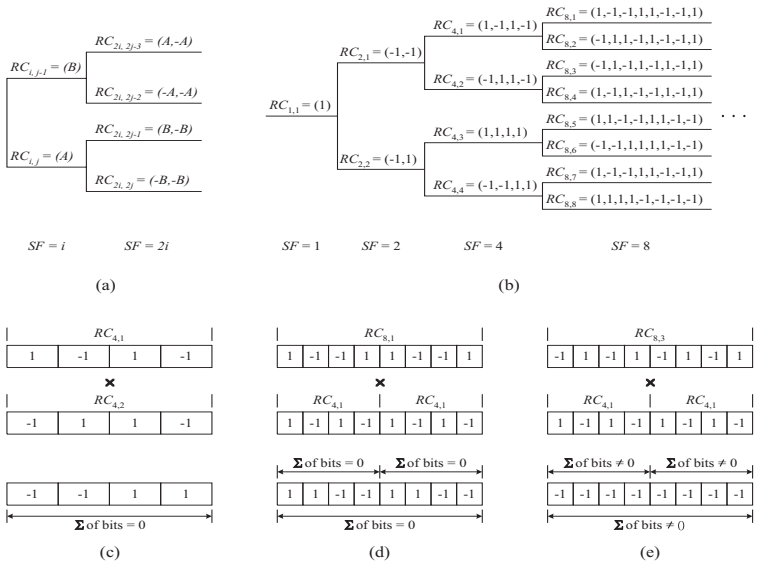

Fig. 1. Rotated OVSF (ROVSF) code-tree

Fig. 1 shows an example of the ROVSF code tree. The key feature existing in the ROVSF-based scheme to reduce the code blocking probability is the *linear-code chain*. Observe that every node of an OVSF code tree is logically mapped to the corresponding node of a ROVSF code tree. These mapping nodes can form a code-chain, which is called as a *linear-code chain*. Observe that, one or more linear-code chains may exist in a ROVSF code tree. A linear-code chain is formally defined as follows.

Definition 2. ***Linear-Code Chain:*** *Given a data rate,* $R_{\max} = 2^{\log_2 R_{\max}}$ *(or called a chain-max-code), we denote* S *as a linear-code chain as follows.*

1) Let linear-code chain S be a subset of $S_k = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \frac{R_{\max}}{2^3}, \cdots, \frac{R_{\max}}{2^k}]$, where $0 \leq k \leq \log_2(R_{\max})$,

   or

2) Let linear-code chain $S = S_k \cup \{\frac{R_{\max}}{2^k}\} = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \frac{R_{\max}}{2^3}, \cdots, \frac{R_{\max}}{2^k}, \frac{R_{\max}}{2^k}]$, where $\frac{R_{\max}}{2^k}$ and $\frac{R_{\max}}{2^k}$ are on the same level of the ROVSF code tree, and $0 \leq k \leq \log_2(R_{\max})$.

Consider a five-layer ROVSF code tree as shown in Fig. 2. For case 1 of definition 2, linear-code chains $[8R, 4R, 2R, 1R]$, $[8R, 4R, 2R]$, and $[8R, 4R]$ are illustrated in Fig. 2(a), (c) and (e), respectively. For case 2 of definition 2, linear-code chains $[8R, 4R, 2R, 1R, 1R]$, $[8R, 4R, 2R, 2R]$, and $[8R, 4R, 4R]$ are illustrated in

