國 立 交 通 大 學

電機與控制工程學系

博 士 論 文

以序的最佳化理論為基礎的多重使用者正交分頻多工系統的適應性次載波指定與位元分配方法

Adaptive Subcarrier Assignment and Bit Allocation Methods for Multiuser OFDM System Using Ordinal Optimization Approach

研 究 生：黃榮壽

指導教授：林心宇　教授

中 華 民 國 九 十 七 年 十 月

以序的最佳化理論為基礎的多重使用者正交分頻多工系統的適應性次載波指定與位元分配方法

Adaptive Subcarrier Assignment and Bit Allocation Methods for Multiuser OFDM System Using Ordinal Optimization Approach

研 究 生：	黃 榮 壽	Student:	Jung-Shou Huang
指導教授：	林 心 宇	Advisor:	Shin-Yeu Lin

國 立 交 通 大 學
電機與控制工程學系
博 士 論 文

A Dissertation
Submitted to Department of Electrical and Control Engineering
College of Electrical Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in

Electrical and Control Engineering

October 2008

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 七 年 十 月

以序的最佳化理論為基礎的多重使用者正交分頻多工系統的適應性次載波指定與位元分配方法

研究生：黃榮壽　　　　　指導教授：林心宇　博士

國立交通大學電機與控制工程學系

摘要

下一世代的無線通信系統被期望能夠在無須考慮使用者移動情況與使用者所在位置的情況下，提供高的資料傳輸率的能力，以符合提供數位影音廣播與無線網際網路等資料存取服務的需求。要達到此目的會面臨的主要挑戰有幾項：(1)惡劣的無線通道環境，(2)服務品質，例如資料錯誤率與使用者的資料需求量，(3)通信系統中資源的配置，例如功率消耗與頻帶資源的管理問題，(4)因為使用者的可移動特性所造成的通信系統狀態變化。最近，正交分頻多工系統受到熱切的關注且被認為適合被用於下一世代無線通信系統之中，因為它能夠有效率地傳輸資料，同時能夠抵抗各種傳播通道破壞並克服符間干擾問題。這個需求趨使我們去發展聰明且有效的資源管理方法，使得在滿足多重使用者的服務品質需求下，能夠即時的調配整個無線通信系統的功率與頻譜配置，以達到通信系統整體最有效率的運用。因此在本論文中，我們將提出兩個以序的最佳化理論為基礎的多重使用者正交分頻多工系統的適應性次載波指定與位元分配方法。

第一個方法採用序的最佳化理論為基礎的四層次演算法來求解一個足夠好的解。在前三個層次中，我們採用較粗略的模型來從候選的解空間中篩選出一些近似足夠好的可行解，直到選出 l (=3)個足夠好的解為止。之後的第四個層次中，採用精確的模型來從這 l 個足夠好的解之中篩選出最佳的那一個當做我們所要尋找的足夠好的解。這個四層次演算法保證能夠獲得一個足夠好的解，但是其中的第一層次耗費很多的計算時間於求解連續性問題的最佳解，因此我們提出可以採用硬體電路為輔助的方法與架構，利用深次微米的技術優勢來快速計算出連續性問題最佳解。

在大維度多重使用者正交分頻多工系統問題之中，採用硬體電路來實現第一層次時會有晶片面積太大的問題，因此在我們提出的第二個以序的最佳化理論為基礎的演算法中，我們採用一個近似目標函數值的計算模型，配合遺傳演算法來從可能的解空間中，快速地搜尋出 l (=200)個足夠好的解。

經由數值模擬以及與其他現行的方法做比較的結果可知，我們所提出的兩個方法在解的優質性與計算效率上都很成功，除了有效改善通信系統的功率效益之外，其中的一個方法可藉由硬體電路的輔助來達到即時應用之需求，而第二個方法很適合用於大維度多重使用者正交分頻多工系統問題之中。

Adaptive Subcarrier Assignment and Bit Allocation Methods for Multiuser OFDM System Using Ordinal Optimization Approach

Student: Jung-Shou Huang Advisor: Dr. Shin-Yeu Lin

Department of Electrical and Control Engineering
National Chiao-Tung University

Abstract

The next generation wireless communication systems are expected to provide high rate transmission in the applications of digital audio broadcast, digital video broadcast and wireless internet access but regardless to the users' mobility and location. The major challenges we are confronted with include the harsh channel conditions, QoS (Quality of Services) requirements such as BER (Bit Error Rate) and users' data rate request, scarce resources such as power and spectrum, and the knowledge of the most updated state of the mobile users or devices. Orthogonal frequency-division multiplexing (OFDM) technology is recently recognized as one of the leading candidates for supporting the next generation wireless communication systems due to its ability to combat inter-symbol-interference (ISI) over harsh channel conditions. This stimulates the development of both intelligent and efficient resource management algorithms to achieve efficient utilization of power and spectrum while providing QoS requirements in the multiuser OFDM communication system. Therefore in this dissertation, we will present two computationally efficient methods to solve the Adaptive Subcarrier Assignment and Bit Allocation (ASABA) problem of multiuser OFDM system using Ordinal Optimization (OO) approach.

Our first method consists of four OO stages to find a good enough solution to the ASABA problem. In the first three stages, we use surrogate models to select a subset of *estimated good enough* feasible solutions from the candidate solution set so as to reduce the search space of subcarrier assignment until l (=3) good enough subcarrier assignment patterns are obtained. Then in the fourth stage, we use the exact objective function to evaluate the l subcarrier assignment patterns, and the best one associated with the corresponding optimal bit allocation is the good enough solution that we seek. The four-stage OO approach ensures the quality of the obtained solution, however at the cost of solving a continuous version of the considered problem in the first stage. To resolve this computational complexity problem, we propose a hardware implementable Dual Projected Gradient (DPG) method to exploit deep submicron technology so as to obtain the optimal continuous solution extremely fast.

Due to the large dimension of the ASABA problem, implementing the first stage in hardware is almost impossible for area concern. Therefore in the first stage of our second method, we develop an approximate objective function to evaluate the performance of a subcarrier assignment pattern and use a genetic algorithm to efficiently search through the huge solution space to find I (=200) good solutions.

Numerical results and comparisons with various existing algorithms are provided to demonstrate the potential of the proposed techniques. It is shown that the proposed resource allocation methods substantially improve the system's power efficiencies and are more computationally efficient. Moreover, the first method can meet the real-time application requirement and the second method is suitable for large dimensional ASABA problems.

誌　謝

首先要感謝我的指導教授林心宇博士在論文研究上給予的細心指導及鼓勵，且提供優良的研究環境，使我能順利完成博士學位。林心宇教授學問淵博、涵養深厚，在學術研究上見解精闢、邏輯縝密，研究認真精神與嚴謹程度實在值得效法。對於本論文的完成，除了林教授的指導外，也要非常感謝諸位口試委員在百忙之中能抽空指導並給予寶貴的意見，使得本論文能更臻完善。

其次，要感謝我的學長林啟新博士、林謝興博士與洪士程博士，實驗室的學弟張紹興、林成梓與詹庭瑜。多年來，因為有你們的參與，使我在交通大學的求學過程之中，添增了一些多采多姿的回憶。

同時也要感謝我的服務單位，義隆電子公司的長官與同事們多年來的協助，使我得以順利完成此論文。

最後，感謝作者的父母與妻兒，在求學期間的支持與陪伴，讓我可以全心全力的專注於論文研究上。僅以此論文獻給我的家人與關心我的師長與朋友們。

Contents

List of Tables

List of Figures

Chapter 1
Introduction

1.1 Motivation

Due to the increase of mobile users and devices in the wireless communication system, various resource management techniques such as the dynamic channel allocation [1] and the dynamic fair resource allocation scheme [2] had been studied. One of the difficulties for such kind of techniques is to keep track of the most updated state of the mobile users or devices caused by their mobility and portability and provide them the appropriate resources. Therefore, the computational efficiency is the premise of the wireless network resource management methods so as to deal with the high mobility of the dynamic behaviors of mobile users and devices. Among the existing dynamic resource management problem in wireless networks, Adaptive Subcarrier Assignment and Bit Allocation (ASABA) of Orthogonal Frequency Division Multiplexing (OFDM) system is a very fundamental issue in mobile communication. There are two types of formulations on this issue. One is the Margin Adaptive (MA) optimization, which minimizes the total consumed power under a data rate constraint [3], and the other is the Rate Adaptive (RA) optimization, which maximizes the data rate under a power constraint [4]. Kim *et al.* had shown in [5] that the RA optimization problem can be solved via recursive MA optimization. Therefore, in this dissertation we will focus on the adaptive subcarrier assignment and bit allocation problem of MA optimization with emphasis on the solution quality and the computational efficiency.

In general, to obtain a better solution of a *hard optimization problem* such as the resource management problem in the wireless communication system usually requires a sophisticated but computationally intensive algorithm. In this dissertation, we will revolt this seemingly correct argument by proposing two methods that will not only obtain a *good enough feasible solution* but also be computationally efficient. The first method can meet the real-time application requirement while with the assistance of hardware. The second method is purely a software but still computationally efficient for solving the large dimension ASABA problem of multiuser OFDM system.

1.2 Problem Statement

The adaptive subcarrier assignment and bit allocation of multiuser OFDM system has been studied for a number of years. This issue is initialized by Wong *et al.* in [3] and is formulated as a nonlinear integer programming problem to minimize the total power consumption while satisfying the users' data communication request and system's constraints. Wong *et al.* employed a Lagrangian relaxation method in [3] to solve the continuous version of the adaptive subcarrier assignment and bit allocation problem. They then rounded the optimal continuous subcarrier assignment solution off to the closest integer solution. Although the algorithm dramatically enhances the power efficiency, the prohibitively high computational complexity renders it impractical. Since then, various methods, ranging from the more computation-time consuming and global-like mathematical programming based approach [5] to the less computation-time consuming and more local-like schemes [6]-[9] were proposed to cope with this NP-hard *constrained combinatorial* optimization problem. In [5], Kim *et al.* had converted the adaptive subcarrier assignment and bit allocation problem formulated in [3] into a linear integer programming problem and employed a suboptimal approach to separately perform the subcarrier assignment and bit allocation. To claim for computational efficiency by not using mathematical programming approach, Ergen *et al.* had proposed in [6] a heuristic two-module scheme, both Kivanc *et al.* and Zhang had proposed two-step subcarrier assignment approaches in [7] and [8], respectively, and Han *et al.* had proposed in [9] an iterative grouping scheme to improve the performance by exchanging subcarrier assignment sets. As a consequence, these approaches cannot yield a better solution while using limited computation time due to the nature of nonlinear optimization.

