Online OVSF Code Assignment with Resource Augmentation

Francis Y. L. Chin[1,⋆], Yong Zhang[1], and Hong Zhu[2,⋆⋆]

[1] Department of Computer Science, The University of Hong Kong, Hong Kong
{chin,yzhang}@cs.hku.hk
[2] Institute of Theoretical Computing, East China Normal University, China
hzhu@sei.ecnu.edu.cn

Abstract. Orthogonal Variable Spreading Factor (OVSF) code assignment is a fundamental problem in Wideband Code-Division Multiple-Access (W-CDMA) systems, which play an important role in third generation mobile communications. In the OVSF problem, codes must be assigned to incoming code requests, with different data rate requirements, in such a way that they are mutually orthogonal with respect to an OVSF code tree. An OVSF code tree is a complete binary tree in which each node represents a code associated with the combined bandwidths of its two children. To be mutually orthogonal, each leaf-to-root path must contain at most one assigned code. In this paper, we focus on the online version of the OVSF code assignment problem, in the often-studied context of the single cell as well as in the more general context of the whole multi-cell cellular network (for which there are no known results). With the help of 1/8 and 11/8 extra bandwidth resources, we are able to give a 5-competitive algorithm in the single cell and the multi-cell context respectively, which means that the competitive ratio is a constant and not a function of the height of the OVSF tree and thereby improving upon past results.

1 Introduction

Wideband Code-Division Multiple-Access (W-CDMA) technology is one of the main technologies widely-developed in recent years for the implementation of third-generation (3G) cellular systems. We consider the well-studied problem of Orthogonal Variable Spreading Factor (OVSF) code assignment in W-CDMA systems [5,6,9,10,12].

OVSF is an implementation of CDMA wherein, before each signal is transmitted, the spectrum is spread according to a unique code, which is derived from an OVSF code tree. An OVSF code tree is a complete binary tree. Users have requests for different data rates, and we accommodate these different requests by assigning codes at different levels of the OVSF code tree, with the root being at

⋆ This research was supported in part by Hong Kong RGC Grant HKU-7113/07E.
⋆⋆ This research was supported in part by National Natural Science Fund (grant no. 60496321).

the highest level and representing the entire bandwidth of the wireless system. The code at any node other than the root denotes the bandwidth half that of its parent in the tree. In any *legal assignment* in the code tree, no two assigned codes lie on a single path from the root to a leaf, i.e., any two assigned codes are *mutually orthogonal*. The subset of nodes in the code tree, which forms a legal assignment, is called a *code assignment* (CA). A node x is said to be *free* if there are no assigned nodes in every root-to-leaf path containing x, and thus making x an assigned node would still result in a legal assignment. For convenience, we use the words "code" and "node" interchangeably. Fig. 1 is an example of an OVSF code tree with the code assignment represented by the darkened nodes marked as c, d, e, g and i.

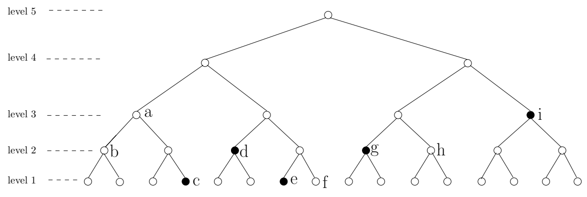

Fig. 1. An example of OVSF code tree, solid circles are the assigned codes

To illustrate the essence of the OVSF code assignment problem, consider the configuration shown in Fig. 1. Let $Req(x)$ denote the request to which code x is assigned, and let *reassign* x *to* y denote the reassignment of $Req(x)$ to code y and the freeing of code x (i.e. making x a free code). Suppose a level-2 code request arrives followed by a level-3 code request. If we assigned code b to the first request, we would have to make two code reassignments before we can assign code a to the second request, e.g. reassign b to h (and thereby freeing b) and reassign c to f (freeing c and consequently a). If, on the other hand, h were assigned to the first request, only one reassignment would be needed to satisfy the second request, i.e. reassign c to f (freeing c and consequently a), and then assign code a to the second request.

Since each reassignment requires processing overhead and may affect the quality of communications, a natural goal would be to minimize the number of reassignments. Note that this problem will not be too difficult and can be solved optimally by a greedy strategy if codes were never released. However, when codes can be released, the code tree can be fragmented so that many reassignments might be needed if a good assignment algorithm were not used.

