An OVSF Code Assignment Scheme Utilizing Multiple Rake Combiners for W-CDMA

Li-Hsing Yen Ming-Chun Tsou
Dept. of Computer Science and Information Engineering
Chung Hua University
Hsinchu, Taiwan 30067, Republic of China
E-mail: {lhyen,m8902018}@chu.edu.tw

Abstract— **Orthogonal variable spreading factor (OVSF) codes have been proposed as the channelization codes used in the wide-band CDMA access technology of IMT-2000. OVSF codes have the advantages of supporting variable bit rate services, which is important to emerging multimedia applications. The objective of OVSF code assignment algorithm is to minimize the probability of code request denial due to inappropriate resource allocation. In this paper, we propose an efficient OVSF code assignment scheme that utilizes multiple Rake combiners in user equipments. Our approach finds in constant time all feasible codewords for any particular request, trying to minimize both rate wastage and code fragments. The simulation result shows that our scheme outperforms previous work in the probability of request denial. The code management overhead is also minimal in our scheme.**

I. Introduction

UMTS/IMT-2000 aims to provide not only voice services that are offered by the second-generation mobile systems, but also higher data rate and variable bit rate services that support differentiated quality-of-service (QoS) for emerging multimedia applications. W-CDMA, selected as the technology for use in the UMTS terrestrial radio access (UTRA), employs direct spread code division multiple access (DS-CDMA) technique, where each user equipment (UE) uses different orthogonal codes on the same frequency band. To support fixed- and mixed data rate services, the 3rd Generation Partnership Project (3GPP) proposed orthogonal variable spreading factor (OVSF) codes as the channelization codes used in W-CDMA. OVSF codes, providing variable data rates by using variable spreading factors (the number of chips for a data bit), can be generated according to an OVSF code tree. OVSF code tree is a binary tree where OVSF codes with spreading factor f are placed at the $(1 + \log f)$-th level. Each code in the tree has twice data rate than its child has. Once a code has been allocated to a UE, all its ancestors and descendants in the tree can no longer be allocated to other UEs. With the dynamic nature of code request arrival time and code usage time, it becomes an important performance issue how to effectively allocate codes to various data rate requests so as to minimize the code blocking rate (the probability that a request is rejected).

Existing code assignment algorithms toward this direction [1], [2], [5], [6] can be classified into two categories: single-code assignment and multi-code assignment. Single-code assignment assumes that each UE has only one Rake combiner so the system can only allocate one code to it. For multi-code assignment, a UE has more than one Rake combiner so its request can be honored by allocating multiple codes. Due to the dynamic nature of code usage, code assignment algorithm may suffer from code fragmentation problem, which refers to the circumstance that the system has enough capacity but cannot grant a request simply because the capacity does not belong to a single code. Code fragmentation may occur to both single-code and multi-code assignments, but it is believed that the problem is less serious in the multi-code case since data rates can be aggregated there. Code fragments can be compacted by using a code replacement algorithm that tries to exchange some allocated codes with unallocated codes of the equal spreading factor in order to obtain codes of lower spreading factors (i.e., higher data rates).

In the paper, we propose a multi-code assignment algorithm that aims to minimize the number of code fragments. The technique of dynamic programming is used to build a codeword table in advance so that when a request for data rate r issued by a UE equipped with k Rake combiners is received, all possible combinations of valid codewords can be found in the corresponding table entries. The one that results in the minimal number of code fragments is then selected for allocation.

The rest of the paper is organized as follows. The OVSF code system is described in Section 2. Section 3 details our channelization code assignment scheme. Simulation model and its results are shown in Section 4. Section 5 concludes our work.

II. Problem Definition

The OVSF codes can be defined using the code tree shown in Fig. 1. Each level in the code tree defines channelization codes of the same spreading factor (SF). Codes of the same SF have equal data rate, which is the double of that of the next level codes. We denote a single channelization code as $C_{f,n}$, where $f = 2^i$ for some integer i is the spreading factor of the code and n is a sequence number ranging from 0 to $f - 1$. We may omit the sequence number whenever it is irrelevant to our discussion.

