Implementation of an Efficient Channelization Code Assignment Algorithm in 3G WCDMA

第三代無線通訊 WCDMA 信道碼配置演算法之設計

Jui-Chi Chen and Wen-Shyen E. Chen

陳瑞奇 陳文賢

Institute of Computer Science
National Chung Hsing University
250 Kuokuang Road, Taichung, Taiwan, R.O.C.
{rjchen, echen}@cs.nchu.edu.tw

摘要

第三代無線通訊 WCDMA 信道碼是非常寶貴的資源，故信道碼配置演算法設計之目的在於盡可能以最小的複雜度支援最多的通訊使用者，過去已有一些配置演算法被設計提出，但都有"碼碎裂"的問題。為了解決此一問題，碼交換與重配置的策略必須被考慮進來，但又會造成系統複雜度的增高。在這篇論文中，我們提出一個有效率的 BLRU 信道碼配置演算法，可以以較小的複雜度且不用碼交換與重配置的方法來趨近最佳化的配置結果。本研究模擬比較幾個方法之執行阻斷率的高低，若在考慮系統複雜度與執行阻斷率的兩項因素下，我們所提出的 BLRU 策略是第三代無線通訊 WCDMA 信道碼配置法一個很好的選擇。

關鍵詞：WCDMA, 信道碼配置, 正交變展頻碼

Abstract

Channelization codes are valuable and scarce resources in 3G WCDMA. The objective of channelization code assignment is to support as many users as possible with less complexity. Some code assignment algorithms have been proposed. Most of the algorithms suffer from code-set fragmentation problem. To resolve this issue, code exchange and reassignment policies can be applied, but the system complexity will be increased. In this paper, we propose an efficient BLRU (Best-fit Least Recently Used) code assignment algorithm with less fragmentation and complexity to approximate the reassignment based optimal solution. Simulation results compare the blocking probability among three approaches. Considering both the system complexity and the blocking probability, the BLRU algorithm is a good candidate for channelization code assignment in 3G WCDMA.

keyword：WCDMA, Channelization Code Assignment, OVSF code

* This research is supported by the National Science Council of the Republic of China under grant number NSC89-2213-E-005-058.

I. Introduction

The third generation (3G) mobile systems are characterized by high throughput, wideband services, and flexibility. Direct sequence code division multiple access (DS-CDMA) [1-2] predominates in the wireless access technology for 3G systems because of its large capacity and high flexibility in offering variable-rate services [3-7]. In Universal Mobile Telecommunication System (UMTS) wideband CDMA (WCDMA), spreading code transmission supports a variety of wideband services from low to very high data rates [8]. One of the spreading codes is channelization code. The channelization codes are orthogonal variable spreading factor (OVSF) codes that preserve the orthogonality between channels of different rates and spreading factors. From the point of view of a code-limited system, they are valuable and scarce resources.

For such a reason, the objective of OVSF code assignment is to support as many users as possible with less complexity. Some single code and multicode assignment algorithms have been proposed [3-7]. Observing some different criteria to compare the single code and multicode policies, none of them provides obvious superiority [9-12]. In addition, both single code and multicode assignment algorithms suffer from fragmentation of the available codes in a system. The fragmentation degrades the performance of code assignment. In order to resolve this issue, code exchange and reassignment policies can be deployed [3]. However, the system complexity (SC) is also increased due to those extra efforts dealing with code exchange and reassignment processes.

In this paper, we propose an efficient code assignment algorithm with less fragmentation and complexity, called Best-fit Least Recently Used (BLRU) algorithm, to

approximate the reassignment based optimal solution concerned with blocking probability (BP). The BLRU algorithm needs neither code exchange nor reassignment process; therefore, SC is decreased. At the same time, we devise a compact data structure to store an OVSF code table and an unused-code list. It can save extra storage for the unused code list and avoid search operation of the OVSF code tree. Additionally, an OVSF code generation function is designed to generate the corresponding codeword, instead of storing a real codeword straight in an array, to save about 89 percent of space. As a result, it is especially suitable for mobile sets (MS).

The rest of this paper is organized as follows. The OVSF code system is described in Section II. Section III illustrates a compact data structure and the BLRU code assignment algorithm. In order to realize the cost of the BLRU algorithm, Section IV evaluates its time and space complexity. Simulation results are shown in Section V. Finally, concluding remarks are given in Section VI.

II. OVSF Code System

The channelization operation in WCDMA transforms each data symbol into a number of chips. The number of chips per data symbol is called spreading factor. The channelizing structure of the reverse link (uplink) in the OVSF code system is shown in Figure 1. The data symbols are spread in channelization operation firstly and then scrambled in scrambling operation [4].