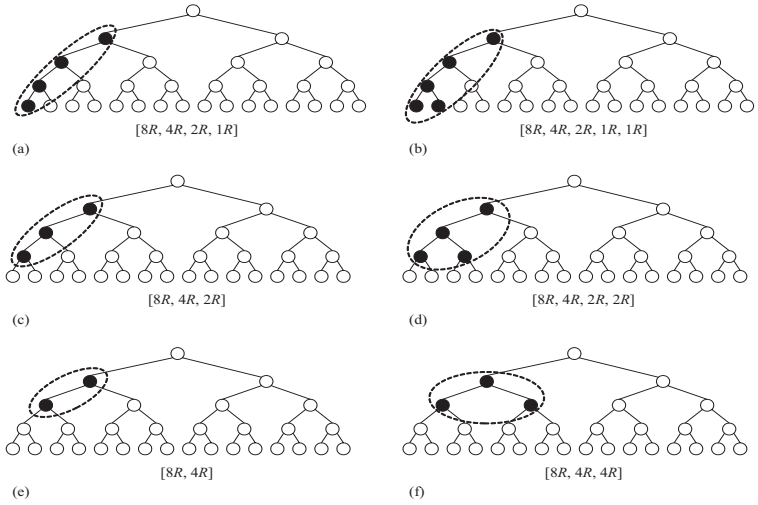

Fig. 2. Examples of LCCs

Fig. 2(b), (d), and (f), respectively, where $[1R, 1R]$, $[2R, 2R]$, and $[4R, 4R]$ are on the same respective level of the ROVSF code tree. The main idea of the ROVSF code placement and replacement scheme is to assign incoming requests to one of the linear-code chains. Furthermore, for case 1 of definition 2, we may use $(k+1)$ bit-word $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, b_1, b_0)$ to represent the linear-code chain, S, as being a subset of $S_k = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \frac{R_{\max}}{2^3}, \cdots, \frac{R_{\max}}{2^k}]$, where $k = \log_2(R_{\max})$. $b_i = 1$ indicates that $\frac{R_{\max}}{2^{k-i}}$ exists, and $b_i = 0$ indicates that $\frac{R_{\max}}{2^{k-i}}$ does not exist, where $0 \leq i \leq k$. For case 2 of definition 2, we also denote $(k+2)$ bit-word $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, (b_j, b_j), 0, \cdots, 0)$, where $0 \leq j \leq k$, as the linear-code chain S as a subset of $S_k = [R_{\max}, \frac{R_{\max}}{2^1}, \frac{R_{\max}}{2^2}, \frac{R_{\max}}{2^3}, \cdots, \frac{R_{\max}}{2^j}, \frac{R_{\max}}{2^j}]$, where $b_i = 1$ indicates that $\frac{R_{\max}}{2^{k-i}}$ exists, and $b_i = 0$ indicates that $\frac{R_{\max}}{2^i}$ does not exist, and $1 \leq i \leq k+1$. The ROVSF code tree has many properties, some important properties of ROVSF code trees used in this paper are summarized as follows.

P1: The ROVSF scheme offers the same code capability as OVSF-based schemes, and with most of the properties of OVSF code trees.

P2: The maximum data rate in an n-layer ROVSF tree is $2^{n-1}R$.

P3: Two ROVSF codes, $RC_{i,j}$ and $RC_{i,j'}$, at the k-th level, $i = 2^k$, of the ROVSF code tree are orthogonal. For instance as shown in Fig. 1(c), $RC_{4,1}$ is orthogonal to the same level code $RC_{4,2}$. Similarly, $RC_{4,3} = (1, 1, 1, 1)$ is orthogonal to $RC_{4,4} = (-1, -1, 1, 1)$.

P4: A ROVSF code, $RC_{i,j}$, is cyclically orthogonal to its two children codes, $RC_{2i,2j}$ and $RC_{2i,2j-1}$. For instance as shown in Fig. 1(d), $RC_{4,1}$ is cyclically orthogonal to $RC_{8,1}$. Similarly, $RC_{4,1} = (1, -1, 1, -1)$ is cyclically orthogonal to $RC_{4,2} = (-1, 1, 1, -1)$.

P5: A ROVSF code, $RC_{i,j}$, is cyclically orthogonal to any descendant codes of $RC_{i,j}$. This property is obtained from property 4 according to the transitive property.

P6: Given a pair of brother codes, $RC_{i,j} = (A, B)$ and $RC_{i,j-1} = (-A, B)$, $RC_{i,j}$ (or $RC_{i,j-1}$) is not cyclically orthogonal to any descendant codes of $RC_{i,j-1}$ (or $RC_{i,j}$). For instance as shown in Fig. 1(e), the cross-correlation between codes $RC_{4,1}$ and $RC_{8,3}$ is non-zero, and hence there will be interference if these codes are used.

P7: Consider for an n-layer ROVSF code tree, there exist $2^{n-\alpha-1}$ LCCs, for which a chain-max-code $R_{\max} = 2^{\alpha-1}$.