In order to keep the orthogonality among codes of different levels, a code can be assigned to a UE if and only if no code on the path from this code to the root or any code in the subtree of

This work was supported by the National Science Council, Taiwan, under Contract NSC-91-2213-E-216-002

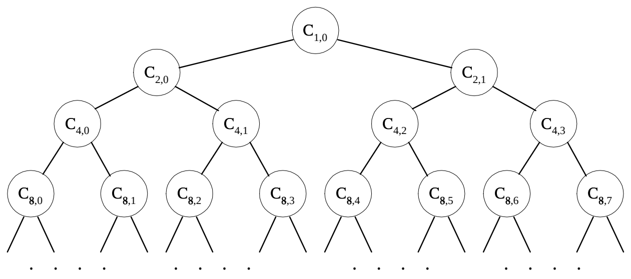

Fig. 1. An OVSF code tree

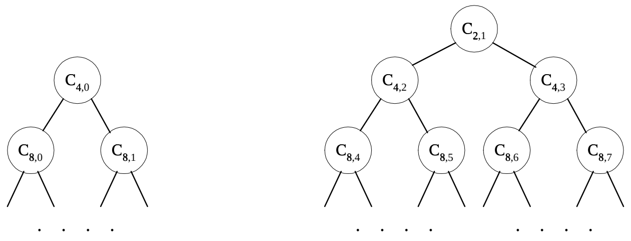

Fig. 2. Code fragments after allocating $C_{4,1}$

the code is assigned. In other words, once a code has been allocated to UE, all codes on the path from the root of the tree to this node and all codes in the subtree of the code can no longer be allocated. If we remove all such codes and associated incident edges from the code tree once a code has been allocated, the result will be a forest of complete binary trees, each represents a code fragment that can be directly allocated or decomposed for further code request. For example, if code $C_{4,1}$ of Figure 1 has been assigned to some UE, codes on the path from the root to $C_{4,1}$, *i.e.*, $C_{1,0}$ and $C_{2,0}$, and all codes in the subtree rooted at $C_{4,1}$ can no longer be allocated. After removing these nodes and associated incident edges, the result will be a forest of two complete binary trees, one representing code fragment $C_{4,0}$ and the other $C_{2,1}$, as shown in Fig. 2.

For practical reasons, the entire code tree is initially divided into four sub-trees, each represents a code fragment of spreading factor 4. Let

$$SF = \langle S_4, S_8, S_{16}, S_{32}, S_{64}, S_{128}, S_{256} \rangle$$

represent the numbers of various spreading codes available for allocation at any time, where S_f denotes the number of available code fragments of spreading factor f. Initially, $SF = \langle 4, 0, 0, 0, 0, 0, 0 \rangle$. Since one code fragment corresponds to one complete binary tree, the number of complete binary trees in the forest is simply $N(SF) = S_4 + S_8 + S_{16} + S_{32} + S_{64} + S_{128} + S_{256}$.

Let R be the basic data rate that a code of spreading factor 256 can offer. Clearly, the data rate offered by a code of spreading factor $f = 2^i$ is $2^{8-i}R$, where $i \in [2, 8]$ is an integer. When a UE requests a data rate r, any code of spreading factor $f' = 2^j$, where j is an integer in $[2, 8]$, such that $2^{8-j}R \geq r$ can be allocated to grant the request. How to select a valid code to grant the request is the *code placement* problem [7]. When no code can be allocated, a *code blocking* occurs. One reason behind code blocking is due to insufficient system capacity. The other reason is because of the occurrence of code fragmentation. *Code fragmentation problem* refers to the circumstance that the system has enough capacity but cannot grant a request simply because the capacity does not belong to a single code. Code fragmentation can be resolved by replacing some allocated codes with ones that are not allocated. For example, a system with $SF = \langle 0, 0, 0, 0, 0, 0, 2 \rangle$ cannot grant a request for data rate $2R$, since there is no code of spreading factor 128. However, by switching a allocated code of spreading factor 256 with an available equal-rate code, two available codes of spreading factor 256 can be combined into a single code of spreading factor 128. How to find and replace code fragments to grant a request is the *code replacement* problem [7]. Code placement scheme, perhaps together with code replacement scheme, aims to reduce code blocking rate, *i.e.*, the probability that a code request cannot be granted.