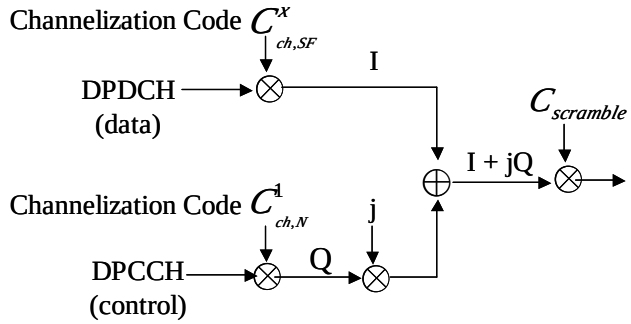


Figure 1: Channelizing structure of the reverse link in the OVSF code system

In channelization operation, a code tree shown in Figure 2 recursively generates the OVSF codes based on a modified Walsh-Hadamard transformation [4]. For example, $C^k_{ch,N}$ uniquely describes the codes, where N is the spreading factor of the code, k is the code number, and $1 \le k \le N$. Let $C_{ch,N}$ denote the set of N binary OVSF codes $\{C^k_{ch,N}\}^N_{k=1}$, where $N=2^n$ ranges from 4 to 256 and $C^k_{ch,N}$ is one N elements' row vector. Then the generation method for the OVSF code-set is described as

$$C^1_{ch,1} = 1,$$

$$\begin{bmatrix} C^1_{ch,2} \\ C^2_{ch,2} \end{bmatrix} = \begin{bmatrix} C^1_{ch,1} & C^1_{ch,1} \\ C^1_{ch,1} & \overline{C^1_{ch,1}} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$C_{ch,N} = \begin{bmatrix} C^1_{ch,N} \\ C^2_{ch,N} \\ \cdots \\ C^N_{ch,N} \end{bmatrix} = \begin{bmatrix} C^1_{ch,N/2} & C^1_{ch,N/2} \\ C^1_{ch,N/2} & \overline{C^1_{ch,N/2}} \\ C^2_{ch,N/2} & C^2_{ch,N/2} \\ C^2_{ch,N/2} & \overline{C^2_{ch,N/2}} \\ \cdots & \cdots \\ C^{N/2}_{ch,N/2} & C^{N/2}_{ch,N/2} \\ C^{N/2}_{ch,N/2} & \overline{C^{N/2}_{ch,N/2}} \end{bmatrix},$$

where $\overline{C^k_{ch,N/2}}$ is the binary complement of $C^k_{ch,N/2}$. The vector $C^k_{ch,N} = (c^k_1, c^k_2, ..., c^k_N) \in \{\pm 1\}^N$ can be called a codeword. If $C^k_{ch,N}$ is used (assigned), other users cannot use all ancestor codes of $C^k_{ch,N}$ and all descendant codes generated from this code. Certainly, all codes of $C^k_{ch,N}$ are orthogonal to each other.

Furthermore, both the forward link (downlink) and the reverse link in WCDMA can apply OVSF code(s) to match the request data rate. Without loss of generality, the data rate described in this paper is normalized by the basic data rate R_b of an OVSF code with the maximum spreading factor (*MaxSF*). We assume that *MaxSF* = 256 here.

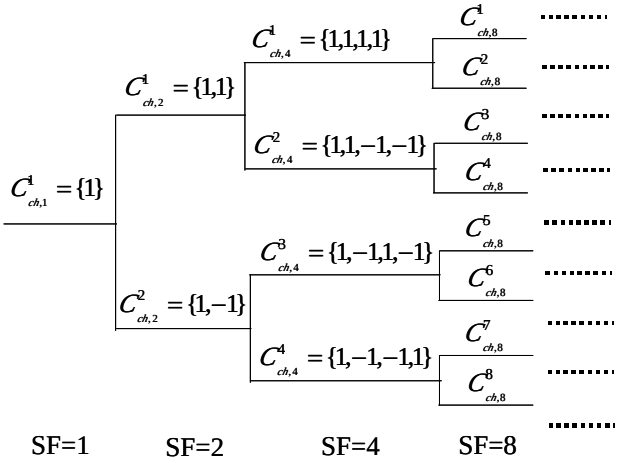


Figure 2: Code tree for generation of the OVSF codes

Actually, the number of the OVSF codes is limited. The base station (BS) may therefore run out of the codes and block an incoming call. Four issues affecting the BP

of the OVSF code system are stated in the following. The first is how much bandwidth the system can provide. An overloaded system blocks incoming calls more frequently. The second is the arrival rate (calls per unit time) and the incoming call-duration. High arrival rate and long call-duration may dramatically increase the system load. The third is the variable request data rate. An incoming call is blocked if the available codes in BS are not enough to fulfill the request. The fourth is the grade of OVSF code-set fragmentation. After a sequence of code assignment and release operations, the OVSF code tree consists of many fragmental nodes. This fragmentation will degrade the performance of the code assignment. For example, as shown in Figure 3, a new call requesting a single code $C^{x}_{ch,2}$, which cannot be supported, will be blocked although the system has enough overall capacity (the aggregation of $C^{3}_{ch,4}$, $C^{1}_{ch,8}$, and $C^{8}_{ch,8}$ equals to the code $C^{x}_{ch,2}$). This is called "fragmentation," which results in less code assignment flexibility and higher BP. A dynamic assignment approach in [3] can gather the available codes together dynamically so as to provide more flexibility by exchanging them. One disadvantage of the approach is the increased SC, because it needs to deal with the code exchange and reassignment processes.