1.3 Dissertation Outline

The dissertation introduces two computationally efficient methods to solve the ASABA problem of multiuser OFDM system using Ordinal Optimization (OO) approach. Since both methods are multi-stage OO based approaches, there are some overlap. For the sake of completeness in presenting each individual method, the overlapping part will be duplicated. Therefore, we organize this dissertation in the following manner.

In chapter 2, some preliminaries are stated to assist the presentation of the dissertation.

These include OFDM architecture based communication system, adaptive subcarrier assignment and bit allocation for multiuser OFDM, OO theory, Artificial Neural Network (ANN), and Genetic Algorithm (GA). The OFDM is a promising technology for high data rate transmission in wide band wireless systems due to its ability to mitigate the effects of frequency selective and combat Inter-Symbol Interference (ISI). The adaptive subcarrier assignment and bit allocation for multiuser OFDM communication system can be formulated as a nonlinear integer programming problem to minimize the total power consumption while satisfying the users' data communication request and system's constraints. The OO theory is a new methodology designed to cope with hard optimization problems such as the considered problem. The GA acts as a heuristic method to select a representative set for the search space of the considered problem. The ANN is trained as an easily computed crude model to roughly evaluate the performance of the considered problem.

In chapter 3, we present an OO theory based four-stage approach to deal with the subcarrier assignment and bit allocation problem of multiuser OFDM system. The four-stage OO approach ensures the quality of the obtained solution, however at the cost of solving a continuous version of the considered problem in the first stage. To resolve this computational complexity problem, we propose a hardware implementable Dual Projected Gradient (DPG) method to exploit deep submicron technology. Therefore, our approach can meet the real-time application requirement through the assistance of hardware.

In chapter 4, we present another OO theory based three-stage approach to deal with the *large-dimension* ASABA problem of multiuser OFDM system. First of all, we reformulate the considered problem to separate it into subcarrier assignment and bit allocation problem such that the objective function of a feasible subcarrier assignment pattern is the corresponding optimal bit allocation for minimizing the total consumed power. Then in the first stage, we develop an approximate objective function to evaluate the performance of a subcarrier assignment pattern and use a genetic algorithm to search through the huge solution space and select s best subcarrier assignment patterns based on the approximate objective values. In the second stage, we employ an off-line trained ANN to estimate the objective values of the s subcarrier assignment patterns obtained in stage 1 and select the l best patterns. In the third stage, we use the exact objective function to evaluate the *l* subcarrier assignment patterns obtained in stage 2, and the best one associated with the corresponding optimal bit allocation is the good enough solution that we seek.

Some conclusions for the dissertation are drawn in Chapter 5. We also suggest some possible future research issues concerning the methods developed in this dissertation.

Chapter 2

Preliminaries

2.1 Multiuser Orthogonal Frequency Division Multiplexing System

The basic idea of OFDM is to divide the available spectrum into several subcarriers so that the information symbols are transmitted in parallel on the subcarriers over the wireless channel. This allows us to design a system as shown in Figure 2.1 to support high data rates transmission.

We assume that the system has K users to share N subcarriers. Each user's data rate request will be allocated to a nonoverlapping set of subcarriers and distributed among them. The allocating period in this model is a time interval consisting of several OFDM symbols and is assumed to be short enough so that users' channel gains will stay approximately constant. It is also assumed that a subcarrier cannot be shared by more than one user.

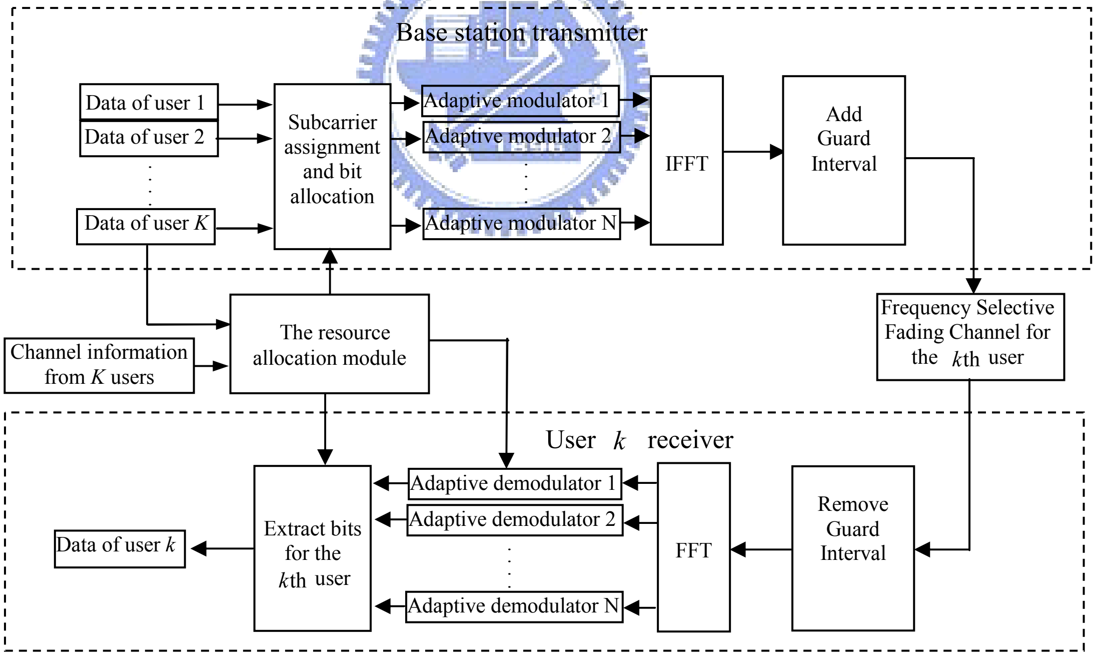


Figure 2.1. Block diagram of a multiuser OFDM system with subcarrier assignment and bit allocation.

In the transmitter part of Figure 2.1, the serial data from K users are fed into the block represented by the proposed adaptive subcarrier assignment and bit allocation algorithm. The algorithm will be executed in every allocating period to assign the set of subcarriers to each user and the number of bits to be transmitted on each assigned subcarrier based on the updated channel information of all users. For each subcarrier, the adaptive modulator will apply a proper modulation scheme to each symbol depending on the number of bits assigned to the subcarrier, and the modulated symbols are transformed into time domain samples by an Inverse Fast Fourier Transform (IFFT) as indicated in Figure 2.1. The guard interval is then added to ensure orthogonality between the subcarriers provided that the maximum time dispersion is less than the guard interval. Finally, the transmitted signals pass through different frequency selective fading channels to different users.

We assume the subcarrier assignment and bit allocation information is sent to the receivers via a separate control channel. For the sake of simplicity, we only show the receiver part of one user in Figure 2.1. At the *k*th user's receiver part, the guard interval is removed to eliminate the ISI, and the time sample of the *k*th user is transformed into modulated symbols using the Fast Fourier Transform (FFT). The modulation information is then used to configure the demodulators while the subcarrier assignment information is used to extract the demodulated bits from the subcarriers assigned to the *k*th user.

2.2 Adaptive Subcarrier Assignment and Bit Allocation (ASABA) Problem

In OFDM communication system, the power level needed for transmitting c bits from transmitter to user k receiver using subcarrier n is $\frac{f_k(c)}{\alpha_{k,n}^2}$, where $f_k(c)$ denotes the required transmission power for c bits of user k when the channel gain is equal to unity and $\alpha_{k,n}$ denotes the magnitude of the channel gain of the nth subcarrier as seen by the kth user. Just as an example, we assume that M-ary QAM is used in the communication system, then the required power $f_k(c)$ in $\frac{f_k(c)}{\alpha_{k,n}^2}$ to transmit c bits/symbol can be derived from [3][1]:

$$f_k(c) = \frac{N_0}{3}[Q^{-1}(\frac{P_e}{4})]^2(2^c - 1) \tag{2.1}$$

where $Q^{-1}(x)$ is the inverse function of

[1] The formula (2.1) is directly borrowed from [3, Sec. V, p.1725], which is an approximation based on the bit-error probability, $4Q(\sqrt{d^2/(2N_0)})$, and the average energy, $(M-1)d^2/6$, of a MQAM symbol, where d is the minimum distance between the points in the signal constellation.

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} e^{\frac{-t^2}{2}} dt \qquad (2.2),$$

N_0 denotes the noise Power Spectral Density (PSD) level, and P_e denotes the BER.

In general, wireless link capacity is generally a scarce resource that needs to be used efficiently. The channel-gain conditions of wireless links between transmitter and mobile users vary in the time domain, and different subcarriers experience different channel gains. Therefore, the subcarriers which appear in deep fade to one user may not be in deep fade for other users. A typical example of channel gains, $\alpha_{k,n}$'s, in $\frac{f_k(c)}{\alpha_{k,n}^2}$ can be shown in Figure 2.2, and a lower magnitude of the channel gain represents a deeper fade channel condition.