If we allocate a code with data rate higher than that the UE actually requests, some amount of date rate is wasted. Rate wastage is critical to the code blocking rate (it diminishes system capacity) and should be kept minimal whenever possible.

Solving code placement problem involves two steps. First, determine the code rate to be allocated. Second, locate a code in the code tree that corresponds to the code rate. If UE is equipped with only one Rake combiner, the first step is straightforward: find a code rate 2^iR that minimizes rate consumption. As to locate a code of rate 2^iR in the code tree, there have been several approaches proposed. Among them, leftmost allocation [4] and crowded-first [7] allocation will be mentioned in our discussion.

If UE is equipped with k Rake combiners ($k > 1$), we can assign UE a *multi-code*, which is a collection of maximal k codes with aggregated data rate $(2^{i_1} + 2^{i_2} + \cdots + 2^{i_k})R$ ($i_1, i_2, \ldots, i_k$ are integers in $[0, 6]$). Determining the set of multi-codes that minimize rate consumption is not trivial since many combinations of codes need to be considered. As an example, if a UE equipped with three Rake combiners demands data rate of $6R$, totally three multi-codes minimize rate consumption (actually they waste no rate), as shown below:

- one code of spreading factor 64 (rate $4R$) plus one code of spreading factor 128 (rate $2R$)
- one code of spreading factor 64 (rate $4R$) plus two codes of spreading factor 256 (rate R each)
- three codes of spreading factor 128 (rate $2R$ each)

In general, rate wastage will be less serious if UEs have multiple Rake combiners rather than only one, since code fragments can be better utilized with multiple Rake combiners. Once the set of minimal-wastage multi-codes is determined, we must select one multi-code to be allocated to the UE. A good multi-code assignment algorithm should select a multi-code that results in lower code blocking rate. However, as future requests cannot be known in advance, an optimal on-line code assignment algorithm seems impossible. Existing solutions thus take heuristic approach. The scheme in [2] favors allocating the multi-code that results in the most small-SF codes left. If more than two multi-codes are favored, the one that uses the least number of codes is selected. In [6], Shueh et al. proposed a scheme that first chooses a multi-code that represents the request rate in binary-number form. The multi-code is then adjusted in accordance with the number of Rake combiners that

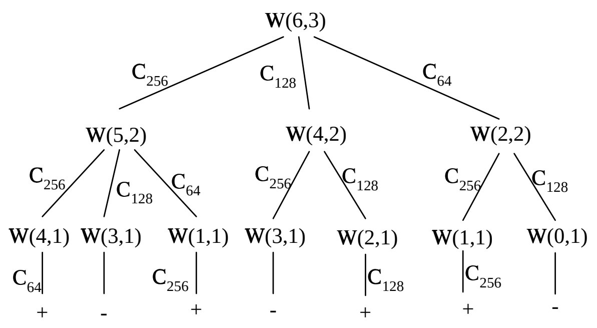

Fig. 3. Process of evaluating $W(6,3)$

the UE has. This approach may allocate more small-SF codes than necessary, therefore decreasing code utilization.

III. Proposed Scheme

Let $Req(r,k)$ be a code request for some data rate r which is submitted by a UE equipped with k Rake combiners. The basic idea behind our scheme is that the set of all possible multi-codes that satisfy $Req(r,k)$ can be evaluated off-line and in advance. In case we have two or more multi-code candidates, we select the one that results in the least number of code fragments left.

A multi-code is denoted as $C = \langle N_4, N_8, N_{16}, N_{32}, N_{64}, N_{128}, N_{256}\rangle$, where N_i denotes the number of codes with SF=i that C uses. The number of codes used by multi-code C is $N(C) = N_4 + N_8 + N_{16} + N_{32} + N_{64} + N_{128} + N_{256}$. When $N(C) = 1$, the multi-code C may be simply denoted as C_i, where i is the spreading factor of the only code used.