3 Code Placement and Replacement Strategies on ROVSF Code Tree

3.1 Code Placement Scheme

The main idea of the ROVSF code placement scheme is to initially assign request data-rate codes to the LCC. If we fail to find any feasible code in LCCs, we then try to find a possible code from NCCs. The main purpose of LCC is to reduce the code blocking probability. Observe that utilization of LCC significantly impacts the code blocking probability. Basically, the more the LCC is utilized, the lower the code blocking probability will be. The code placement scheme is divided into LCC and NCC phases as follows.

Linear-Code Chain (LCC) Placement Phase Consider an n-layer ROVSF code tree, for which $2^{n-\alpha-1}$ LCCs exist, where the chain-max-code $R_{\max} = 2^{\alpha-1}$. Our LCC placement phase attempts to assign incoming data rate, xR, to one of the $2^{n-\alpha-1}$ LCCs from left to right. Since each linear-code chain has its own bit-word, BW, this work can be achieved by checking the bit-word sequence $[BW_1, BW_2, \cdots, BW_{2^{n-\alpha-1}}]$. A *leftmost-assignment* strategy is performed as follows. We initially try to assign the incoming data rate, xR, to BW_1. If this fails, we continue by trying BW_2. We repeatedly execute the above operation until xR can be assigned to BW_j, where $j \leq 2^{n-\alpha-1}$. If we still cannot assign xR to $BW_{2^{n-\alpha-1}}$, then we perform the NCC placement phase, which is described later.

Now we describe how to assign xR to an LCC with BW, where $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, b_1, b_0)$, $b_i = 1, 0$, or $(1, 1)$, and $0 \leq i \leq k$, and BW is one among $BW_1, BW_2, \cdots$, and $BW_{2^{n-\alpha-1}}$. Let $\beta = \log_2 x$, where xR is the request data rate. Observe that if the LCC keeps $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, (b_j, b_j), 0, \cdots, 0)$, where $b_j = 1$, then we cannot assign any data rate, xR, into the LCC if $\beta < j$, where $\beta = \log_2 x$. But we still can assign data rate xR into the LCC if $\beta > j$ and $b_\beta = 0$. The formal algorithm of LCC placement operation is given.

S1: Repeatedly perform the following assignment to assign the incoming data rate, xR, into the i-th linear-code chain with a bit-word, BW_i, where $1 \leq i \leq 2^{n-\alpha-1}$ and $\beta = \log_2 x$.

(a) Try to assign xR into the i-th LCC to change BW_i from $(b_k, b_{k-1}, b_{k-2}, \cdots, (1,0)_j, 0, \cdots, 0)$ to $(b_k, b_{k-1}, b_{k-2}, \cdots, (1,1)_j, 0, \cdots, 0)$ if $b_{j=\beta} = 1$ and all $b_\gamma = 0$, where $r < \beta = j$. If the assignment is successful, then go to step **S2**.

(b) If a $(b_k, b_{k-1}, b_{k-2}, \cdots, (b_j, b_j), 0, \cdots, 0)$ exists and $\beta < j$, then the assignment fails even if $b_\beta = 0$. If $\beta > j$ and $b_\beta = 0$, then the assignment is successful; go to step **S2**.

(c) If $b_\beta = 1$ and one value of $b_\gamma = 1$, where $r < \beta$, then the assignment fails; go to step **S1**.

S2: If the incoming data rate, xR, cannot be assigned to the last LCC with bit-word, $BW_{2^{n-\alpha-1}}$, then execute the NCC placement phase.

Non-Linear-Code Chain (NCC) Placement Phase If incoming data rate, xR, fails to be assigned in the LCC placement phase, then the NCC placement phase is performed to assign xR into one of NCCs, where $\beta = \log_2 x$. Observe that an NCC is a subtree which contains no LCC. We give the formal algorithm of NCC placement operation.

S1: Repeatedly perform the following procedure to assign the incoming data rate, xR, to nodes of a neighboring subtree of the i-th LCC with bit-word, BW_i, where $1 \leq i \leq 2^{n-\alpha-1}$ and $\beta = \log_2 x$.

(a) If an LCC with bit-word, $BW_i = (b_\beta = 1, 0, 0, 0, \cdots, 0)$, indicates that node α in LCC is already assigned to the same data rate as xR; then we may assign xR to a neighboring node of node α of the LCC on level β of the ROVSF code tree.