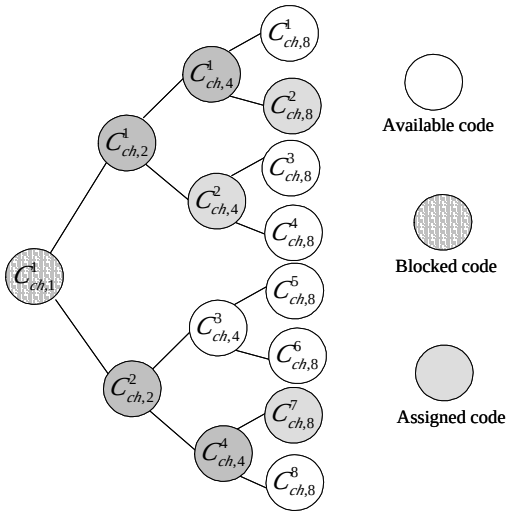
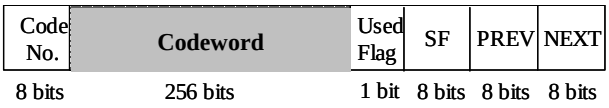


Figure 3: An example of OVSF call blocking due to fragmentation

III. Best-fit Least Recently Used Algorithm

To achieve less fragmentation and complexity, we propose an efficient BLRU code assignment algorithm.

Firstly, a compact data structure is devised to store the OVSF code table and the unused code list. It can save extra storage for the list and avoid searching the code tree. The primary structure of a code entry in an OVSF code table implemented by a 2D array is shown in Figure 4. Field Code No. (e.g., *n*) is the index to the *n*th code entry. Shaded Codeword is a binary sequence that represents an OVSF code by taking its prefix SF bits. This field is not required if the Decimal Walsh Code Generating Function (DWCGF, will be illustrated in Section IV) is further applied. The Used Flag field is set to "false" initially and is changed to "true" as the codeword in the same entry is assigned. Field SF stores the current spreading factor possessed by a code entry. Both fields PREV and NEXT are used for the unused (available) OVSF code list shown in Figure 6. Given a code entry in the unused code list, field NEXT points to its successor, and field PREV points to its predecessor. The unused code list is therefore embedded in the code table; that is, we do not need any other additional memory to store the list. Moreover, the code table embeds the OVSF code tree, all codeword entries, and used code records.



SF: spreading factor
PREV: the pointer to its predecessor
NEXT: the pointer to its successor

Figure 4: The primary structure of a code entry in an OVSF code table

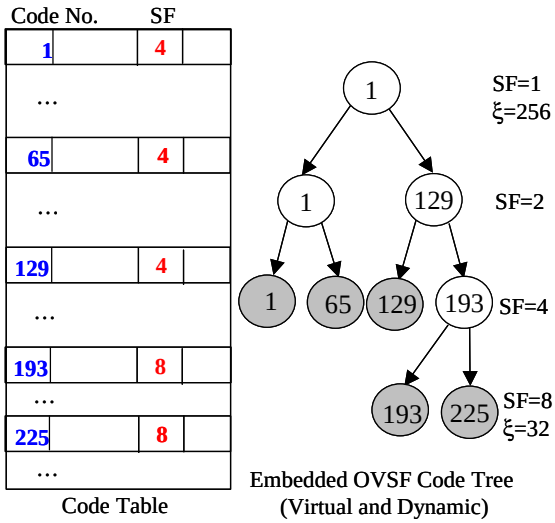


Figure 5: An example presents that the code table embeds the OVSF tree

An example of the embedded OVSF code tree can be seen in Figure 5. For each code *i*, its dynamic left child

code is code no. i and right child code is code no. i + $MaxSF/2$*SF. When a user requires a code with SF=8, Code 193 (SF = 4) needs to be split into two codes:Code 193 and Code 225, with SF = 8. The value of field SF of each code entry is dynamically changed according to the current status. When Code 1 and Code 65 with SF = 4 are released, these two codes should be merged into Code 1 with SF = 2. Thus, the code table simultaneously maintains the OVSF code tree. Meanwhile, this code table requires only 256 code entries and saves 255 nodes and 254 links in a real OVSF code tree.