Let $C = \langle N_4, N_8, N_{16}, N_{32}, N_{64}, N_{128}, N_{256}\rangle$ and $C' = \langle N'_4, N'_8, N'_{16}, N'_{32}, N'_{64}, N'_{128}, N'_{256}\rangle$ be two multi-codes. We define $C + C'$ as $\langle N_4 + N'_4, N_8 + N'_8, N_{16} + N'_{16}, N_{32} + N'_{32}, N_{64} + N'_{64}, N_{128} + N'_{128}, N_{256} + N'_{256}\rangle$. Let $W(r,k)$ be the set of all possible multi-codes that have data rate r and use k codes. We define $C \oplus W(r,k)$ as $\{C + C' | C' \in W(r,k)\}$. $W(r,k)$ can be evaluated as follows. $W(r,1) = \{C_f\}$ if $r = 256/f$ and $W(r,1) = \emptyset$ otherwise. For a general $W(r,k)$, we can examine every integer g such that $0 \le g \le \lfloor \log_2 r \rfloor$ to see if any multi-code of $W(r,k)$ may include a code of spreading factor 2^{8-g} (*i.e.* $C_{2^{8-g}}$). If the code is indeed included, all the multi-codes in $W(r,k)$ that include it can be obtained by $C_{2^{8-g}} \oplus W(r - 2^g, k-1)$. Therefore we have

$$W(r,k) = \bigcup_{g=0}^{\lfloor \log_2 r \rfloor} C_{2^{8-g}} \oplus W(r - 2^g, k-1)$$

In this way, we decompose the problem of solving $W(r,k)$ into smaller sub-problems, which can be further decomposed. Fig. 3 shows the process of evaluating $W(6,3)$. Each path from the root to a leaf labeled '$+$' sign represents a valid multi-code. The multi-code consists of every code that is associated with an edge on the path.

In solving a particular $W(r,k)$, there may be some duplicated $W(r',k')$'s to be evaluated, where $r' < r$ and $k' < k$.

$k \backslash r$	1	2	3	4	5	6	…
1	(0,0,0,0,0,0,1)	(0,0,0,0,0,1,0)	-	(0,0,0,0,1,0,0)	-	-	…
2	-	(0,0,0,0,0,0,2)	(0,0,0,0,0,1,1)	(0,0,0,0,0,2,0)	(0,0,0,0,1,0,1)	(0,0,0,0,1,1,0)	…
3	-	-	(0,0,0,0,0,0,3)	(0,0,0,0,0,1,2)	(0,0,0,0,0,2,1)	(0,0,0,0,1,0,2) (0,0,0,0,0,3,0)	…
4	-	-	-	(0,0,0,0,0,0,4)	(0,0,0,0,0,1,3)	(0,0,0,0,0,2,2)	…
…	…	…	…	…	…	…	…

TABLE I
Table for $W(r,k)$

Multi-code	SF'	$N(SF')$
$\langle 0,0,0,0,1,1,0\rangle$	$\langle 0,0,0,0,0,2,1\rangle$	3
$\langle 0,0,0,0,1,0,2\rangle$	$\langle 0,0,0,0,0,2,2\rangle$	4
$\langle 0,0,0,0,0,3,0\rangle$	$\langle 0,0,0,0,1,0,1\rangle$	2

TABLE II
Possible code allocations for $Req(6,3)$ with $SF = \langle 0,0,0,0,1,3,1\rangle$. SF' denotes the value of SF after code allocation.

We use the technique of dynamic programming [3] to avoid redundant computation. Table I shows the result.

Let $M(r,k) = \bigcup_{1 \le j \le k} W(r,j)$. Given a request $Req(r,k)$, all possible multi-codes that can be assigned without rate wastage are in the set $M(r,k)$. If $|M(r,k)| = 0$, we look up for the minimal $r' > r$ such that r' is not greater than the system capacity and $M(r',k)$ is not empty. If no such r' can be found, this request is rejected due to insufficient system capacity. Otherwise, we grant the request with rate wastage $(r' - r)$.