(b) If an LCC with bit-word, $BW_i = (0, 0, \cdots, 0, (1,1)_j, 0, \cdots, 0)$ exists if $\beta = j$, then we may assign xR to nodes on the same level, j, of the neighboring subtree of the LCC.

S2: If the incoming data rate, xR, eventually cannot be assigned to neighboring codes of any LCCs, then xR is blocked.

3.2 Code Replacement Scheme

Our replacement scheme on the ROVSF code tree is investigated to completely eliminate the code blocking problem with less reassignment cost. The *code fragment* problem always occurs in a code tree after a number of variable data codes have been allocated and released. A code placement scheme already investigated the allocation policy in Section 3.1 to allocate the free code of rate xR, where x is a power of two. However, when no such free code exists but the remaining capacity of the code is sufficient, code blocking occurs. To completely eliminate code blocking with less reassignment cost, a replacement scheme is developed to relocate some codes in the ROVSF code tree to "squeeze" a free space for the new call. One other purpose of the replacement scheme is to overcome the drawback of the fixed lengths of the LCC and NCC; therefore a dynamic adjustment operation of LCC and NCC was designed.

To minimize the number of reassigned codes, Minn and Siu [3] developed a dynamic assignment of OVSF codes to provide a dynamic code assignment (DCA) scheme. The DCA algorithm is guaranteed to eliminate the code blocking problem. The use of linear-code chains can reduce the code blocking probability during the code placement operation. Our code replacement scheme aims to reduce the reassignment cost for ROVSF code tree management. Our proposed code replacement scheme on the ROVSF code tree is modified from the DCA algorithm [3] as follow.

Our ROVSF-based DCA-modified algorithm is given by providing a different rule of the *minimum-cost branch* [3] for the ROVSF code tree. Consider that a ROVSF code tree has sufficient code capacity, assuming that there are $2^{n-\alpha-1}$ nodes, $RC_{2^{n-\alpha},i}$, where $1 \leq i \leq 2^{n-\alpha}$. Node, $RC_{2^{n-\alpha},i}$, denoted as a *branch node* is used to maintain a *branch cost*. The *branch cost* is a count variable used to estimate the number of assigned codes. The *branch cost* also denotes the number of assigned codes which needs to be reassigned. The greater the number of the count variable is, the higher the reassignment cost will be. Each node, $RC_{2^{n-\alpha},i}$, forms a sub-tree, for which $RC_{2^{n-\alpha},i}$ is viewed as its root. Each node, $RC_{2^{n-\alpha},i}$, connects to an LCC or NCC, depending on whether the value of i is odd or even. For each $RC_{2^{n-\alpha},i}$, the branch cost, denoted $C_i = C(RC_{2^{n-\alpha},j})$, is determined by the total number of assigned nodes in the neighboring sub-tree or branch. Assuming that $RC_{2^{n-\alpha},i}$ and $RC_{2^{n-\alpha},j}$ are a pair of neighboring nodes, a minimum-cost branch is determined as follow. A branch node is said to be a minimum-cost branch, which is determined by $Min\ C_i$, where $1 \leq i \leq 2^{n-\alpha}$. To reorganize the ROVSF code tree structure in order to obtain a code with higher data rate, all assigned codes in the neighboring sub-tree are released and reallocated to other subtrees. The reallocating operation occurs by sequentially performing our code placement algorithm for each of the assigned codes in the neighboring sub-trees. The code replacement algorithm is formally given.

S1: Check to see if the code tree has sufficient code capacity to offer a new call with the data rate requirement xR. If it fails, reject this request.

S2: Once the request is allowed, a ROVSF-based DCA-modified algorithm is performed; a *minimum-cost branch* is initially found which accommodates the date rate, xR.

S3: If $\log_2(xR) > \log_2(R_{\max})$, where $R_{\max}$ is the *chain-max code* of the LCC, then go to step **S4** to execute the dynamic adjustment operation. Otherwise, go to step **S5**.

S4: If there exists $BW = (b_k, b_{k-1}, b_{k-2}, \cdots, b_1, b_0)$, where $b_i = 0$, the data rate of the incoming request is $2^{k+t}R$, and $1 \leq t \leq n-k$, the LCC is enlarged to be $(b_{k+t}, \cdots, b_k, b_{k-1}, b_{k-2}, \cdots, b_1, b_0)$. Observe that if all $b_{k+t}, b_{k+t-1}, \cdots,$ and b_{k+1} of all LCCs become to zero, then the LCCs are restored to be $(b_k, b_{k-1}, b_{k-2}, \cdots, b_1, b_0)$ or $(b_k, (b_{k-1}, b_{k-1}), 0, \cdots, 0)$.