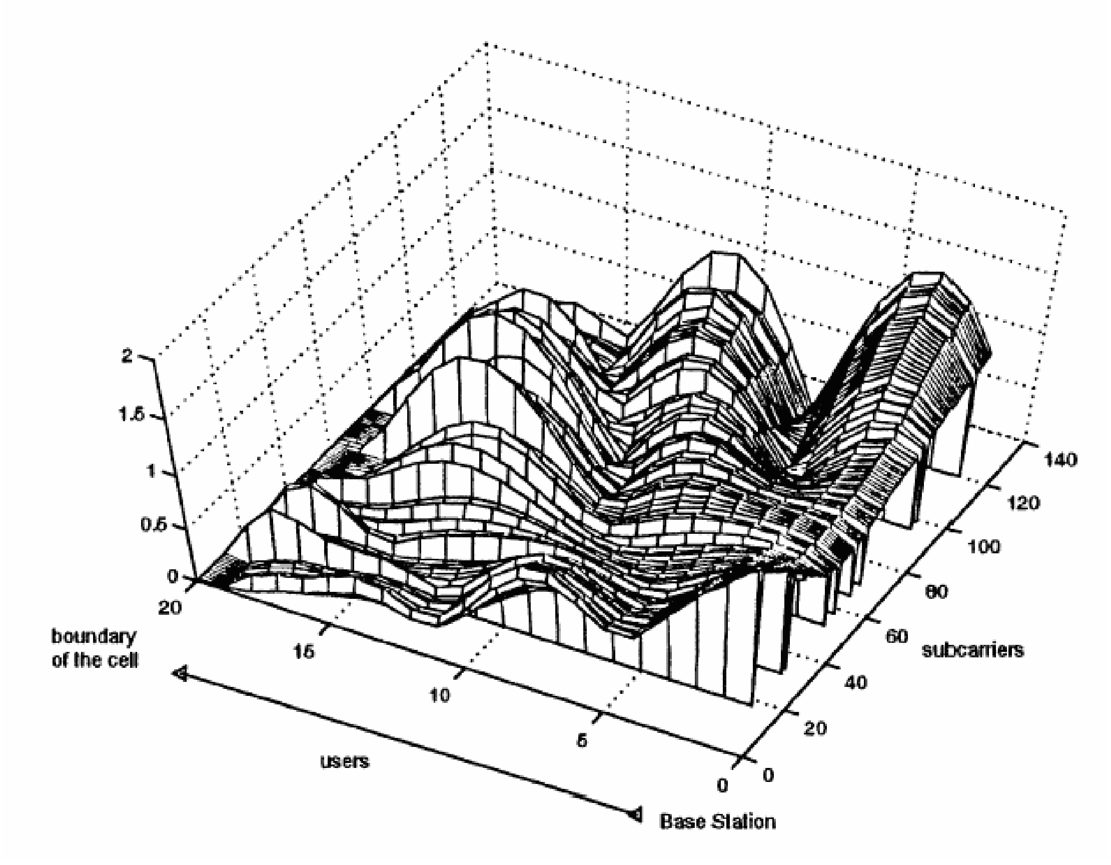


Figure 2.2. An example of channel gain.

Therefore, multiuser OFDM communication system can take the advantage of channel diversity among users in different locations to adaptively assign subcarriers and allocate modulation levels. In particular, large subcarrier gains result in higher order modulation to carry more bits/symbol, while subcarriers in deep fade carry one or even zero bits/symbol.

Hence power consumption can be greatly reduced under a good resource allocation scheme. Then, the adaptive subcarrier assignment and bit allocation for multiuser OFDM system can be formulated as a nonlinear integer programming problem as shown in (2.3) to minimize the total power consumption while satisfying the users' data communication request and system's constraints[2]. In this dissertation, we focus on proposing an efficient and effective algorithm to solve (2.3) for a good enough feasible solution.

$$\min_{c_{k,n},\rho_{k,n}} P_T \left(= \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k(c_{k,n})\right)$$

subject to

$$R_k = \sum_{n=1}^{N} c_{k,n},\ k = 1,\ldots,K$$

$$\sum_{k=1}^{K}\rho_{k,n} = 1,\ n = 1,\ldots,N$$

$$c_{k,n} \in D, \text{ for all } k,n$$

$$\rho_{k,n} = \begin{cases} 0 & \text{if } c_{k,n} = 0 \\ 1 & \text{otherwise} \end{cases}, \text{ for all } k,n \qquad (2.3)$$

where P_T denotes the total transmission power to be minimized; $\rho_{k,n}$ is an indicator variable, and a subcarrier can be occupied by at most one user as described by the equality constraint on $\rho_{k,n}$; R_k (bits per OFDM symbol) denotes the requested data rate of the kth user; $c_{k,n}$ denotes the number of bits of the kth user assigned to the nth subcarrier, and $D = \{0, 1, 2, \ldots, M\}$ denotes the set of all possible values for $c_{k,n}$, thus the first equality constraint in the problem formulated in (2.3) implies that the subcarrier assignment and bit allocation should meet the user's data rate request.

Clearly, (2.3) is a nonlinear integer programming problem or a constrained combinatorial optimization problem, because $\rho_{k,n}$ and $c_{k,n}$ are integers for all k, n, and P_T is nonlinear. To cope with the computational complexity of this problem, we will employ the OO theory based algorithms to efficiently seek a good enough solution with high probability instead of searching the optimal solution.

[2] Notation employed here is followed from [3].

2.3 Review of Ordinal Optimization Theory

The Ordinal Optimization theory [10]-[12] is a new methodology designed to deal with hard problems such as the lack of structure problems, problems with uncertainties, or problems with huge sample space that grows exponentially with respect to the problem size. The problem considered in this dissertation is of the latter kind. There are two basic tenets of the OO theory. The first is that of order versus value in decision making. Obviously, to determine whether $P_T(\rho_1, c_1) < P_T(\rho_2, c_2)$ is much easier than to determine $P_T(\rho_2, c_2) - P_T(\rho_1, c_1) = ?$. In other words, consider the intuitive example of determining which of the two boxes in two hands is heavier versus identifying how much heavier one is than the other. The second tenet is the goal softening. Instead of asking the best for sure in optimization, it settles for the good enough with high probability. A conclusion drawn from the OO theory is the following.

Suppose we simultaneously evaluate a large set of alternatives very approximately and order them according to the approximate evaluation. Then there is high probability that we can find the actual good alternatives if we limit ourselves to the top *n*% of the observed good choices.

Firstly, we use only a very rough model to order the goodness of a solution relying on the robustness of ORDER against noise and model error to separate the good from the bad. Second, we soften the goal of the problem and look for a good enough solution, which is among the top *n*% of the search space, with high probability. These two steps greatly reduce the computational burden and search difficulties of the problem. A summary of these search procedures for obtaining a good enough feasible solution of ASABA problem with high probability can be described in the following: i) Using either a uniform selection or a heuristic method to select a feasible representative set **I** with size *I* for the search space. ii) Using an easily computed crude model to roughly evaluate and order the performance of each sample in **I** and collect the top *s* samples to form a selected subset (SS), which is the estimated good enough subset. The OO theory guarantees that SS consists of actual good enough solutions with high probability. iii) Evaluating the exact objective value for each sample in SS to obtain the good enough solution.

2.4 Review of Genetic algorithm

Genetic algorithm [13]-[16] is population-based searching technique based on the idea of "survival of the fittest", which repeats evaluation, selection, crossover, mutation and repair after initialization until a stopping criterion is satisfied. In a GA, a set of solutions are analyzed and modified by genetic operations simultaneously, where selection operator can select some "good" solutions as seeds, crossover operator can generate new solutions hopefully retaining good features from parents, mutation operator can enhance diversity and provide a chance to escape from local optima, and repair operator can avoid infeasible solutions during the evolving processes.

In this dissertation, GA is used in the second method to select *I* good solutions from the search space of the considered problem. The flow chart of the GA used in our approach is shown in Figure 2.3.

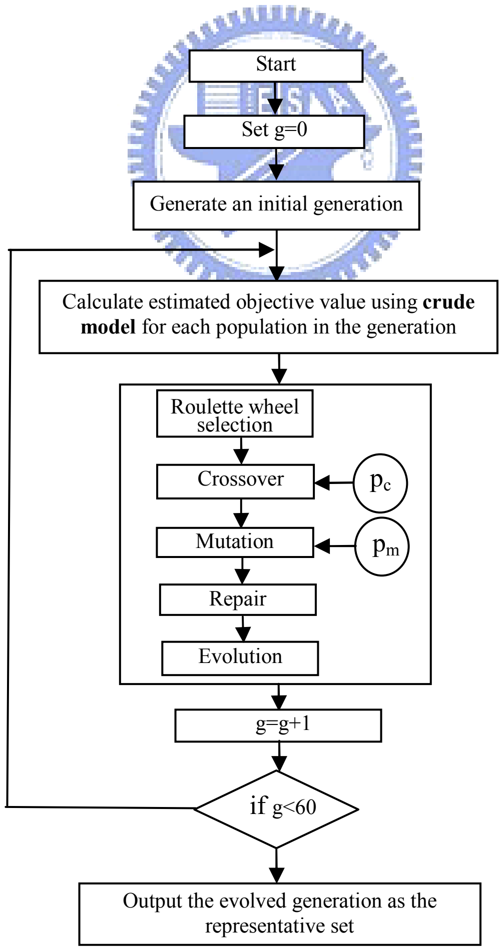

Start
Set g=0
Generate an initial generation
Calculate estimated objective value using **crude model** for each population in the generation
Roulette wheel selection
Crossover
p_c
Mutation
p_m
Repair
Evolution
g=g+1
if g<60
Output the evolved generation as the representative set

Figure 2.3. The GA procedure to the proposed method.

2.5 Review of Artificial Neural Network

An Artificial Neural Network [17]-[19] is a mathematical model or computational model based on biological neural networks, which learns from previously prepared input/output data then to determine the output data for a given input data. The key element of ANN is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. Through a learning process, various types of ANN models can be effectively used for many applications, such as pattern recognition, function approximation or data classification. In this dissertation, a simple feed-forward ANN is employed as a crude model to roughly evaluate the objective value of the considered problem, (2.3).

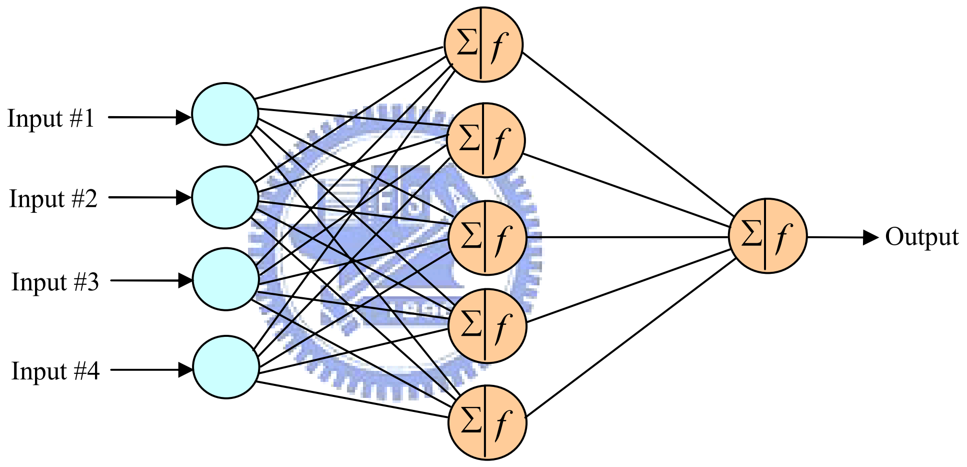


Figure 2.4. Diagram of a simple feed-forward ANN with a single hidden layer.