Figure 6 demonstrates the unused code list organized by a logical double-linked list and ordered by the spreading factor. In fact, the list is embedded in the code table as described above. Besides pointers PREV and NEXT in each code entry, the code table requires some other pointers. Pointer Head points to the first element of the list, pointer Tail points to the last element of the list, and other $\log_2(MaxSF) + 1$ pointers (namely, SFpointer[x], $0 \le x \le 8$ if $MaxSF = 256$) let the BLRU algorithm directly index into the best-fit code entries. The unused code list needs only to link those code entries which are the roots of unused code sub-trees. For example, as shown in Figure 3, only $C_{ch,4}^{3}$, $C_{ch,8}^{1}$, and $C_{ch,8}^{8}$ codes have to be linked in the list, but not $C_{ch,8}^{5}$ or $C_{ch,8}^{6}$.

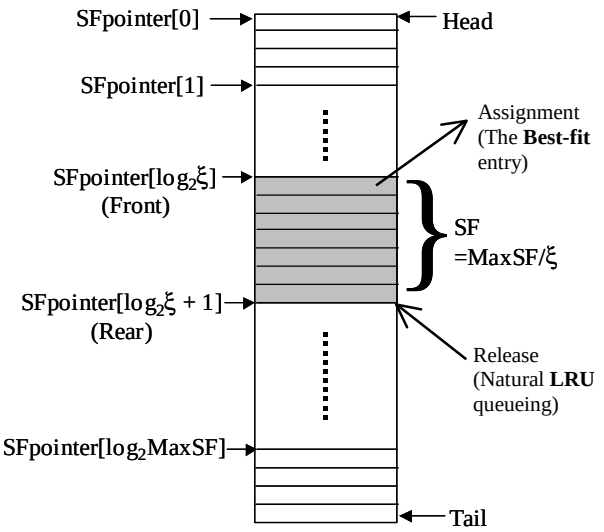


Figure 6: The unused OVSF code list organized by a logical double-linked list

A data rate R_i for call i is a multiplication of the basic transmission rate R_b, where 1 (spreading factor = 256) $\le R_i \le$ 64 (spreading factor = 4). Then calculate R_i's approximate single-code rate $\P(R_i) = 2^{\lfloor \log_2(2*R_i - 1) \rfloor}$. Subsequently, user admitted code can be represented by the prefix $MaxSF/\xi$ bits of the codeword indexed by SFpointer[$\log_2(\xi)$] if the index is not NULL and $MaxSF/\xi$ equals to the field SF. That is the best-fit strategy. Perhaps none of the unused codes has the same spreading factor to $MaxSF/\xi$, the code has to be split if the field SF is greater than $MaxSF/\xi$. Otherwise, the call is blocked if the pointer index is NULL. This best-fit strategy in the BLRU algorithm is the most straightforward one to select an unused code from the set of available codes. It always chooses the code with the best spreading factor first.

In static code assignment, assigning a code at random certainly results in a lot of fragmental codes. In dynamic code assignment, as described in [3], the optimal dynamic code assignment algorithm alternatively suffers from the additional overhead of code exchange and reassignment, although the overhead is minimized as much as possible. This overhead increases the SC dramatically, in a software and/or hardware.

Intuitively, a good approximation to the optimal solution is based on the observation that codes assigned first will have a higher probability to be released first. The popular least recently used (LRU) algorithm is considered to have an adequate performance. It tends to minimize the number of fragmental codes. In our BLRU algorithm, searching for a LRU code is not required since the corresponding SFpointer points to the front of the best-fit entries and each entry group with the same spreading factor is a natural LRU queue. Of course, timestamps or time counters are not required either. The brief flowchart of the BLRU algorithm is plotted in Figure 7 and the algorithm is shown as follows.

BLRU code assignment algorithm

Input: A request data rate R_i for call i
Output: An allocated code number N_i if call accepted or a Null code number if call blocked.
Step 1:
▷ Calculate the approximate single-code rate ξ which is slightly higher than R_i.
Formula: $\P(R_i) \leftarrow 2^{\lfloor \log_2(2*R_i - 1) \rfloor}$.
Step 2:
$MaxSF \leftarrow$ Maximum spreading factor
▷ Check if the code rate ξ is greater than the remaining available rates in system. If so, this call will be blocked.
If $\xi >$ ($MaxSF$ – UsedDataRates) then goto Step 8.
Step 3:
▷ Check if there exists a code with SF $\ge MaxSF/\xi$. If not, this call will be blocked.

If SFpointer[$\log_2 \xi$].SF = NULL then goto Step 8.