S5: Sequentially execute the code placement algorithm to relocate all assigned codes in the neighboring subtree for the *minimum-cost branch*.

4 Simulation Results

To examine the effectiveness of our scheme, it mainly compared with OVSF-based code placement/replacement schemes in [6]. To make a fair comparison, we provide ROVSF-version *leftmost first*, *crowded-first,* and *mostuser-first* code placement strategies based on *leftmost first*, *crowded-first* and *mostuser-first* code placement strategies developed for the OVSF code tree [6]. In the following simulator, we used OR, OL, OC, OM, RL, RC, and RM, to denote different type of code trees and code placement strategies, where the first letter indicates the type of code tree (O = *OVSF-based* scheme and R = *ROVSF-based* scheme), and the second letter indicates the code placement strategy (R = *random*, L = *leftmost*, C = *crowded-first*, and M = *mostuser-first*). The system parameters in our simulator were as follows.

- Capacity test: code-limited.
- Maximum spreading factors: 64 and 256 to reflect values following IS-95 and WCDMA standards.
- Call arrival process: Poisson distribution with a mean arrival rate $\lambda = 1$ to 16 calls/unit time ($SF = 64$), and $\lambda = 4$ to 64 calls/unit time ($SF = 256$).
- Mean call duration time: Exponentially distributed with a mean value of four time units.
- Possible transmission rates: $1R$, $2R$, $4R$, and $8R$.

The performance metrics observed in this study are defined below.

1. *Blocking Probability*: The probability that a new incoming request cannot be accepted, although the system still has sufficient code capacity.
2. *Utilization of LCC*: The number of incoming requests assigned to LCCs divided by the total number of accepted requests.
3. *Number of Reassigned Codes*: The total number of code reassignments of all occupied codes for supporting a new call.

4.1 Impact of Code Placement

Different code placement strategies were adopted to observe the code blocking probability of a new call in the OVSF or ROVSF code tree even if the code tree has sufficient code capacity. The traffic pattern used was $1R$: $2R$: $4R$: $8R$ = 1: 1: 1: 1 to indicate that the probabilities of incoming requests when rates $1R$, $2R$, $4R$, and $8R$ are uniformly distributed. Fig. 3(a) shows the code blocking probability if only code placement is implemented, under a fixed mean arrival rate, and $SF = 256$. The ROVSF-based scheme with the leftmost strategy (RL) has the lowest code blocking probability. This is because code blocking always occurs for new calls with high transmission rates. A lower code blocking probability is obtained because our scheme tried to allocate most code requests onto LLCs. Using RL, our scheme can allocate more codes into LCCs. Only a few code requests, which are allocated outside LCCs, may experience code blocking

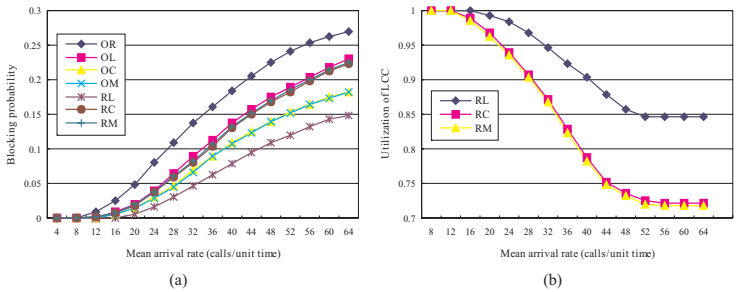

Fig. 3. $SF = 256$: (a) blocking probability at different traffic loads and (b) utilization of linear-code chains

problems. For example, when the mean arrival rate is equal to 64 with $SF = 256$, the code blocking probabilities are 15%, 18%, 18%, 22%, 22%, 23%, and 27% for the RL, OC, OM, RC, RM, and OR schemes, respectively.