If the set of minimal-wastage multi-codes has only one element ($|M(r,k)| = 1$ or $|M(r',k)| = 1$), the only multi-code is used. If the set has two or more multi-codes, we select the one that results in the least number of code fragments. For example, assume that $SF = \langle 0,0,0,0,1,3,1\rangle$ and a request $Req(6,3)$ arrives. From Table I we know that $M(6,3) = \{\langle 0,0,0,0,1,1,0\rangle, \langle 0,0,0,0,1,0,2\rangle, \langle 0,0,0,0,0,3,0\rangle\}$. Table II shows all possible allocations. Clearly, multi-code $\langle 0,0,0,0,0,3,0\rangle$ results in the least number of code fragments and thus will be selected for allocation.

If two or more multi-codes result in the least number of fragments, the one that uses least codes will be selected. As an example, consider $Req(6,3)$ and $SF = \langle 0,0,0,1,2,1,0\rangle$. Table III shows all possible allocation results. As all three multi-codes result in two code fragments left, the one that uses least codes, i.e., $\langle 0,0,0,0,1,1,0\rangle$, will be selected.

Once the multi-code is selected, how to locate all code fragments of the multi-code in the code tree becomes straightforward. We may allocate these codes one by one, using any approach that is proposed for single-code allocation. If any one of the code fragment cannot be located due to code fragmentation problem, a code replacement algorithm is invoked to resolve the problem.

Multi-code	SF'	$N(SF')$
$\langle 0,0,0,0,1,1,0\rangle$	$\langle 0,0,0,1,1,0,0\rangle$	2
$\langle 0,0,0,0,1,0,2\rangle$	$\langle 0,0,0,1,1,0,0\rangle$	2
$\langle 0,0,0,0,0,3,0\rangle$	$\langle 0,0,0,1,1,0,0\rangle$	2

TABLE III
POSSIBLE CODE ALLOCATIONS FOR $Req(6,3)$ WITH $SF = \langle 0,0,0,1,2,1,0\rangle$. SF' DENOTES THE VALUE OF SF AFTER CODE ALLOCATION.

IV. SIMULATION

We evaluated the performance of our multi-code assignment algorithm by simulation. The performances of [2] (referred to as CL00) and [6] (referred to as SLC00) are also evaluated for comparison. We assume that the depth of the OVSF code tree is 9 and the initial set of multi-codes of the system is $\langle 0,0,4,0,0,0,0,0,0\rangle$. Code requests are randomly generated by a Poisson process with mean arrival rate λ. The time interval for which a multi-code is used is assumed to be an exponentially distributed random variable with mean $1/\mu$. Each request $Req(r,k)$ is randomly generated with r being uniformly distributed between 1 and M, where M is 64 or 128. The value of k depends on r. When $1 \leq r \leq 64$, k is a random variable uniformly distributed between 1 and 4. When $65 \leq r \leq 128$, k is uniformly distributed in [2, 4]. Total 50000 requests are generated for each round. All algorithms use the same trace of each round as their input.

We measured the code blocking rate caused by each algorithm. The measured code blocking rate is essentially the sum of two components: one is due to insufficient system capacity and the other is due to code fragmentation. In the following figures, our method is labeled with TY02B. TY02A refers to a variation of our method that considers only multi-codes consisting of exactly k codes when handling $Req(r,k)$ (while TY02B considers all multi-codes that use k *or less* codes). Once a multi-code has been chosen, leftmost [4] or crowded-first [7] code placement algorithm is used to allocate one by one all associated codes.

Figs. 4 and 5 show the results of $M = 64$ with leftmost and crowded-first code placement algorithms, respectively. Clearly, TY02B results in the lowest blocking rate in both scenarios. The performance of these four methods is about the same with $M = 128$, though TY02B slightly outperforms others. We also found that the performances with leftmost and with crowded-first code placement algorithms are about the same.