Step 4:

▷ Best-fit strategy

SFpointer[$\log_2 \xi$].CodeNo is the best-fit choice

If SFpointer[$\log_2 \xi$].SF = *MaxSF*/ξ then goto Step 6.

Step 5:

▷ Code Splitting

CSF ← SFpointer[$\log_2 \xi$].SF

while (CSF > *MaxSF*/ξ)

Split one code with spreading factor CSF into two codes with spreading factor CSF/2.

CSF ← CSF/2.

Modify SFpointer[$\log_2 \xi$] and some codes' fields PREV and NEXT.

Step 6:

▷ LRU code assignment

Retrieve the front element of the entry group with spreading factor *MaxSF*/ξ.

N_i ← SFpointer[$\log_2 \xi$].CodeNo

▷ On the contrary, the code will be inserted into the rear position of the entry group with spreading factor *MaxSF*/ξ as a code is released.

Step 7:

▷ Call accepted

Code No. ← N_i

goto Step 9

Step 8:

▷ Call blocked

Code No. ← NULL

Step 9:

Return

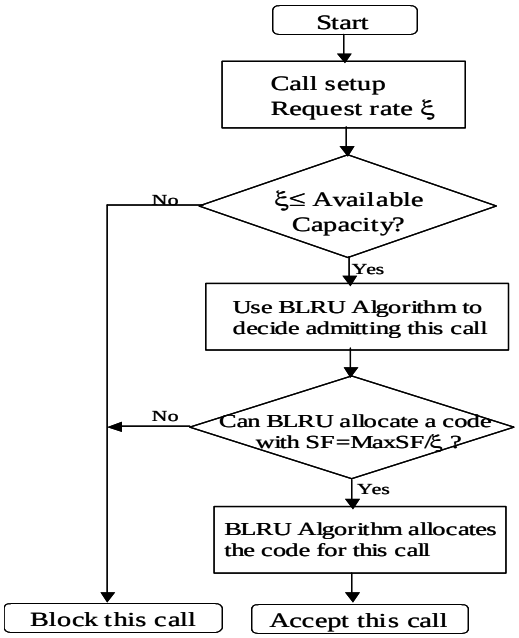


Figure 7: A brief flow chart of the BLRU algorithm.

IV. Complexity Analysis

To analyze the time complexity of the code assignment operation, all the steps in the BLRU algorithm are considered. Each step needs to execute only one time from Step 1 to Step 4. So do those steps from Step 7 to Step 9. In Step 6, the retrieve operation runs in O(1) time since at most some pointers must be modified for the unused code list. Only in Step 5, to split a code, the WHILE loop needs running k times, where k is no more than $\log_2(MaxSF)$. Therefore, the time complexity of the assignment part can be illustrated as

$$T(k) = k + O(1) \le \log_2(MaxSF) + O(1)$$
$$= 8 + O(1), \quad \triangleright \text{ if } MaxSF = 256$$
$$= O(1).$$

Similarity, with according to the code release operation of the BLRU algorithm, all steps take O(1) time except for the codes merging step. For codes merging, the WHILE loop has to run m times. Hence, the time complexity of the release part can be represented by

$$T(m) = m + O(1) \le \log_2(MaxSF) + O(1)$$
$$= 8 + O(1), \quad \triangleright \text{ if } MaxSF = 256$$
$$= O(1).$$

Totally, the BLRU algorithm takes only constant time in the worst case. Neither any dynamic code exchange and reassignment nor the code searching operation is required.

On the other hand, an OVSF code generation function – DWCGF -- is designed to generate the corresponding codeword, instead of storing a real codeword straight in an array. Given a code number (index) N_i, DWCGF $\delta(SF, \gamma)$ is an efficient and correct code generating function that can save about 89 percent of space. How the DWCGF works is described as follows.

The data rate R_i for call i can be assigned an approximate single-code rate ξ defined by

$$\rho(R_i) = 2^{\lfloor \log_2(2*R_i - 1) \rfloor}.$$

Assume that code number N_i, an index of a code entry, is assigned. Let us define that

$$\gamma = \frac{(N_i - 1)}{MaxSF/SF} + 1 = \frac{(N_i - 1)}{\rho} + 1 = \frac{N_i + \rho - 1}{\rho}$$

, where $1 \le \gamma \le SF$, $\gamma \in N$. Following the data-structure rule described as above, (N_i-1) must be the multiplication of ξ and the codes from N_i to $N_i+\xi$-1 will not be used. From the point of view on the code tree, the codes from N_i+1 to $N_i+\xi$-1 are the descendants of the code N_i. Simultaneously, the used flag associated with the code N_i turns to "true." If each code entry consists of a real codeword (256 bits) in

the code table, the prefix SF bits of the codeword are fetched directly. Alternatively, the DWCGF formula can derive the codeword with spreading factor SF in only O(1) running time. The DWCGF formula is shown by

$\delta(1, 1) = 0.$

$$\delta(SF, \#) = \delta(SF/2, \lceil \#/2 \rceil) * (\sqrt{2}^{SF} + 1) + [(\# - 1) \bmod 2] * [\sqrt{2}^{SF} - 1 - 2 * \delta(SF/2, \lceil \#/2 \rceil)].$$

Next, we translate the decimal value of $\delta(SF, \#)$ to a sequence of binary digits and retrieve the least significant SF bits. With digit 0 represents +1 and digit 1 represents –1, this maps the result to an OVSF code.