In general, the algorithm for OVSF code assignment is expected to handle a sequence $\sigma = (C_1, C_2, \ldots, C_k, \ldots)$ of code operations over time, each operation C_k being either to *request* a code at a particular level or to *release* an assigned code. Note that, if the total bandwidth of any set of mutually orthogonal free codes is less than the bandwidth required by a code request, the new code request cannot be satisfied. In this paper, without loss of generality, we assume that all requests in σ can be satisfied (since we can easily check whether a new code request can be satisfied), and after each request is satisfied, a legal assignment results.

The OVSF code assignment problem is hard, and the approach has often been to produce heuristics, whose performance is measured by the approximation (or competitive) ratio, which compares the cost of the algorithm to the cost of optimal off-line scheme where cost is the total number of assignments or reassignments done by the algorithm.

The problem has been studied extensively in recent years, and we have the off-line and online versions of this problem.

- **Off-line CA Problem**
  Given a sequence σ of code operations, find a sequence of code assignments such that the total number of reassignments is minimum, assuming the initial code tree is empty. This problem was proved to be NP-hard by Marco Tomamichel [11], who also gave an exponential-time algorithm to solve it.
- **Online CA Problem**
  The operations in the sequence σ arrive through time. At any time $t > 0$, we only know about the operations until t and have no information about any future operation $C_{t'}$ with $t' > t$. The problem is to find a sequence of code assignments such that the total number of reassignments is minimum. Erlebach et al [5] gave an $O(h)$-competitive algorithm for this problem, where h is the height of the OVSF code tree. They also proved that the lower bound on the competitive ratio of this problem is 1.5. With resource augmentation [7], which means the online algorithm is allowed to use more bandwidth than the optimal scheme, a 4-competitive algorithm with a double-bandwidth code tree was given in [5] .

In this paper, we focus on the Online CA Problem. In Section 2, for the single-cell context, we give a new constant-competitive algorithm, using less extra bandwidth (less resource augmentation) to improve upon previous results. As far as we know, the Online CA Problem for multi-cell cellular networks has not been studied. In Section 3, we apply the techniques used in Section 2 in the context of the whole multi-cell cellular network to give a new constant-competitive algorithm.

2 Code Assignment in a Single Cell

Our online algorithm, Online-CA-Cell, makes use of extra resource in the form of an additional, albeit smaller, OVSF tree. Therefore, we talk about code assignments in the *main* OVSF tree and in a separate *extra* OVSF tree.

There are two properties that Online-CA-Cell seeks to maintain: (a) for any set of mutually orthogonal free codes in the main tree, there is at most one free code at each level; and (b) at each level of the extra tree, there is at most one assigned code. Fig. 2(a) shows an example of a main tree that have assigned codes, which are *sorted* and *compacted*. "Sorted" means that assigned codes are in non-decreasing order in terms of level (from left to right); and "compacted" means that, at each level, there is at most one free code. Fig. 2(b) gives an

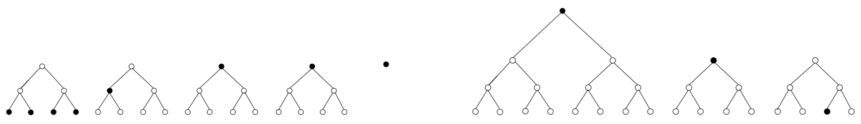

Fig. 2. Structure with assigned codes (shown as solid circles)

example of an extra tree. Online-CA-Cell is comprised of the method for handling each code request and the method for handling each code release.

When a level-i code request arrives, Online-CA-Cell first tries to satisfy the request in the main tree. Note that, we need only to consider assignment or reassignment at level i or higher in the main tree, since the total bandwidth of any set of mutually orthogonal free codes at lower levels is not large enough to satisfy a request for level i (because there is at most one free code at every level). If there is no free code at level i in the main tree, Online-CA-Cell will then try to satisfy the request by assigning a code from the extra tree, and if the extra tree already contains an assigned code at level i, Online-CA-Cell will reassign the leftmost assigned code at the lowest level $j > i$ in the main tree so as to free its offsprings in the main tree in order to accommodate the new request and the request of the assigned level-i code in the extra tree. The pseudo-code for code request is as follows:

```
**Code-Request**($R$,$i$)——*allocate a free code to satisfy the request $R$ of level $i$*
**if** the main tree has a free code at level $i$ **then**
    Assign that free code to $R$.
**else if** the extra tree contains no assigned node at level $i$ **then**
    Assign the rightmost free code at level $i$ in the extra tree to $R$.
**else**
    Let $w$ be the leftmost assigned code at lowest level $j > i$ in the main tree.
    Apply **Code-Request**($Req(w)$,$j$) so as to free up $w$ and its offspring codes.
    Assign the leftmost free code of level $i$ in the main tree to $R$.
    For those assigned codes in the extra tree from level $i$ to level $j-1$
        (if they exist) reassign them to the main tree.
**end if**
```