We also conducted experiments to investigate the impacts of code replacement algorithm on code blocking rate. Table IV shows the result of using DCA scheme [5] as the code replacement algorithm. With the aid of code replacement algorithm, code fragmentation can be totally eliminated and code blocking now occurs only for the reason of insufficient system capacity. Therefore, all four methods result in the same code blocking rate. However, each method requires different number of code replacements. The number of code replacements is considered the overhead of code management. Figs. 6 to 9 show the number of code replacements in different settings. In all settings, TY02B requires the least number of code replacements.

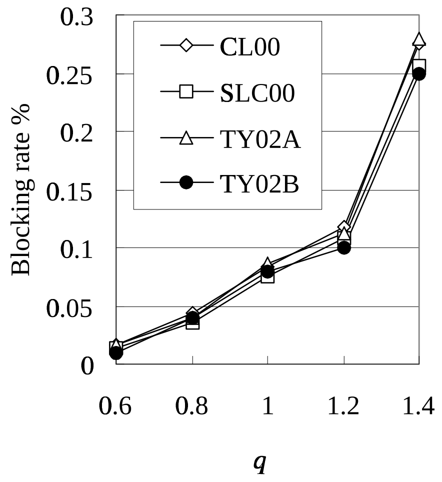

Fig. 4. Code blocking rate with leftmost code placement ($M = 64$, $q = \lambda/\mu$)

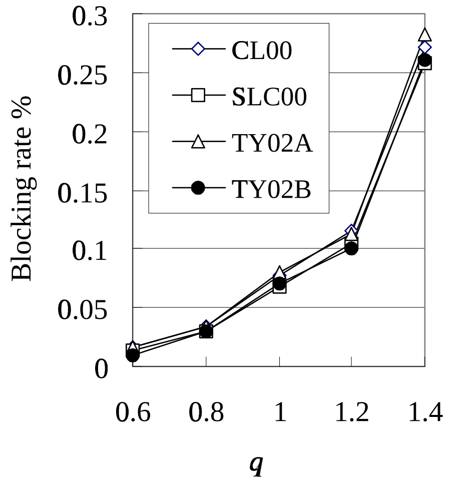

Fig. 5. Code blocking rate with crowded-first code placement ($M = 64$, $q = \lambda/\mu$)

As we mentioned before, TY02A is a variation of TY02B that considers only multi-codes in $W(r,k)$ rather than multi-codes in $M(r,k) = \bigcup_{1\leq j\leq k} W(r,j)$ when handling request $Req(r,k)$. Apparently, TY02A examines less multi-codes than TY02B does. We are interested in the amount of multi-code examinations that TY02A can save. Table V shows the ratio of average number of multi-codes examined by TY02A to that examined by TY02B. We can see that about two third examinations can be saved.

	λ/μ				
	0.6	0.8	1.0	1.2	1.4
$M = 64$	0.012%	0.026%	0.062%	0.086%	0.218%
$M = 128$	2.56%	3.88%	5.46%	7.44%	8.65%

TABLE IV
CODE BLOCKING RATE WITH CODE REPLACEMENT

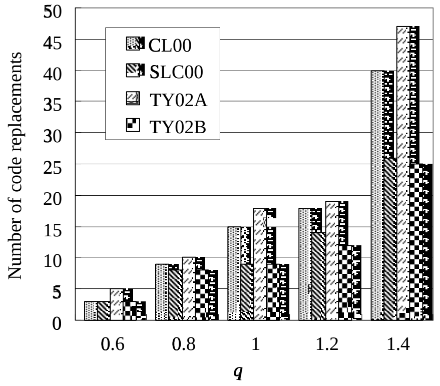

Fig. 6. Number of code replacements with leftmost code placement ($M = 64$, $q = \lambda/\mu$)

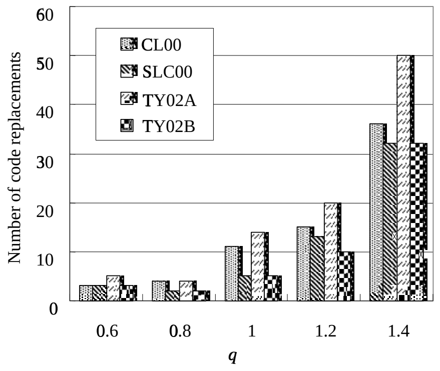

Fig. 7. Number of code replacements with crowded-first code placement ($M = 64$, $q = \lambda/\mu$)