Examples of the OVSF code generation via DWCGF are shown in Table 1. The boundary condition of $\delta(SF, \#)$ is that $\delta(1, 1) = 0$. By DWCGF and the table index N_i, BS can obtain any OVSF codeword with at most $\log_2(MaxSF)$ times recursively.

Let n = spreading factor. The time complexity of function $\delta(SF, \#)$ can therefore be described by the recurrence

$T(n) = T(n/2) + O(1)$,

where $T(1) = O(1)$ is the boundary case.

By iteration method,

$T(n) \le T(MaxSF)$ ▷ in the worst case
$= T(MaxSF/2) + O(1)$
$= (T(MaxSF/4) + O(1)) + O(1)$
$= \dots$
$= T(1) + \log_2(MaxSF) * O(1)$
$= O(1) + \log_2(MaxSF) * O(1)$
$= 9 * O(1)$, ▷ if $MaxSF = 256$
$= O(1)$

Thus, the DWCGF function also performs only in constant time.

To analyze the space requirement, we do not require the real OVSF code tree and the unused code list. And, each code entry in Figure 4 without the codeword part requires only 33 bits when DWCGF is used. An implemented OVSF code table with 256 entries plus eleven additional pointers requires only about 1K bytes of memory. On the contrary, if DWCGF is absent, the code table requires about 9K bytes of memory. In brief, applying the function DWCGF can save further 89 percent of space. This alternative is especially feasible and suitable for the MSs with a very small memory size.

Table 1: Examples of OVSF code generation via DWCGF

SF	CGF	Walsh code		OVSF code	
		Value	SF bits	Code word	Code
1	δ(1,1)	0	0	(1)	$C_{1,1}$
2	δ(2,1)	0	00	(1,1)	$C_{2,1}$
	δ(2,2)	1	01	(1,-1)	$C_{2,2}$
4	δ(4,1)	0	0000	(1,1,1,1)	$C_{4,1}$
	δ(4,2)	3	0011	(1,1,-1,-1)	$C_{4,2}$
	δ(4,3)	5	0101	(1,-1,1,-1)	$C_{4,3}$
	δ(4,4)	6	0110	(1,-1,-1,1)	$C_{4,4}$
8	δ(8,1)	0	00000000	(1,1,1,1,1,1,1,1)	$C_{8,1}$
	δ(8,2)	15	00001111	(1,1,1,1,-1,-1,-1,-1)	$C_{8,2}$
	δ(8,3)	51	00110011	(1,1,-1,-1,1,1,-1,-1)	$C_{8,3}$
	δ(8,4)	60	00111100	(1,1,-1,-1,-1,-1,1,1)	$C_{8,4}$
	δ(8,5)	85	01010101	(1,-1,1,-1,1,-1,1,-1)	$C_{8,5}$
	δ(8,6)	90	01011010	(1,-1,1,-1,-1,1,-1,1)	$C_{8,6}$
	δ(8,7)	102	01100110	(1,-1,-1,1,1,-1,-1,1)	$C_{8,7}$
	δ(8,8)	105	01101001	(1,-1,-1,1,-1,1,1,-1)	$C_{8,8}$

V. Simulation Results

Three approaches, namely, the random, the BLRU, and the reassignment based optimal code assignment algorithms, are simulated here for comparison. With respect to the code-limited system, the total BP represents the performance of the channelization code assignment algorithm. In case of approximate single code, the BS will assign only one code whose rate is slightly higher than the requested. Request data rate is normalized to 2^n style. These simulations use the following parameters as in [3].

- Call arrival process is Poisson with mean arrival rate λ = 4-64 calls/unit time.
- Call duration is exponentially distributed with a mean value of $1/\mu = 0.25$ unit of time.
- Maximum spreading factor = 256.
- Capacity test: code-limited.