When a level-i code release of code x arrives, Online-CA-Cell might have to reassign the rightmost assigned code y at level i to x in order to maintain the compactness of the tree. The freeing up of y, however, could mean that there would be more than one free node at level i. So, part of the algorithm is to fix this up. The pseudo-code for code release is as follows:

```
**Code-Release**($x$,$i$)——*release code $x$ at level $i$*
Let $y$ be the rightmost assigned code at level $i$ (in either main or extra tree).
**if** $x = y$ **then** Free $x$
**else** Reassign $y$ to $x$, freeing up $y$.
```

```
end if
if there are two free codes at level $i$ of the main tree, they must be children of $z$ then Apply Code-Release($z$,$i+1$)
end if
```

Let the amortized cost of each code request be 3 credits and each code release be 2 credits because a code assignment/reassignment costs 1 credit, while freeing a code costs 0. Next, we will define the potential function f (Fig. 3). We shall show that the amortized cost of a code request or a code release can pay for the assignment/reassignment costs and the change of potentials.

The intuition behind the definition of potential function f is the following. The seven different cases shown in Fig. 3 exhaust all possible configurations of the assigned codes at level i. Configurations C_2, C_5 and C_7, which have 0, 1 and more than one assigned codes in the main tree, respectively, have an assigned code in the extra tree and have a potential value of 2 credits to compensate for the reassignment cost of bringing the assigned code in the extra tree back to the main tree if needed. Their corresponding configurations without assigned codes in the extra tree (i.e. C_1, C_4 and C_6) do not carry any potential credits. The remaining configuration C_3, which is the only configuration having an assigned code and a free code in the main tree (thus no assigned code in the extra tree), is associated with 1 potential credit to compensate for the cost of the reassignment of the rightmost code upon any code release at this level or lower levels.

Lemma 1. *Assume each code request is associated with 3 credits. The number of credits at each level of the main and extra tree, as defined by the potential function f as given in Fig. 3, will be maintained after each code request as described in Online-CA-Cell.*

Proof. According to Code-Request(R,i), when a request R arrives at level i:

if the main tree has a free code at level i: This configuration may be C_1 or C_3, whose potential value is 0 or 1. Note that C_1 can have one or no free codes. After the assignment, the configuration is changed to C_4 or C_6 and the remaining number of credits is either 2 or 3 (*code request cost* + *configuration potential* − *assignment cost*), which is larger than the potential values of C_4 and C_6.

else if the extra tree contains no assigned code at level i: This configuration may be C_1, C_4 or C_6, whose potential value is 0. After the assignment, the configuration is changed to C_2, C_5 or C_7 whose potential value is 2. Thus, the amortized cost of the code request can cover exactly the assignment cost and change of potential.

else (There is an assigned code in the extra tree.) The configuration may be C_2, C_5 or C_7 with potential value 2. After the assignment and Code-Request(*Req* (w),j), the configuration is changed to C_6 with potential value 0. Since we have to do two assignment/reassignments in this level, 3 credits will be left behind to cover the reassignment costs at higher levels, i.e. Code-Request(*Req*(w),j). Note that, for those levels between i and $j-1$ whose