A diagram of a simple feed-forward ANN with a single hidden layer is shown in Figure 2.4. The ANN consists of one input layer, one hidden layer, and one output layer. Each layer contains neurons (circles in the figure), and the neurons in each layer are fully connected to the nearest layers above or below by the lines. A weight is associated with each arc line. The neurons in the input layer receive the input vectors, and neurons in the output layer produce the output vectors in response to the input vectors. The layers are connected through the weighted arcs. Neurons in hidden layers and the output layer perform two operations: they sum the products of arc weights and the signals from the previous layer, and pass that sum through a transfer function—often a *sigmoid*, *hyperbolic* *tangent sigmoid, or linear* function.

Supervised learning can be used to train an appropriately configured ANN to implement a mapping. In our methods, an ANN is off-line trained and is employed as a surrogate model to estimate the objective value of the considered problem.

Chapter 3

A Real-Time Method for Adaptive Subcarrier Assignment and Bit Allocation problem of Multiuser OFDM System

To deal with the high mobility of the dynamic behaviors of mobile users and devices, the real-time application requirement is the premise of the wireless network resource management solution methods. Therefore, in this chapter we will present an OO theory based four-stage approach for the subcarrier assignment and bit allocation problem of multiuser OFDM system with emphasis on the solution quality and the computational efficiency to meet the real-time application requirement.

In the first three stages, we will use surrogate models to select a subset of *estimated good enough* feasible solutions from the candidate solution set so as to reduce the search space of subcarrier assignment until l (=3) good enough subcarrier assignment patterns are obtained. The surrogate models in these three stages are refined from stage to stage, because the required computation time has become less as the size of the candidate solution set diminishes. Then in the fourth stage, we will employ a greedy algorithm [20] for single user on each of the l subcarrier assignment patterns to obtain the corresponding optimal bit allocation, and the one achieving the smallest power consumption will be the good enough feasible solution that we seek. The most computationally intensive stage among the four lies in the first stage, in which we need to solve a continuous version of the considered problem. To cope with the computational complexity caused by the nonlinear programming algorithm, we propose a *hardware implementable* numerical method to exploit the merit of nowadays *integrated circuit technology* so as to obtain the optimal continuous solution extremely fast. Therefore, our OO theory based four-stage approach not only obtains a good enough feasible solution but also meets the real-time application requirement.

We organize chapter 3 in the following manner. In Section 3.1, we will present the Dual Projected Gradient (DPG) method to solve the continuous version of the considered problem. In the meantime, we will also present the hardware architecture of the DPG method. In

Section 3.2, we will present the OO theory based four-stage approach for finding a good enough feasible solution. In Section 3.3, we will present some test results and compare our approach with some existing methods in the aspects of power consumption and computation time. In Section 3.4, we will make a conclusion.

3.1 The Dual Projected Gradient Method and Its Hardware Architecture

3.1.1 Reformulation

Since problem (2.3) is a computationally intractable *combinatorial problem*, Wong *et al.* introduced the variable $r_{k,n} = c_{k,n}\rho_{k,n}$ to transform (2.3) into the following continuous-variable convex optimization problem over a convex set.

$$\min_{\substack{r_{k,n}\in[0,M\rho_{k,n}]\\ \rho_{k,n}\in[0,1]}} \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k\left(\frac{r_{k,n}}{\rho_{k,n}}\right)$$

subject to

$$R_k = \sum_{n=1}^{N} r_{k,n},\ k=1,\dots,K$$

$$1 = \sum_{k=1}^{K}\rho_{k,n},\ n=1,\dots,N \tag{3.1}$$

where both $\rho_{k,n}$ and $r_{k,n}$ are continuous variables, satisfying $0 \le \rho_{k,n} \le 1$, and $0 \le r_{k,n} \le M\rho_{k,n}$, respectively. Note: when $\rho_{k,n}=0$, then $r_{k,n}=0$, and $\frac{0}{0}$ becomes undefined, therefore, we define $f(\frac{0}{0})$ in (3.1) as $f(0)$.

If we apply a typical Lagrangian relaxation method to solve (3.1), there will be a *singularity* problem in the variable $\rho_{k,n}$ just like that shown in [3]. In order to develop a hardware implementable dual-type method, we need to eliminate this singularity problem by adding extra terms in the objective function of (3.1) to strictly convexify $\rho_{k,n}$, for every k,n, as follows:

$$\min_{\substack{r_{k,n}\in[0,M\rho_{k,n}]\\ \rho_{k,n}\in[0,1]}} \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k\left(\frac{r_{k,n}}{\rho_{k,n}}\right) + \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\sigma}{2}\rho_{k,n}^2$$

subject to $$R_k = \sum_{n=1}^{N} r_{k,n},\ k = 1,\dots,K$$

$$1 = \sum_{k=1}^{K} \rho_{k,n},\ n = 1,\dots,N \tag{3.2}$$

If $\sigma > 0$, (3.2) is a convex programming problem with strictly convex objective function.

Remark 3.1: (i) Adding the extra terms $\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\sigma}{2}\rho_{k,n}^2$ will help us build the surrogate model in Stage 1 of our OO theory based four-stage approach as will be shown later. (ii) The optimal solution of (3.2) is a good approximate solution of (3.1) provided that σ is small enough.

3.1.2 The Dual Projected Gradient (DPG) Method for Solving (3.2)

The DPG method will solve the dual problem of (3.2), as shown in (3.3), instead of solving (3.2) directly.

$$\max \phi(\lambda) \tag{3.3}$$

where $\lambda = (\lambda_1^r,\dots,\lambda_K^r,\lambda_1^\rho,\dots,\lambda_N^\rho)^T$ is the Lagrange multiplier vector such that λ_k^r corresponds to the kth user's data rate request constraint, and λ_n^ρ corresponds to the nth subcarrier assignment constraint, and the dual function $\phi(\lambda)$ is defined by

$$\phi(\lambda) = \min_{\substack{r_{k,n}\in[0,M\rho_{k,n}]\\ \rho_{k,n}\in[0,1]}} \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k(\frac{r_{k,n}}{\rho_{k,n}}) + \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\sigma}{2}\rho_{k,n}^2 + \sum_{k=1}^{K}\lambda_k^r(R_k - \sum_{n=1}^{N} r_{k,n}) + \sum_{n=1}^{N}\lambda_n^\rho(\sum_{k=1}^{K}\rho_{k,n} - 1) \tag{3.4}$$

By suitably rearranging the terms, (3.4) can be rewritten in a more compact form as

$$\phi(\lambda) = \min_{\substack{r_{k,n}\in[0,M\rho_{k,n}]\\ \rho_{k,n}\in[0,1]}} \sum_{n=1}^{N}\sum_{k=1}^{K}\{\frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k(\frac{r_{k,n}}{\rho_{k,n}}) + \frac{\sigma}{2}\rho_{k,n}^2 + \frac{1}{N}\lambda_k^r R_k - \lambda_k^r r_{k,n} + \lambda_n^\rho \rho_{k,n} - \frac{1}{K}\lambda_n^\rho\} \tag{3.5}$$

The DPG method employs the following iterations to solve (3.3):

$$\lambda(t+1) = \lambda(t) + \beta(t)\nabla\phi(\lambda(t)) \tag{3.6}$$

where t denotes the iteration index, $\beta(t)$ is a positive step-size and $\nabla\phi(\lambda(t)) = (\frac{\partial\phi(\lambda(t))}{\partial\lambda_1^r},\dots,\frac{\partial\phi(\lambda(t))}{\partial\lambda_K^r},\frac{\partial\phi(\lambda(t))}{\partial\lambda_1^\rho},\dots,\frac{\partial\phi(\lambda(t))}{\partial\lambda_N^\rho})^T$ is the gradient of $\phi(\lambda(t))$ evaluated at $\lambda = \lambda(t)$, which can be calculated by the following formula [21]

$$\frac{\partial \phi(\lambda(t))}{\partial \lambda_k^r} = R_k - \sum_{n=1}^{N} \hat{r}_{k,n}, k = 1,\ldots,K \tag{3.7}$$

$$\frac{\partial \phi(\lambda(t))}{\partial \lambda_n^\rho} = \sum_{k=1}^{K} \hat{\rho}_{k,n} - 1, n = 1,\ldots,N \tag{3.8}$$

The $(\hat{r}^T, \hat{\rho}^T) = (\hat{r}_{1,1},\ldots,\hat{r}_{K,N}, \hat{\rho}_{1,1},\ldots,\hat{\rho}_{K,N})$ in (3.7) and (3.8) is the solution of the minimization problem on the RHS of (3.5) for a given $\lambda(t)$. Therefore to obtain $\nabla\phi(\lambda(t))$, we need to solve for $\hat{r}$ and $\hat{\rho}$ first, and the key for making the DPG method *hardware implementable* is we use a two-phase strategy to fulfill this task.

The first phase is to solve the minimization problem on the RHS of (3.5) without the inequality constraints on $r_{k,n}$ and $\rho_{k,n}$, which is shown in (3.9) and will be called the *unconstrained minimization problem*.

$$\min \sum_{n=1}^{N}\sum_{k=1}^{K} \{\frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k(\frac{r_{k,n}}{\rho_{k,n}}) + \frac{\sigma}{2}\rho_{k,n}^2 + \frac{1}{N}\lambda_k^r R_k - \lambda_k^r r_{k,n} + \lambda_n^\rho \rho_{k,n} - \frac{1}{K}\lambda_n^\rho\} \tag{3.9}$$

We denote the optimal solution of the unconstrained minimization problem (3.9) by $(\tilde{r}^T, \tilde{\rho}^T) = (\tilde{r}_{1,1},\ldots,\tilde{r}_{K,N}, \tilde{\rho}_{1,1},\ldots,\tilde{\rho}_{K,N})$. $(\tilde{r}^T, \tilde{\rho}^T)$ can be obtained from solving the first order necessary conditions, which can be fully decomposed into $N \cdot K$ independent 2x2 equations as shown below: For $n = 1,\ldots,N,\ k = 1,\ldots,K,$

$$\frac{1}{\alpha_{k,n}^2} f_k'(\frac{\tilde{r}_{k,n}}{\tilde{\rho}_{k,n}}) - \lambda_k^r(t) = 0 \tag{3.10}$$

$$\frac{1}{\alpha_{k,n}^2}[f_k(\frac{\tilde{r}_{k,n}}{\tilde{\rho}_{k,n}}) - f_k'(\frac{\tilde{r}_{k,n}}{\tilde{\rho}_{k,n}})\frac{\tilde{r}_{k,n}}{\tilde{\rho}_{k,n}}] + \sigma\tilde{\rho}_{k,n} + \lambda_n^\rho(t) = 0 \tag{3.11}$$

For each n and each k, the simple 2x2 equations shown in (3.10) and (3.11) can be solved analytically by

$$\tilde{r}_{k,n} = \tilde{\rho}_{k,n} f_k'^{-1}(\lambda_k^r(t)\alpha_{k,n}^2) \tag{3.12}$$

$$\tilde{\rho}_{k,n} = \frac{\lambda_n^\rho(t) - \frac{1}{\alpha_{k,n}^2}[f_k(f_k'^{-1}(\lambda_k^r(t)\alpha_{k,n}^2)) - \lambda_k^r(t)\alpha_{k,n}^2 f_k'^{-1}(\lambda_k^r(t)\alpha_{k,n}^2)]}{\sigma} \tag{3.13}$$

where f^{-1} is the inverse function of f, which can be derived once f is given, for example the f given in (2.1).