In addition,

- R_{max} denotes the maximum request data rate
- R_{mean} denotes the mean request data rate

The simulations employ two types of request rate-source models: a uniform distribution and an exponential one. The uniform simulation generates the request rate R_i distributed uniformly with the mean rate = R_{mean} and the maximum rate = R_{max} for call i. Figures 8 to 10 show the results with uniform rate-distribution, where R_{mean} and R_{max} is up to $32R_b$ and $64R_b$, respectively. The

mean arrival rate λ ranging from 4 to 64 calls per unit time is represented on the horizontal axis. And the average BP for a long period of time is indicated on the vertical axis. From the three figures, the optimal solution has the lowest BP almost always. The BLRU algorithm is close to the optimal solution; the random approach usually is the worst one. Figure 8 represents the typical case and the practical operational region. Generally, the BLRU algorithm is always superior to the random case.

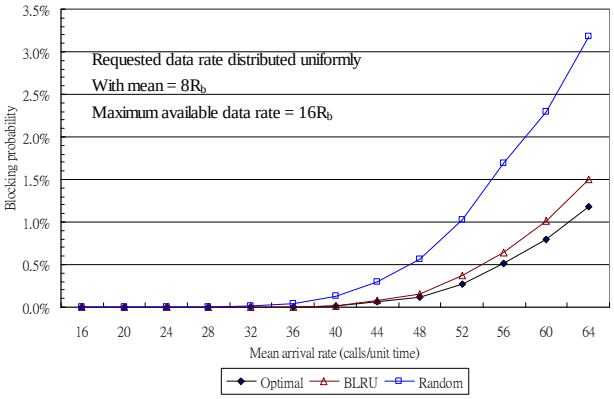


Figure 8: Comparison among three different algorithms using uniform rate distribution with $R_{mean} = 8R_b$ and $R_{max} = 16R_b$.

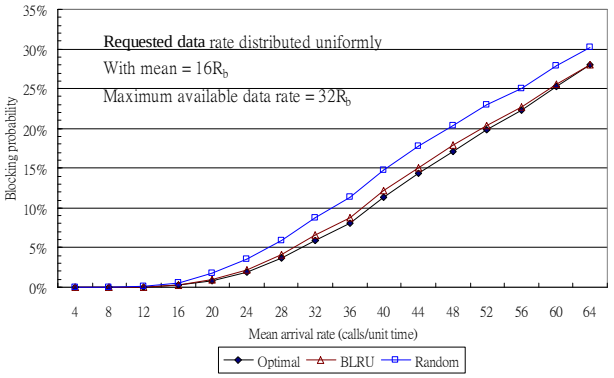


Figure 9: Comparison among three different algorithms using uniform rate distribution with $R_{mean} = 16R_b$ and $R_{max} = 32R_b$.

Figure 9 shows the case that $R_{mean} = 16R_b$ and $R_{max} = 32R_b$. As the maximum request data rate grows, the BPs of all approaches increase and the BLRU solution is getting closer to the optimal case. For the case shown in Figure 10, the three lines behave almost closely. The heavy traffic causes the system to move beyond the practical region. In practical region, the BP needs smaller than 5 to 10 percent generally. Consequently, we can conclude that the BLRU algorithm performs a well result. And, that the BLRU algorithm needs neither code exchange nor reassignment process is notably mentioned, although the optimal case slightly outperform it.

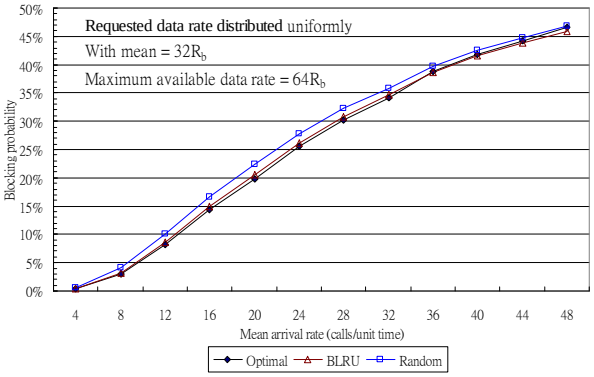


Figure 10: Comparison among three different algorithms using uniform rate distribution with $R_{mean} = 32R_b$ and $R_{max} = 64R_b$.

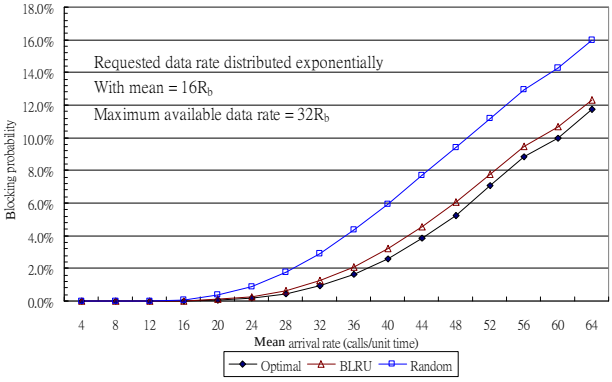


Figure 11: Comparison among three different algorithms using exponential rate distribution with $R_{mean} = 16R_b$ and $R_{max} = 32R_b$.