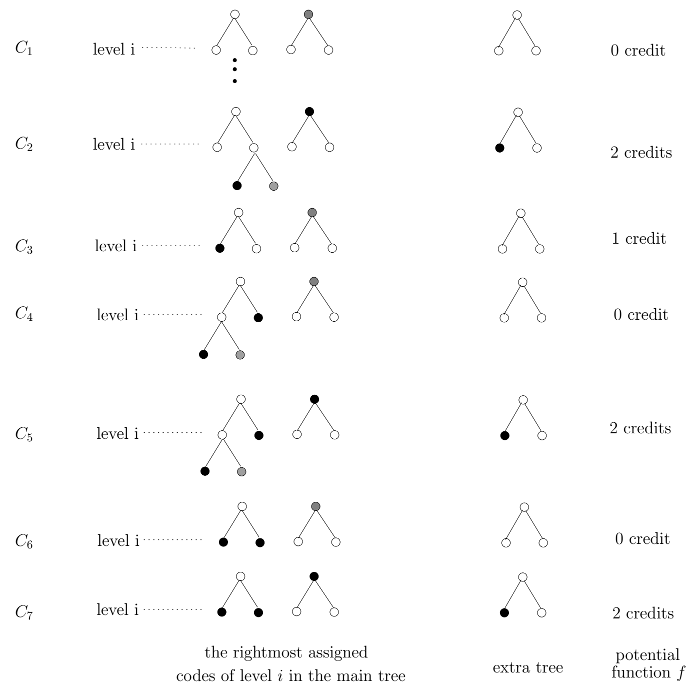

Fig. 3. Potential function of each level of the code tree, solid circle denotes an assigned code, empty circle denotes a free code, gray circle may or may not be an assigned code

assigned codes are reassigned to the main tree, the reassignment costs can be covered by the change of potential. □

Lemma 2. *Assume each code release is associated with 2 credits. The number of credits at each level of the main and extra tree as defined by the potential function f as given in Fig. 3 will be maintained after each code release as described in Online-CA-Cell.*

Proof. According to Code-Release(x, i), when a code x of level i is released:

Reassign y to x where y is the rightmost assigned code at level i: This costs one credit if $y \neq x$ exists.

After the assignment, the configuration cannot be C_2, C_5 nor C_7.

If there is at most one free code at level i of the main tree: The configuration must be C_1, C_4, C_3 or C_6 derived respectively from C_4, C_5, C_6 or C_7. We can easily check that the 2 credits associated with the code release operation can cover one reassignment cost and the increase of potential value of C_3 (no increase of potential value for C_1, C_4 and C_6).

If there are two free codes at level i of the main tree: After Code-Release(z,$i+1$), the final configuration may be C_1, C_4 or C_6 with potential value 0, respectively derived from the initial configuration C_4, (C_3 or C_5), or

(C_3 or C_7). Since the code release operation has 2 credits and freeing a code costs nothing, the remaining credits in level i is at least 2 (from the change of potential), which can cover the reassignments at level $i+1$, i.e. Code-Release($z, i+1$). □

Theorem 1. *Given a code tree T and extra tree T', which contains half the bandwidth of T, Algorithm Online-CA-Cell is 5-competitive.*

Proof. Since the extra tree can contain at most one code at all levels from 1 to $h-2$, the total bandwidth of the extra tree is at most half of T. Note that an extra code at level $h-1$ is not possible because that would imply the total bandwidth of all assigned codes is larger than the bandwidth of T.

Suppose there are m_1 code requests and m_2 code releases in the sequence. From Lemmas 1 and 2, we can see that, at any step, the credit of each level is at least 0. Thus, the total cost of Online-CA-Cell is at most $3m_1 + 2m_2 \leq 5m_1$ since the number of releases is at most the number of requests. The optimal cost is at least m_1, and so, the competitive ratio of Online-CA-Cell is at most 5. □

With the observation that the algorithm is k-competitive without any extra resource if each code request/release involves only a fixed number k of levels, we have a new 5-competitive algorithm with a extra code tree of 1/8 full bandwidth. We partition the main code tree into a *lower* part and an *upper* part. The lower part contains all the assigned codes from level 1 to $h-4$ and its configuration is similar to the main tree and extra tree as described before. The extra tree contains codes from levels 1 to $h-4$, and thus, the total bandwidth of the extra tree is 1/8 of the code tree. The upper part contains the assigned codes from level $h-3$ to $h-1$ which are *sorted* and *compacted*. Between the lower part and upper part, there is no free code of level $h-3$. The potential of configurations at level i ($1 \leq i \leq h-4$) is the same as the potential function defined in Fig. 3, and each code request and code release is associated with 3 and 2 credits respectively. The algorithm for code request/release is the same as before for levels 1 to $h-4$ and keeps the assigned codes sorted and compacted for levels $h-3$ to $h-1$.

When a code request (or release) arrives at level i ($1 \leq i \leq h-4$), if the algorithm does not affect upper level $h-3$, we can say the competitive ratio in this case is at most 5; otherwise, the algorithm in the lower part will give 3 (2 for release) credits to the upper part, which is enough to cover the cost of reassignments in the three levels from $h-3$ to $h-1$ (in the case of release, the two levels $h-3$ and $h-2$ since reassignment at level $h-1$ is not possible). So, we can also say that the competitive ratio in this case is at most 5.