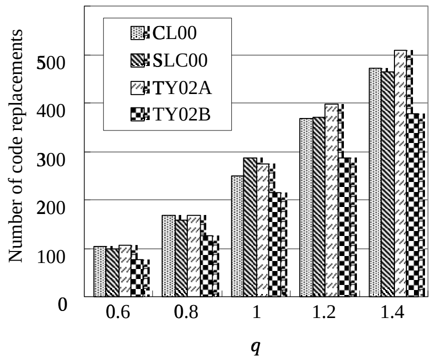

Fig. 8. Number of code replacements with leftmost code placement ($M = 128$, $q = \lambda/\mu$)

	λ/μ				
	0.6	0.8	1.0	1.2	1.4
$M = 64$	36.29%	36.28%	36.28%	36.27%	36.29%
$M = 128$	31.39%	31.43%	31.47%	31.49%	31.50%

TABLE V

RATIO OF AVERAGE MULTI-CODE EXAMINATIONS BY TY02A TO THAT BY TY02B

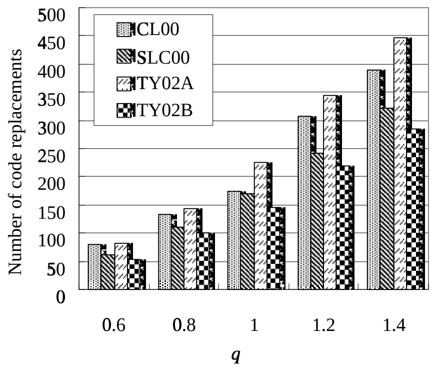

Fig. 9. Number of code replacements with crowded-first code placement ($M = 128$, $q = \lambda/\mu$)

V. CONCLUSIONS

The objective of OVSF multi-code assignment algorithm is to minimize code blocking rate. The causes of code blocking are due to insufficient system capacity and code fragmentation. To preserve system capacity, rate wastage must be kept minimized. To eliminate code fragmentation, we can invoke a code replacement procedure. If UEs have multiple Rake combiners, rate wastage can be reduced while the frequency of invoking code replacement can be lowered.

In this paper, we have proposed a multi-code assignment scheme that utilizes multiple Rake combiners in UEs. By using dynamic programming technique, our approach finds in constant time all feasible multi-codes for any particular request. It then selects the one that will bring about both minimal rate wastage and least code fragments. The simulation result shows that our scheme outperforms previous work in the code blocking rate. The number of code replacements required is also minimal in our scheme.

REFERENCES

[1] Wen-Tsuen Chen, Hung-Chang Hsiao, and Ya-Ping Wu. A novel code assignment scheme for W-CDMA systems. In *Proc. of IEEE 2001 Vehicular Technology Conference*, volume 2, pages 1182–1186, 2001.

[2] Ray-Guang Cheng and Phone Lin. Ovsf code channel assignment for IMT-2000. In *Proc. of IEEE 2000 Vehicular Technology Conference*, volume 3, pages 2188–2192, May 2000.

[3] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to algorithms*. McGraw-Hill Book Company, 1990.

[4] R. Fantacci and S. Nannicini. Multiple access protocol for integration of variable bit rate multimedia traffic in UMTS/IMT-2000 based on wideband CDMA. *IEEE Journal on Selected Areas in Communications*, 18(8):1441–1454, August 2000.

[5] 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.

[6] Fenfen Shueh, Zu-En Liu, and Wen-Shyen Chen. A fair, efficient, and exchangeable channelization code assignment scheme for IMT-2000. In *Proc. of 2000 IEEE International Conference on Personal Wireless Communications*, pages 429–433, 2000.

[7] Yu-Chee Tseng, Chih-Min Chao, and S.-L. Wu. Code placement and replacement strategies for wideband CDMA OVSF code tree management. In *Proc. of IEEE GlobeCom*, 2001.