On the other hand, the exponential simulation generates the request rate R_i distributed exponentially for call i. The results in Figures 11 to 12 with exponential rate-distribution are similar to those described above, except for an interesting region. The interesting region is on the right upper corner in Figure 12, where the mean arrival rate ranges from 50 to 64. In the region, the BLRU algorithm outperforms the optimal case. The reason is that the fragmentation with a moderate number of fragmental codes has more total call accepted times in the BLRU case rather than those in the optimal case, when both the arrival rate and the request data rate are high enough. However, this is only a phenomenon but not in practical working area. High BP caused by high arrival rate and high request rate has to be decreased by increasing the system capacity, e.g., more micro-cells established. As a result, if we take

both the BP and the SC issues into account, the BLRU algorithm is a better approach.

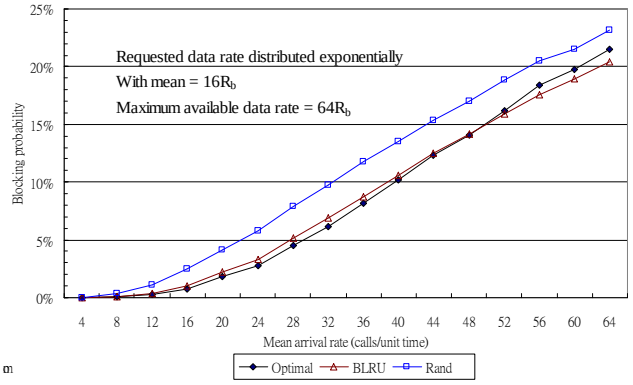


Figure 12: Comparison among three different algorithms using exponential rate distribution with R_{mean} = $16R_b$ and R_{max} = $64R_b$.

VI. Conclusions

Some OVSF code assignment methods for variable rate transmission have been investigated in the literature. Both single code and multicode assignment algorithms suffer from fragmentation problem, which degrades the performance of the code assignment.

In this paper, we proposed to use the Best-fit Least Recently Used (BLRU) algorithm to resolve the performance issue. In addition, two different approaches were simulated and compared to the BLRU algorithm in terms of Blocking Probability. As a consequence, considering both the Blocking Probability and System Complexity, , the BLRU algorithm is a very suitable algorithm for the channelization code assignment in 3G WCDMA.

References

1. A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communications*, Addison-Wesley, 1995.
2. R. Kohno, R. Meidan, and B. Milstain, "Spread spectrum access methods for wireless communications," *IEEE Commun. Mag.*, vol. 33, pp. 58-67, Jan. 1995.
3. T. Minn and K.-Y. Siu, "Dynamic Assignment of Orthogonal Variable-Spreading- Factor Codes in W-CDMA," *IEEE Journal on Selected Areas in Commun.*, vol. 18, no. 8, pp.1429-1440, August 2000.
4. 3GPP Technical Specification 25.213, v3.3.0, Spreading and modulation (FDD) (Released in 1999), June 2000.
5. I. Chih-Lin and R. D. Gitlin, "Multi-code CDMA Wireless Personal Communications Networks," in *Proc. of ICC'95*, pp. 1060-1064, June 1995.
6. T. Ottosson and T. Palenius, "The Impact of Using Multicode Transmission in the WCDMA System," in *Proc. of IEEE VTC'99*, vol. 2, pp. 1550-1554, May 1999.
7. F. Adachi, K. Ohno, A. Higashi, T. Dohi, and Y. Okumura, "Coherent Multicode DS-CDMA Mobile Radio Access," *IEICE Trans. Commun.*, vol.E79-B, no.9, pp.1316-1325, Sept. 1996.
8. Harri Holma and Antti Toskala, *WCDMA for UMTS*, John Wiley & Sons, 2000.
9. E. Dahlman and K. Jamal, "Wide-band Services in a DS-CDMA Based FPLMTS System," in *Proc. of the 46th IEEE VTC'96*, pp.1656-1660, Atlanta, April 1996.
10. S. Ramakrishna and J. M. Holtzman, "A Comparison between Single Code and Multiple Code Transmission Schemes in a CDMA System," in *Proc. of the 48th IEEE VTC'98*, pp.791-795, May 1998.
11. S. J. Lee, H. W. Lee, and D. K. Sung, "Capacities of Single-Code and Multicode DS-CDMA Systems Accommodating Multiclass Services," *IEEE Trans. On Vehicular Tech.*, vol. 48, no. 2, pp.376-384, March 1999.
12. P. Agin and F. Gourgue, "Comparison between Multicode with Fixed Spreading and Single Code with Variable Spreading Options in UTRA/TDD," in *Proc. of the 2nd IEEE SPAWC'99*, pp.325-328, May 1999.