The second phase is to handle the inequality constraints disappearing in (3.9) by projecting $(\tilde{r}_{k,n}, \tilde{\rho}_{k,n})$ onto the range $[0, M\rho_{k,n}]\times[0,1]$, for each k and each n. The projection can be calculated based on simple geometry as shown in (3.14).

$$(\hat{r}_{k,n}, \hat{\rho}_{k,n}) = \begin{cases} (M,1) & \text{if } \tilde{r}_{k,n} > M,\ \tilde{r}_{k,n} > M + \frac{1}{M} - \frac{1}{M}\tilde{\rho}_{k,n} \\ \left(\frac{M}{M^2+1}(M\tilde{r}_{k,n} + \tilde{\rho}_{k,n}), \frac{M}{M^2+1}(\tilde{r}_{k,n} + \frac{1}{M}\tilde{\rho}_{k,n})\right) & \text{if } \tilde{r}_{k,n} > M\tilde{\rho}_{k,n},\ -\frac{1}{M}\tilde{\rho}_{k,n} \le \tilde{r}_{k,n} \le M + \frac{1}{M} - \frac{1}{M}\tilde{\rho}_{k,n} \\ (\tilde{r}_{k,n}, \tilde{\rho}_{k,n}) & \text{if } 0 \le \tilde{\rho}_{k,n} \le 1,\ 0 \le \tilde{r}_{k,n} \le M\tilde{\rho}_{k,n} \\ (0, \tilde{\rho}_{k,n}) & \text{if } 0 \le \tilde{\rho}_{k,n} \le 1,\ \tilde{r}_{k,n} < 0 \\ (\tilde{r}_{k,n}, 1) & \text{if } \tilde{\rho}_{k,n} > 1,\ 0 \le \tilde{r}_{k,n} \le M \\ (0,1) & \text{if } \tilde{\rho}_{k,n} > 1,\ \tilde{r}_{k,n} < 0 \\ (0,0) & \text{if } \tilde{\rho}_{k,n} < 0, \tilde{r}_{k,n} < -\frac{1}{M}\tilde{\rho}_{k,n} \end{cases} \tag{3.14}$$

The resulted projection will be the optimal solution, $(\hat{r}^T, \hat{\rho}^T) = (\hat{r}_{1,1}, \ldots, \hat{r}_{K,N}, \hat{\rho}_{1,1}, \ldots, \hat{\rho}_{K,N})$, of the minimization problem on the RHS of (3.5) as had been proven in [22] and [23].

Convergence of the DPG method using a positive constant step-size $\hat{\beta}$ (i.e. setting $\beta(t) = \hat{\beta}$ for every t) can be proved like that of the Dual Projected Pseudo Quasi Newton method in [22] and [23]. A typical value of $\hat{\beta}$ is 0.5.

As indicated previously, the optimal solution of (3.2) will be an approximate solution of (3.1) if σ is sufficiently small. However, larger σ will speed up the convergence of the DPG method. Thus, we let $\sigma_0(=1), \sigma_1, \ldots, \sigma_{j_{\max}}$ be a decreasing sequence of σ such that $\sigma_{j+1} = \eta\sigma_j$, where $\eta(<1)$ is a reducing factor, and $j_{\max}$ is a positive integer that makes $\sigma_{j_{\max}} (= \eta^{j_{\max}})$ small enough. Then, we can initially set $\sigma = \sigma_0$ in (3.2) and use the obtained optimal solution as the initial guess to solve (3.2) again with $\sigma = \sigma_1$. Repeating this process until $\sigma = \sigma_{j_{\max}}$, and the corresponding optimal solution of (3.2) will be a very good approximate solution of (3.1).

3.1.3 Hardware Implementable Algorithm of the DPG Method

To enhance the computation speed further, we will propose a hardware architecture to implement the DPG method. To do so, we need to put the DPG method in algorithmic steps that can be mapped into the operations of the arrays of Processing Elements (PEs), which are defined as the hardware for carrying out the arithmetic operations in the DPG method. First of all, we should modify the convergence criteria of the DPG method by setting a large enough number of iterations, say $t_{\max}$, such that if $t \geq t_{\max}$, we assume the DPG method converges. Furthermore, we should predetermine the value of $j_{\max}$, which is the number of times that σ will be reduced.

It can be observed that all the computation formulae of the DPG method, (3.7), (3.8), (3.12), (3.13), and (3.14), achieve a complete decomposition property, that is the computations for each k and each n can be carried out independently. Furthermore, all these computations consist of simple arithmetic operations only. These facts imply that the DPG method is very suitable for hardware implementation. However, it is not wise to assign a PE to calculate each individual component, for example calculating $\tilde{\rho}_{k,n}$ for every k and every n in (3.13), because this will make the size of the integrated circuit chip too big to be implemented. Therefore, to render the difficulty of chip size we can use N arrays of PEs such that the nth PE array will take care of all the K computations corresponding to one n in (3.7), (3.8), (3.12)-(3.14). Although such arrangement seems to degrade the computational speed, in fact it will not affect the purpose of real-time application as shown in Section 3.3. On the basis of using N PE arrays, we can put the DPG method in the following parallel-computation algorithmic steps:

Step 0: Set the values of $\lambda(0)$, $\sigma(0)$, η (<1), $t_{\max}$, $j_{\max}$; set $j = 0$ and $t = 0$.

Step 1: Set $k = 1$, $R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_n^{\rho}}) = -1$ for each n. (Note: $R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_n^{\rho}})$ represents a temporary memory for the term $\frac{\partial\phi(\lambda(t))}{\partial\lambda_n^{\rho}}$).

Step 2: Compute in parallel $(\tilde{r}_{k,n}, \tilde{\rho}_{k,n})$ by calculating (3.12) and (3.13) for each n.

Step 3: Project in parallel $(\tilde{r}_{k,n}, \tilde{\rho}_{k,n})$ onto the range $([0, M\rho_{k,n}], [0,1])$ for each n using (3.14) to obtain $(\hat{r}_{k,n}, \hat{\rho}_{k,n})$ for each n.

Step 4: Compute $\frac{\partial \phi(\lambda(t))}{\partial \lambda_k^r} = R_k - \sum_{n=1}^{N} \hat{r}_{k,n}$.

Step 5: Compute in parallel $R(\frac{\partial \phi(\lambda(t))}{\partial \lambda_n^\rho}) := R(\frac{\partial \phi(\lambda(t))}{\partial \lambda_n^\rho}) + \hat{\rho}_{k,n}$ for each n.

Step 6: Update $\lambda_k^r(t+1) = \lambda_k^r(t) + \hat{\beta}\frac{\partial \phi(\lambda(t))}{\partial \lambda_k^r}$.

Step 7: If k=K, go to **Step 8**; otherwise, set $k = k+1$ and return to **Step 2.**

Step 8: Update in parallel $\lambda_n^\rho(t+1) = \lambda_n^\rho(t) + \hat{\beta}\frac{\partial \phi(\lambda(t))}{\partial \lambda_n^\rho}$ for each n. (Note: the value of $\frac{\partial \phi(\lambda(t))}{\partial \lambda_n^\rho} = \sum_{k=1}^{K} \hat{\rho}_{k,n} - 1$ is stored in $R(\frac{\partial \phi(\lambda(t))}{\partial \lambda_n^\rho})$.)

Step 9: If $t \geq t_{\max}$, go to **Step 10**; otherwise set $t = t+1$ and return to **Step 1**.

Step 10: Set $\sigma(j+1) = \eta\sigma(j)$.

Step 11: If $j \geq j_{\max}$, go to **Step 12**; otherwise, set $j = j+1$, $\lambda_k^r(0) = \lambda_k^r(t)$ for each k, $\lambda_n^\rho(0) = \lambda_n^\rho(t)$ for each n, $k=1$, $t=1$, and return to **Step 1**.

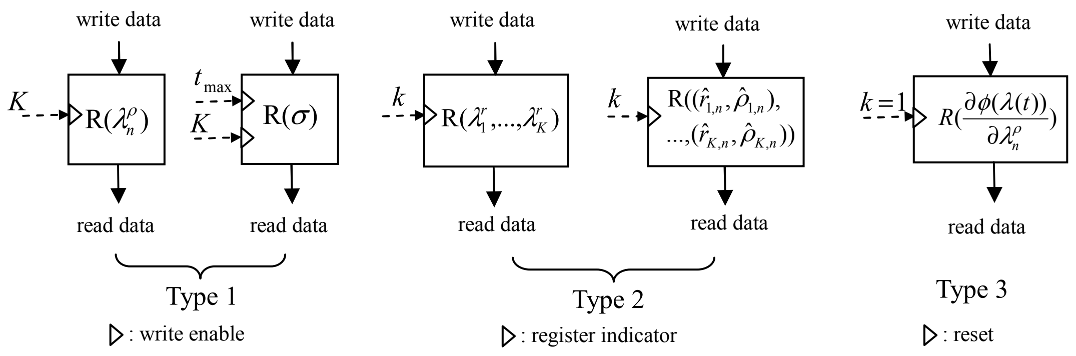
Step 12: Stop.

Remark 3.2: The reason why we execute Step 5 for K iterations to calculate $\frac{\partial \phi(\lambda(t))}{\partial \lambda_n^\rho}$ for each n is because we use N instead of $N \cdot K$ PE arrays.