To understand the effect of utilization of LCCs on the code blocking probability, the utilization of LCCs for RL, RC, and RM are given in Fig. 3(b) for $SF = 256$. Basically, the higher the utilization of LCCs is, the lower the code blocking probability that is obtained. For instance, the utilizations of LCC of RL are 1 and 0.85, and the code blocking probabilities are 0 and 0.15, respectively. Similar results for the RC and RM schemes can be obtained in Fig. 3(b). Observe that our scheme tries to properly assign most codes into LCCs, thus lower code blocking probabilities than other schemes can be obtained.

4.2 Impact of Code Replacement

The code replacement scheme is designed to completely eliminate the code blocking problem since there is sufficient free capacity. To see the efficiency of our code replacement scheme, observe the total number of code reassignments. In this simulation, we first considered the traffic pattern to be $1R$: $2R$: $4R$: $8R = 1$: 1: 1: 1. This indicates that we first considered equal probabilities for all incoming requests with data rates of $1R$, $2R$, $4R$, and $8R$. Various traffic patterns are investigated later. Considering the code replacement, therefore there is no code blocking problem. Therefore, we only observe the number of code reassignments. Fig. 4 shows the results of the number of code reassignments under a fixed mean arrival rate, λ, and $SF = 256$ and 64. This result shows that RL still has the lowest reassignment cost, when compared to all other schemes. This is possibly because the RL has better utilization of LCCs, thus it can reduce the total number of code reassignments to accommodate new calls with high data rates. For example as shown in Fig. 4(a) for $SF = 256$, if the mean arrival rate is 64, the numbers of code reassignments are 530, 660, 669, 784, 793, 810, and 1160 for RL, OM, OC, RC, RM, OL, and OR, respectively. Our ROVSF-based scheme

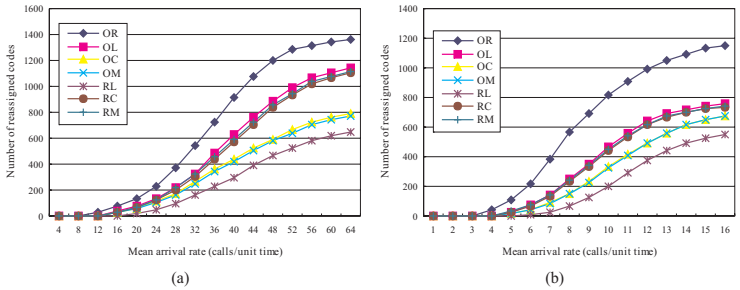

Fig. 4. Number of code reassignments to remove code-blocking: (a) $SF = 256$ and (b) $SF = 64$

has an improved number of code reassignments. Similar results can be found in Fig. 4(b) for $SF = 64$.

5 Conclusions

In this paper, we developed the code placement and replacement schemes on a ROVSF (rotated-orthogonal variable spreading factor) code tree. The simulation results illustrate that our code placement and replacement results based on the ROVSF code tree can actually improve the code blocking probability and lower the code reassignment cost.

References

1. A. Baier, U. C. Fiebig, W. Granzow, W. Koch, P. Teder, and J. Thielecke. Design Study for a CDMA-Based Third-Generation Mobile Radio System. *IEEE Journal on Selected Areas in Communications*, 12(4):733–743, May 1994.
2. M. Dell'Amico, M. L. Merani, and F. Maffioli. Efficient Algorithms for the Assignment of OVSF Codes in Wideband CDMA. In *Proceedings of IEEE International Conference on Communications (ICC'02)*, volume 5, pages 3055–3060, New York City, USA, April 2002.
3. T. Minn and K. Y. Siu. Dynamic Assignment of Orthogonal Variable-Spreading-Factor Codes in W-CDMA. *IEEE Journal on Selected Areas in Communications*, 18(8):1429–1440, August 2000.
4. Third Generation Partnership Project. Technical Specification Group Radio Access Network. Technical report, Spreading and Modulation, http://www.3gpp.org, 1999.
5. L. F. Tsaur and D. C. Lee. Symbol Rate Adaptation and Blind Rate Detection Using FOSSIL (Forest for OVSF-Sequence-Set-Inducing Lineages). In *Proceedings of IEEE International Conference on Communications*, volume 6, pages 1754–1759, 2001.
6. Y. C. Tseng and C. M. Chao. Code Placement and Replacement Strategies for Wideband CDMA OVSF Code Tree Management. *IEEE Transactions on Mobile Computing*, 1(4):293–302, October-December 2002.