When a code request or release happens at level i ($h-3 \leq i \leq h$), since the process does not affect the lower part, the competitive ratio in this case is at most 3. Thus, we have the following result.

Theorem 2. *Given a code tree T and extra tree T' which contains 1/8 the bandwidth of T, we can have a 5-competitive algorithm.*

3 Code Assignment in Cellular Networks

The geographic area of a mobile communications network is usually divided into many small cellular regions and call requests may be made at any cell. The code assignment in a single cell can be considered as a special case of code assignment in multi-cell cellular networks. In this section, we study the online OVSF code assignment problem in multi-cell cellular networks.

In cellular networks, each cell contains a base station, which communicates with other base stations via a high-speed wired network. Communications between any two users (even within the same cell) must be established through base stations. When a call request arrives, the nearest base station must assign a code, whose bandwidth matches with the bandwidth of the call request and which is orthogonal to all the assigned codes in the OVSF code tree of its own cell and its neighboring cells in order to avoid interference.

Communications in cellular networks have been widely studied [1,2,3,4,8], but mainly focus on Frequency Division Multiplexing (FDM) networks, which are the backbone for 1G/2G wireless communications. This is the first study on OVSF code assignment in cellular network. We will give a constant-competitive algorithm for OVSF code assignment in cellular networks with resource augmentation [6].

The cellular network can be 3-colored with $\{R, G, B\}$ so that each cell has one color and neighboring cells have different colors. Each cell with a different color can use a different code tree for handling code requests and releases initiated at that cell. In order to achieve a constant-competitive ratio, a total of six code trees with full bandwidth will be needed using the resource augmentation approach given in [5], whereas 3.375 code trees with full bandwidth would be sufficient using the approach in Section 2. However, we will introduce here an algorithm with a competitive ratio of 5 that uses only $19/8 = 2.375$ code trees of full bandwidth. Fig. 4 depicts three code trees T_1 , T_2 and T_3, with the heights of T_1

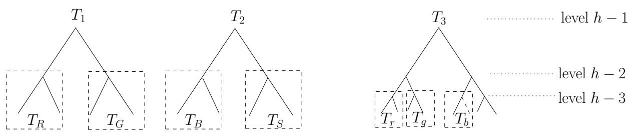

Fig. 4. Three code tree used by Online-CA-Network

and T_2 being h and the height of T_3 being $h-2$. The off-line optimal algorithm uses only one code tree with height h. Let the left and right subtree of T_1 be T_R and T_G, and those of T_2 be T_B and T_S, and let the three equal-bandwidth subtrees with height $h-4$ of T_3 be T_r, T_g and T_b.

The cell with color X will use T_X as its main tree and T_x as its extra tree. T_S will be shared by all cells. We will show in Theorem 3 that conflicts from different cells can never occur.

Online algorithm **Online-CA-Network** can be described as follows:

– For cells colored X ($X \in \{R, G, B\}$), perform Online-CA-Cell on T_X and T_x as the main and extra trees for all call operations in the cell colored X. In case the bandwidth of T_X is not enough, use T_S and T_x for those calls that cannot be accommodated by T_X.
– When a code request asks for the bandwidth of the whole code tree, assign the root of T_1, since there will be no other assigned code in this cell and its neighboring cells.

Theorem 3. *Online-CA-Network is a 5-competitive algorithm for the online OVSF code assignment problem with augmentation ratio of 19/8.*

Proof. As all neighboring cells use different code trees, there will be no interference as long as all the assigned codes in each cell are mutually orthogonal. Since Online-CA-Cell is applied to handle code operations in each cell, orthogonal assigned codes are ensured if Online-CA-Cell works properly with T_X and T_S, which is trivially true if T_S is not used by any other cells. Suppose T_S contains codes at level l assigned to call requests from a cell X. The total bandwidth of these codes must be greater than the total bandwidth of the free codes in T_X (the main tree is "sorted" and "compacted"). Since the bandwidth of T_X is $\mathcal{B}/2$, assuming that the total bandwidth of the code tree is $\mathcal{B}$, the total bandwidth of the assigned codes in the cell X is greater than $\mathcal{B}/2$. Since the off-line optimal algorithm uses only one code tree, the total bandwidth of neighboring cells is no more than $\mathcal{B}$. So the total bandwidth in each cell neighboring with X must be strictly less than $\mathcal{B}/2$. Therefore, we can ensure that T_S is used by at most one cell at any time, which means that the assigned codes in the main trees in cellular network do not affect each other.