3.1.4 Hardware Computing Architecture of the DPG Algorithm

Mapping the DPG algorithm into a hardware architecture needs to consider the following four parts: (i) the data storage, (ii) the computations, (iii) the iteration count and the convergence detection for branching the data flow, and (iv) the interconnections between PEs and data storage elements. In the following, we will present the details of each part.

(i) Regarding the data storage, we employ registers to store the constants, η, $t_{\max}$, $j_{\max}$, $\hat{\beta}$, R_k, $\alpha_{k,n}^2$, and $1/\alpha_{k,n}^2$, of the algorithm, and the temporary values of the variables, $\lambda_n^\rho(t)$, $R(\frac{\partial \phi(\lambda(t))}{\partial \lambda_n^\rho})$, $\lambda_k^r(t)$, $k = 1,\ldots,K$, $\sigma(j)$, $(\hat{r}_{k,n}, \hat{\rho}_{k,n})$, $k = 1,\ldots,K$, generated in the algorithm. For the sake of simplicity in interconnection, the registers for storing the constants are embedded in the PE responsible for the computations that need the constants. However, there are three types of registers denoted by $\mathrm{R}(\Delta)$ for storing the temporary computed-values of the variables Δ as shown in Figure 3.1. For example, $\mathrm{R}(\lambda_n^{\rho})$ denotes the register for storing the computed value of λ_n^{ρ}. The type 1 registers are for storing the computed values of λ_n^{ρ} and σ; λ_n^{ρ} is updated when $k=K$, and σ is updated when both $k=K$ and $t=t_{\max}$ as indicated in Steps 7-8 and Steps 9-10, respectively. Therefore, a write enable controlled by the value of *k* and *t* are needed as shown in Figure 3.1. The type 2 registers are *K*-bank registers for storing the computed values of $\lambda_k^{r}, k=1,\ldots,K,$ or $(\hat{r}_{k,n}, \hat{\rho}_{k,n}),\ k=1,\ldots,K,$ such that the *k*th bank will store the value of λ_k^{r} or $(\hat{r}_{k,n}, \hat{\rho}_{k,n})$. Since the iteration index of the innest loop of the DPG algorithm is *k*, we need a register indicator *k* to point to the corresponding register bank as shown in Figure 3.1. Furthermore, λ_k^{r} and $(\hat{r}_{k,n}, \hat{\rho}_{k,n})$ are computed for every *k*, thus type 2 registers are always write enabled. The type 3 register is for storing the value of $\sum_{l=1}^{k} \hat{\rho}_{l,n} - 1$; therefore type 3 register is always write enabled, however its value has to be reset to -1 when $k=1$ as shown in Figure 3.1. It should be noted that the action of writing data into the registers occurs at the end of the clock pulse (i.e. at the positive edge of the next clock pulse) when write enable is active. For example, σ_j in $\mathrm{R}(\sigma)$ is updated to σ_{j+1} at the end of the clock pulse corresponding to both $k=K$ and $t=t_{\max}$.



write data
read data
K
$\mathrm{R}(\lambda_n^{\rho})$
$t_{\max}$
K
$\mathrm{R}(\sigma)$
k
$\mathrm{R}(\lambda_1^{r},\ldots,\lambda_K^{r})$
k
$\mathrm{R}((\hat{r}_{1,n},\hat{\rho}_{1,n}),\ldots,(\hat{r}_{K,n},\hat{\rho}_{K,n}))$
$k=1$
$R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_n^{\rho}})$
Type 1
▷ : write enable
Type 2
▷ : register indicator
Type 3
▷ : reset

Figure 3.1. The three types of registers for storing the temporary computed values of the DPG algorithm.

(ii) Regarding the computations, we employ seven types of PE to carry out all the arithmetic operations required in the DPG algorithm. These PEs are named as $\mathrm{PE}_{\mathrm{n}}^{1}$, $\mathrm{PE}_{\mathrm{n}}^{2}$, $\mathrm{PE}_{\mathrm{n}}^{3}$, PE^{4}, PE^{5}, $\mathrm{PE}_{\mathrm{n}}^{6}$, and PE^{7}, where the superscript denotes the type of PE, the subscript denotes the index of the array, and the PE without any subscript means it is single in the N PE arrays. Each type of PE consists of different hardware components needed for calculating a specific step in the DPG algorithm and yields the results needed in other step or steps. We will state the hardware components in a PE and its corresponding algorithmic step in the following. In the nth PE array, $\mathrm{PE}_{\mathrm{n}}^{1}$ performs Step 2 and outputs the computed $(\tilde{r}_{k,n}, \tilde{\rho}_{k,n})$ to $\mathrm{PE}_{\mathrm{n}}^{2}$; its hardware components depend on the function $f_k(c)$. In addition, $\mathrm{PE}_{\mathrm{n}}^{1}$ consists of registers for storing $\alpha_{k,n}^{2}$ and $1/\alpha_{k,n}^{2}$ for all k and a register indicator k used to choose the corresponding register. $\mathrm{PE}_{\mathrm{n}}^{2}$ consists of six multipliers, four adders, and several comparators to perform Step 3 and output the computed $(\hat{r}_{k,n}, \hat{\rho}_{k,n})$ to $\mathrm{PE}_{\mathrm{n}}^{6}$, PE^{4}, and $\mathrm{R}((\hat{r}_{k,n}, \hat{\rho}_{k,n}), k = 1,\ldots,K)$; $\mathrm{PE}_{\mathrm{n}}^{3}$ consists of a register for storing the constant $\hat{\beta}$, one adder and one multiplier to perform Step 8 and output the computed $\lambda_n^{\rho}(t+1)$ to $\mathrm{R}(\lambda_n^{\rho})$; $\mathrm{PE}_{\mathrm{n}}^{6}$ consists of one adder to perform Step 5 and output the computed $\sum_{k=1}^{K} \hat{\rho}_{k,n} - 1$ to $R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_n^{\rho}})$ and $\mathrm{PE}_{\mathrm{n}}^{3}$. The single PE^{4} consists of $\log_2(N+1)$ adders to perform Step 4 and output the computed $\frac{\partial\phi(\lambda(t))}{\partial\lambda_k^{r}}$ to the single PE^{5}. In addition, PE^{4} consists of registers for storing R_k for all k and a register indicator k for choosing the corresponding register. The single PE^{5} consists of a register for storing the constant $\hat{\beta}$, one adder and one multiplier to perform Step 6; PE^{5} outputs the computed $\lambda_k^{r}(t+1)$ to $\mathrm{R}(\lambda_1^{r},\ldots,\lambda_K^{r})$. The single PE^{7} consists of a register for storing the constant η and a multiplier to perform Step 10; PE^{7} outputs the computed $\sigma(j+1)$ to $\mathrm{R}(\sigma)$. We summarize the characteristics of these PEs in Table 3.1 to indicate the corresponding algorithmic step, embedded constant, input data [from], output data

[to], and the computation complexity of a PE, which is shown in the last column and will be analyzed later.

Table 3.1
The characteristics of PEs

PE	Algorithmic Step	Embedded Constant	Input Data [from]	Output Data [to]	Computation Complexity
$\mathrm{PE}_{\mathrm{n}}^{1}$	Step 2	$\alpha_{k,n}^{2}, 1/\alpha_{k,n}^{2}$	$\lambda_{n}^{\rho}(t)[\mathrm{R}(\lambda_{n}^{\rho})]$, $\sigma(j)[\mathrm{R}(\sigma)]$, $\lambda_{k}^{r}(t)[\mathrm{R}(\lambda_{1}^{r},\ldots,\lambda_{K}^{r})]$	$(\tilde{r}_{k,n}, \tilde{\rho}_{k,n})[\mathrm{PE}_{\mathrm{n}}^{2}]$	$5\otimes$ & $2\oplus$ & $1\,\mathrm{ROM}$[§]
$\mathrm{PE}_{\mathrm{n}}^{2}$	Step 3	-	$(\tilde{r}_{k,n}, \tilde{\rho}_{k,n})[\mathrm{PE}_{\mathrm{n}}^{1}]$	$\hat{r}_{k,n}[\mathrm{PE}^{4}], \hat{\rho}_{k,n}[\mathrm{PE}_{\mathrm{n}}^{6}]$, $(\hat{r}_{k,n}, \hat{\rho}_{k,n})[\mathrm{R}((\hat{r}_{k,n}, \hat{\rho}_{k,n}), k=1,\ldots,K)$	$2\otimes$ &$1\oplus$
$\mathrm{PE}_{\mathrm{n}}^{3}$	Step 8	$\hat{\beta}$	$\sum_{l=1}^{K}\hat{\rho}_{l,n}-1[\mathrm{PE}_{\mathrm{n}}^{6}]$, $\lambda_{n}^{\rho}(t)[\mathrm{R}(\lambda_{n}^{\rho})]$	$\lambda_{n}^{\rho}(t+1)[\mathrm{R}(\lambda_{n}^{\rho})]$	$1\otimes$ & $1\oplus$
PE^{4}	Step 4	R_{k}	$\hat{r}_{k,1}[\mathrm{PE}_{1}^{2}],\ldots,\hat{r}_{k,N}[\mathrm{PE}_{\mathrm{N}}^{2}]$	$\frac{\partial\phi(\lambda(t))}{\partial\lambda_{k}^{r}}[\mathrm{PE}^{5}]$	$\log_{2}(N+1)\oplus$
PE^{5}	Step 6	$\hat{\beta}$	$\frac{\partial\phi(\lambda(t))}{\partial\lambda_{k}^{r}}[\mathrm{PE}^{4}]$, $\lambda_{k}^{r}(t)[\mathrm{R}(\lambda_{1}^{r},\ldots,\lambda_{K}^{r})]$	$\lambda_{k}^{r}(t+1)[\mathrm{R}(\lambda_{1}^{r},\ldots,\lambda_{K}^{r})]$	$1\otimes$ & $1\oplus$
$\mathrm{PE}_{\mathrm{n}}^{6}$	Step 5	-	$\sum_{l=1}^{k-1}\hat{\rho}_{l,n}-1[R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_{n}^{\rho}})]$, $\hat{\rho}_{k,n}[\mathrm{PE}_{\mathrm{n}}^{2}]$	$\sum_{l=1}^{k}\hat{\rho}_{l,n}-1[R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_{n}^{\rho}}), \mathrm{PE}_{\mathrm{n}}^{3}]$	$1\oplus$
PE^{7}	Step 10	η	$\sigma(j)[\mathrm{R}(\sigma)]$	$\sigma(j+1)[\mathrm{R}(\sigma)]$	$1\otimes$

[§] $\otimes$: operation of a multiplication; $\oplus$: operation of an addition; ROM: operation of accessing data of a ROM.

(iii) There are three loops in the DPG algorithm, and the number of iterations in each loop has been set fixed. A branching will occur when completing the iterations of a loop as described in Steps 7, 9, and 11. Therefore, we need three counters to count the number of iterations consumed in each loop so as to control the branching of the DPG algorithm.

The three counters are the *k*-counter, *t*-counter and *j*-counter, denoted by CT_*k*, CT_*t* and CT_*j*, respectively, and represented by the square blocks shown in Figure 3.2. The values of CT_*k* and CT_*t* will be fed into the corresponding registers for proper operation. For example, the value *k* of CT_*k* will be used to indicate the iteration index of the innest loop, Steps 1-7, to point to the corresponding register bank in the Type 2 registers. When $k=K$, there will be a branching at Step 7, such that the output data of $\mathrm{PE}_{\mathrm{n}}^{3}$, which performs Step 8, will be written into the register $\mathrm{R}(\lambda_n^\rho)$. Similar reason applies to CT_t that when $t = t_{\max}$, there will be a branching at Step 9, such that the output data of PE^7, which performs Step 10, will be written into the register $\mathrm{R}(\sigma)$. Furthermore, when the value of CT_k changes from K to 1, the value of the register $R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_n^\rho})$ will be reset to -1 to perform Step 1. The value of CT_j will be used to detect the convergence of the DPG algorithm. Thus, when $j = j_{\max}$, the DPG algorithm will be stopped and output the approximate solution of (3.1), that is $(\hat{r}_n, \hat{\rho}_n), k = 1, \dots, K, n = 1, \dots, N$.

The counters are designed such that CT_k will circulate from 1 to K and increase by 1 for every clock pulse, CT_t will circulate from 1 to $t_{\max}$ and increase by 1 for every K clock pulses, and CT_j will increase by 1 for every $K \cdot t_{\max}$ clock pulses.

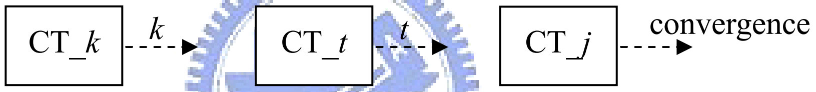


Figure 3.2. The three counters.

(iv) Now, we are ready to interconnect the array PEs, registers and counters so as to execute the DPG algorithm. From the columns of the input data [from] and the output data [to] in Table 3.1, we can use solid lines with arrow heads to indicate the direction of the data flow to interconnect the PEs and registers as shown in Figure 3.3, in which we assume N=3 and do not show the system clock for the sake of simplicity.

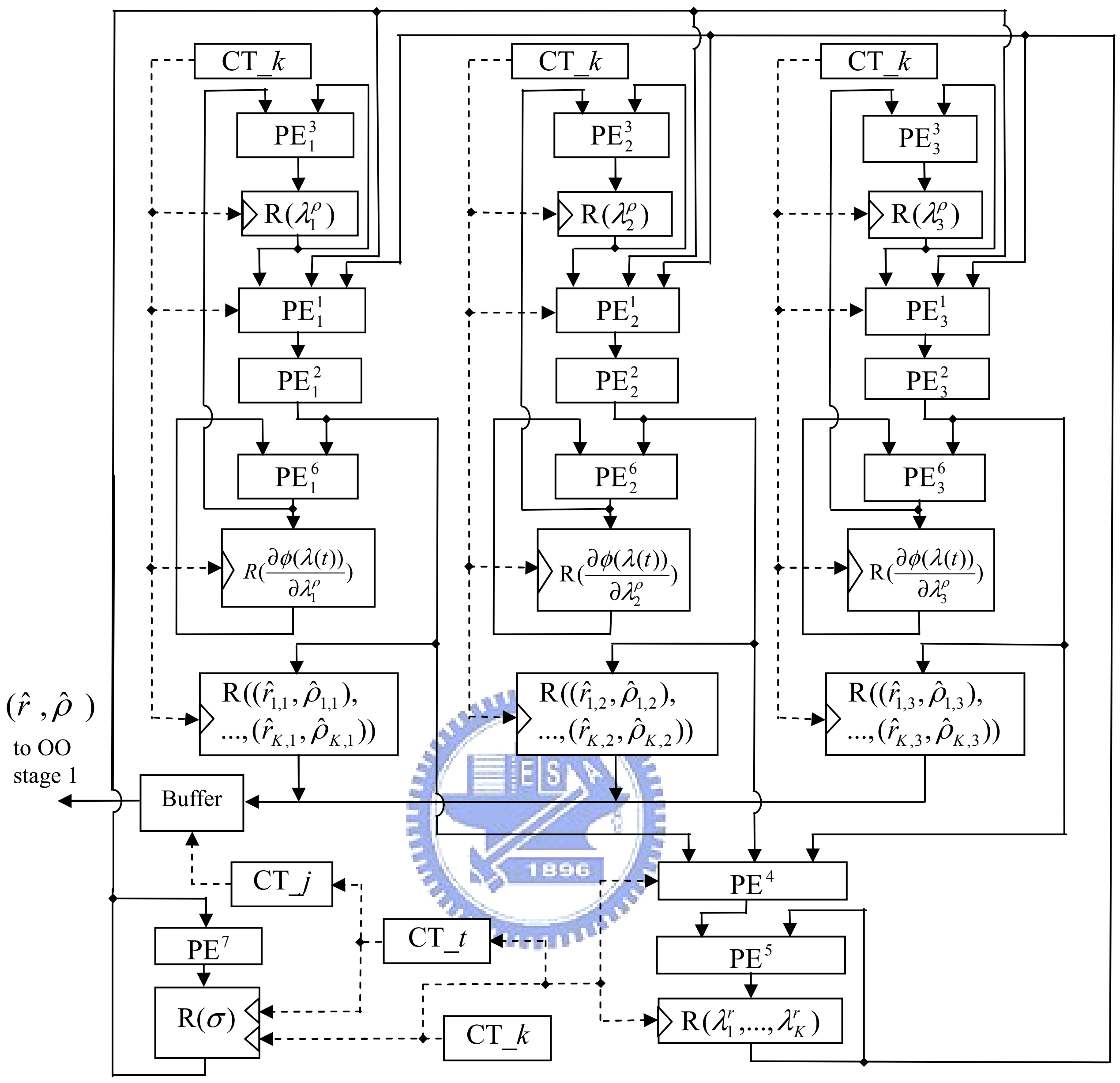
The three counters, CT_k, CT_t and CT_j, as described previously, are used to count the number of iterations and control the branching of data flow. Therefore, to distinguish from the regular data flow, we use the dashed lines with arrow heads to indicate the flow of counter values to the corresponding registers as shown in Figure 3.3.

For the sake of simplicity, we will illustrate how the hardware architecture executes the DPG algorithm for one array as follows.

We initialize the values of registers $R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_n^\rho})$ (Step 1: $R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_n^\rho}) = -1$), $\mathrm{R}(\lambda_n^\rho)$

(Step 0: $\lambda_n^\rho = 0$), $\mathrm{R}(\lambda_1^r,\ldots,\lambda_K^r)$ (Step 0: $\lambda_k^r = 0, k = 1,\ldots,K$), $\mathrm{R}(\sigma)$ (Step 0: $\sigma_0 = 1$), the counter values of CT_*k* (Step 1: *k*=1), CT_*t* (Step 0: *t*=1), CT_*j* (Step 0: *j*=1), and command PE_n^1 to start execution. Then PE_n^1 will perform Step 2 and output the resulted $(\tilde{r}_{k,n}, \tilde{\rho}_{k,n})$ to PE_n^2 as shown in Figure 3.3. Then PE_n^2 will perform Step 3 and output the resulted $(\hat{r}_{k,n}, \hat{\rho}_{k,n})$ to PE_n^6, PE^4, and $\mathrm{R}((\hat{r}_{k,n}, \hat{\rho}_{k,n}), k = 1,\ldots,K)$. It should be noted that the data $(\hat{r}_{k,n}, \hat{\rho}_{k,n})$ will be written into the *k*th register bank of $\mathrm{R}((\hat{r}_{k,n}, \hat{\rho}_{k,n}), k = 1,\ldots,K)$ as selected by the value *k* of the counter CT_*k* as shown in Figure 3.3. Once PE^4 receives the computed $\hat{r}_{k,n}, n = 1,\ldots,N$ from the N PE_n^2's, it will perform Step 4 and output the value of $\dfrac{\partial\phi(\lambda(t))}{\partial\lambda_k^r}$ to the single PE^5. Then PE^5 will perform Step 6 to update $\lambda_k^r(t)$, and the updated value will be sent to register banks $\mathrm{R}(\lambda_1^r,\ldots,\lambda_K^r)$. In the meantime, PE_n^6 will perform Step 5 using the computed data $\hat{\rho}_{k,n}$ from PE_n^2 and the data $\sum_{l=1}^{k-1}\hat{\rho}_{l,n} - 1$ in $R(\dfrac{\partial\phi(\lambda(t))}{\partial\lambda_n^\rho})$, then output the resulted data $\sum_{l=1}^{k}\hat{\rho}_{l,n}$ to $R(\dfrac{\partial\phi(\lambda(t))}{\partial\lambda_n^\rho})$ and PE_n^3 as shown in Figure 3.3. The above calculations and data flow complete one iteration of the innest loop, Steps 1-7, and should be done in *one clock pulse*, whose period need be long enough such that the output data of all PEs can reach steady states. The counter CT_*k* will increase by 1 for each activation of a clock pulse (Step 7). As *k* increases by 1, the output of the register $\mathrm{R}(\lambda_1^r,\ldots,\lambda_K^r)$ will be λ_{k+1}^r instead of λ_k^r, thus next iteration of the innest loop starts.

The above process will repeat and a branching will occur when the value of CT_*k* reaches *K*. When *k*=*K* the write enable of the register $\mathrm{R}(\lambda_n^\rho)$ will be active, and the output data of PE_n^3, which performs Step 8 with input data from PE_n^6 and $\mathrm{R}(\lambda_n^\rho)$ as shown in Figure 3.3, will be written into $\mathrm{R}(\lambda_n^\rho)$. After this clock pulse, the value of CT_*k* will start from 1 again, and the value of CT_*t* will be increased by 1 (Step 9). As *t* increases by 1, the value of $\lambda_n^\rho, n = 1,\ldots,N$, as well as $\lambda_k^r, k = 1,\ldots,K$, have been updated, then next iteration of the middle loop starts.