As for each extra tree in T_3, the bandwidth is 1/8 of T_1 (T_X and T_S), i.e., $\mathcal{B}/8$. From Section 2, we know this is sufficient to handle all the extra codes, and the three subtrees of T_3 in the whole cellular network do not affect each other.

Since the analysis and code operations are similar to that in Online-CA-Cell, the competitive ratio of Online-CA-Network in the cellular network should be the same as the competitive ratio of Online-CA-Cell, which is 5-competitive. □

The following result applies in the special case where all code requests are at the same level.

Theorem 4. *There exists a 2-competitive algorithm with augmentation ratio of 2 that solves the code assignment problem when all requests are at the same level.*

Proof. We use two code trees T_1 and T_2. Similar to Fig. 4, let the two subtrees of T_1 be T_R and T_G, and let those of T_2 be T_B and T_S. Since all the requested bandwidth are the same, the extra trees as given in T_3 may not be needed.

Suppose there are m_1 code requests and m_2 code releases in a given sequence of code operations. Since each code request has 1 credit to pay for the assignment, and each code release has 1 credit to pay for the reassignment of the rightmost assigned code, the cost of our scheme is at most $m_1 + m_2 \leq 2m_1$ since the number of releases is no more than the number of requests. Note that the cost of the optimal off-line algorithm is at least m_1, and so we can say that our scheme is 2-competitive. □

Acknowledgements. The authors thank Dr. Bethany M. Y. Chan for her efforts in making this paper more readable.

References

1. W. T. Chan, F. Y. L. Chin, D. Ye, Y. Zhang and H. Zhu. Frequency Allocation Problem for Linear Cellular Networks. In Proc. of the 17th Annual International Symposium on Algorithms and Computation (ISAAC 2006), LNCS 4288, 61-70.
2. W. T. Chan, F. Y. L. Chin, D. Ye, Y. Zhang and H. Zhu. Greedy Online Frequency Allocation in Cellular Networks. *Information Processing Letters*, 102(2007), 55-61.
3. W. T. Chan, F. Y. L. Chin, D. Ye and Y. Zhang. Online Frequency Allocation in Cellular Networks. To appear in Proc. of the 19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '07).
4. I. Caragiannis, C. Kaklamanis, and E. Papaioannou. Efficient on-line frequency allocation and call control in cellular networks. *Theory Comput. Syst.*, 35(5):521–543, 2002. A preliminary version of the paper appeared in SPAA 2000.
5. T. Erlebach, R. Jacob, M. Mihalak, M. Nunkesser, G. Szabo and P. Widmayer. An algorithmic view on OVSF code assignment. In proc. of 21th Symposium on Theoretical Aspects of Computer Science (STACS 2004), LNCS 2996, pp. 270-281.
6. Xiang-Yang Li and Peng-Jun Wan. Theoretically Good Distributed CDMA/OVSF Code Assignment for Wireless Ad Hoc Networks. In Proc. the 11th Annual International Conference of Computing and Combinatorics (COCOON05), pp. 126-135.
7. B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. In Proceedings of the 36th IEEE Symposium on Foundations of Computer Science, pages 214-221, 1995.
8. C. McDiarmid and B. A. Reed. Channel assignment and weighted coloring. *Networks*, 36(2):114-117, 2000.
9. T. Minn and K. Y. Siu. Dynamic assignment of orthogonal variable-spreadingfactor codes in W-CDMA. IEEE Journal on Selected Areas in Communications, 18(8):1429-1440, 2000.
10. Angelos N. Rouskas, Dimitrios N. Skoutas. OVSF codes assignemnt and reassignment at the forward link OFW-CDMA 3G systems. In Proc. of the 13 th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2002.
11. Marco Tomamichel. Algorithmische Aspekte von OVSF Code Assignment mit Schwerpunkt auf Offline Code Assignment. menuscript.
12. Peng-Jun Wan, Xiang-Yang Li and Ophir Frieder. OVSF-CDMA Code Assignment in Wireless Ad Hoc Networks. In Proc. of DIAL M-POMC 2004 joint workshop on Foundations of mobile computing, pp. 92-101, 2004.