CT_k
PE_1^3
$R(\lambda_1^\rho)$
PE_1^1
PE_1^2
PE_1^6
$R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_1^\rho})$
$R((\hat{r}_{1,1},\hat{\rho}_{1,1}),\ldots,(\hat{r}_{K,1},\hat{\rho}_{K,1}))$
CT_k
PE_2^3
$R(\lambda_2^\rho)$
PE_2^1
PE_2^2
PE_2^6
$R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_2^\rho})$
$R((\hat{r}_{1,2},\hat{\rho}_{1,2}),\ldots,(\hat{r}_{K,2},\hat{\rho}_{K,2}))$
CT_k
PE_3^3
$R(\lambda_3^\rho)$
PE_3^1
PE_3^2
PE_3^6
$R(\frac{\partial\phi(\lambda(t))}{\partial\lambda_3^\rho})$
$R((\hat{r}_{1,3},\hat{\rho}_{1,3}),\ldots,(\hat{r}_{K,3},\hat{\rho}_{K,3}))$
$(\hat{r},\hat{\rho})$ to OO stage 1
Buffer
CT_j
PE^7
$R(\sigma)$
CT_t
CT_k
PE^4
PE^5
$R(\lambda_1^r,\ldots,\lambda_K^r)$

Figure 3.3. The hardware architecture of the DPG algorithm.

The above process will repeat and a branching will occur when the value of CT_*t* reaches $t_{\max}$ (Step 9), which will activate the write enable of $R(\sigma)$, and the output data of PE^7, which performs Step 10 with input data from $R(\sigma)$ as shown in Figure 3.3, will be written into $R(\sigma)$. In the meantime, the value of CT_*k* will start from 1 again, and CT_*t* also starts from 1 while the value of CT_*j* will be increased by 1 (Step 11). As *j* increases by 1, the above process start all over with a new σ $(=\sigma_{j+1})$. This process will proceed until CT_*j* reaches $j_{\max}$, which implies the DPG algorithm converges (Step 12), then as shown in Figure 3.3, the buffer will be activated to output the data $(\hat{r}_{k,n}, \hat{\rho}_{k,n})$, $k = 1,\ldots,K$, $n = 1,\ldots,N$, the solution of (3.2) when $\sigma = \sigma_{j_{\max}}$ or the approximate solution of (3.1). This completes the description of the execution of the DPG algorithm for one PE array.

From Figure 3.3, we see that the structure is very regular, modular, and locally interconnected; hence it is hardware implementable.

Remark 3.3: Exploiting the merits of hardware computation and parallelism of the DPG algorithm, the computation time estimated based on the hardware architecture is almost independent of the *N*, however at the cost of large area when *N* is large. Taking *N*=128 for example, we need 1666 multipliers in the hardware architecture, which is huge indeed. Manufacturing an integrated circuit with large gate counts is a challenging issue, however it can be resolved due to the advancement of semiconductor manufacturing technology.

3.1.5 Computation Complexity of the DPG Algorithm

As indicated previously, the clock period should be long enough such that the outputs of all PEs can reach steady states during an iteration. In other words, the clock period should be longer than the computation time of the *critical path* i.e. the most time consuming path of the DPG hardware architecture. To identify the critical path, we need to analyze the computation complexity of each PE first. The computation complexity of a PE can be directly derived from its corresponding arithmetic operations. For example, PE^4 performing Step 4 of the DPG algorithm requires $\log_2(N+1)$ stages of adders, therefore, it takes $\log_2(N+1)\oplus$, where $\oplus$ denotes an arithmetic operation of addition. Similar reasoning applies to PE_n^2, PE_n^3, PE^5, PE_n^6 and PE^7. However, the hardware component of PE_n^1 depends on $f_k(c)$. A typical $f_k(c)$ can be $B\cdot(2^c - 1)$, where *B* is a constant, then we may use six multipliers, three adders, and one Read Only Memory (ROM) to perform Step 2, though the details of which are omitted here. We have reported the computation complexity of all PEs in the last column of Table 3.1. The data propagation time between PEs and registers are negligible compared with addition or multiplication, and the actions of writing data into or reading data from a register consumes the time no more than that of one addition. We let T_{clock} denote the to be designed clock period. Identifying the critical path in Figure 3.3, we have

$$T_{\text{clock}} = T_{\text{PE}_{\text{n}}^{1}} + T_{\text{PE}_{\text{n}}^{2}} + T_{\text{PE}^{4}} + T_{\text{PE}^{5}} + T_{\text{PE}^{7}} + 2T_{\oplus} \qquad (3.15)$$

where $T_{\text{PE}_{\text{n}}^{\text{i}}}$ and $T_{\text{PE}^{\text{j}}}$ denotes the time complexity of executing $\text{PE}_{\text{n}}^{\text{i}}$ and PE^{j}, respectively, and $2T_{\oplus}$ represents the time needed for writing data into and reading data from registers. Note that the action of writing data into the three registers in the critical path occurs simultaneously. Therefore, writing data into three registers only counts for one $T_{\oplus}$. Similar reasoning applies to reading data from these registers. Hence, the computation time of the proposed DPG algorithm is

$$K \cdot t_{\max} \cdot j_{\max} \cdot T_{\text{clock}} \qquad (3.16)$$

3.2 The Ordinal Optimization Theory Based Four-Stage Approach

To obtain a good enough solution of the considered problem, (2.3), while using limited computation time to meet the real-time application requirement in the OFDM system, we employ an OO theory based four-stage approach as presented in the following.

3.2.1 Stage 1: Reduce the Search Space of Subcarrier Assignment Using Continuous Optimal Solution Based Model

Intuitively, a true solution that is neighboring to the optimal solution of the continuous version of the considered problem may be a good enough solution. However, all the possible subcarrier assignment $\rho_{k,n}$, $k = 1,..., K$, $n = 1,..., N$, are neighboring to the optimal continuous solution, which is denoted by $\rho_{k,n}^{*}$. Wong *et al.* in [3] chose the closest one, which, however, may cause *infeasibility*, and even if it is feasible, there is no guarantee that it is a *good enough* subcarrier assignment. Thus, in this stage, we will reduce the subcarrier assignment patterns by excluding all the ineffective subcarrier assignments based on our solution process for obtaining the approximate continuous optimal solution of (3.1). As indicated previously, $\sigma_0(=1), \sigma_1,..., \sigma_{j_{\max}}$ is a decreasing sequence of σ such that $\sigma_{j+1} = \eta\sigma_j$. We let $\rho_{k,n}^{*}(\sigma_j)$ denote the optimal continuous $\rho_{k,n}$ of (3.2) when $\sigma = \sigma_j$. Then we claim that subcarrier *n* is inappropriate to be assigned to user *k* if $\rho_{k,n}^{*}(\sigma_j) = 0$ for every $j = 0,1,..., j_{\max}$. The reason for this is simple as stated in the following.

We let $\rho^*_{k',n}(\sigma)$ denote the largest $\rho^*_{k,n}(\sigma)$ among all $k=1,\ldots,K$ for the given n. The term in the objective function of (3.2) regarding σ is $\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{\sigma}{2}\rho^2_{k,n}$. The sensitivity of this term with respect to $\rho^*_{k,n}(\sigma)$ is $\sigma\rho^*_{k,n}(\sigma)$. Suppose we increase σ by $\Delta\sigma$, then the decrease of $\rho^*_{k,n}(\sigma)$ by the amount $-\Delta\rho_{k,n}$, where we assume $\Delta\rho_{k,n}>0$, will cause an approximate *extra-reduction* of the objective value due to the increase of σ by $-\rho^*_{k,n}(\sigma)\Delta\sigma\Delta\rho_{k,n}$. Thus, decreasing $\rho^*_{k',n}(\sigma)$, will be most beneficial if we increase σ. This implies that increasing σ in (3.2) will force $\rho^*_{k',n}(\sigma)$ to decrease. Then by the constraint $\sum_{k=1}^{K}\rho_{k,n}=1$ for a given n, if $\rho^*_{k',n}(\sigma)$ decreases, there must be at least one $k''\in\{1,\ldots,K\}, k''\neq k'$, such that $\rho^*_{k'',n}(\sigma)$ will increase. Among $\rho^*_{k,n}(\sigma), k=1,\ldots,K, n=1,\ldots,N$, the ones with $\rho^*_{k,n}(\sigma)=0$ are the most possible candidates to be $\rho^*_{k'',n}(\sigma)$, because the increase of such $\rho^*_{k,n}(\sigma)$ will cause an approximately zero increment in the objective value due to $\rho^*_{k,n}(\sigma)\Delta\sigma\Delta\rho_{k,n}=0$. Thus, if they don't, it simply implies that it is inappropriate to assign subcarrier n to those users with $\rho^*_{k,n}(\sigma)=0$ and keeping 0 while σ increases. We have solved a sequence of (3.2) with σ_j, $j=0,1,\ldots,j_{\max}$, and obtained $\rho^*_{k,n}(\sigma_j)$, $j=0,1,\ldots,j_{\max}$, for all k and n. Then based on the above argument, we have that $\rho^*_{k,n}(\sigma_j)=0$, $j=0,1,\ldots,j_{\max}$, implies subcarrier n is not suitable to be assigned to user k, because when σ_{j+1} increases to σ_j ($\sigma_j>\sigma_{j+1}$) but $\rho^*_{k,n}(\sigma_j)$ is still 0.

Therefore, our crude model for selecting roughly good subcarrier assignment patterns in this stage can be stated as follows. We first set $\rho_{k,n}=0$ for the $(k,n)'$s whose corresponding optimal continuous $\rho^*_{k,n}(\sigma)=0$ for all $\sigma=\sigma_0,\sigma_1,\ldots,\sigma_{j_{\max}}$. We denote γ_n as the number of $\rho_{k,n}$'s that are not set to be 0 for a given n. Then there will be γ_n possible subcarrier assignment patterns for a given n, that means these γ_n nonzero $\rho_{k,n}$'s take turns to be 1. Subsequently, we will choose *feasible* subcarrier assignment patterns from the $\prod_{n=1}^{N}\gamma_n$ possible patterns and form the set of roughly good subcarrier assignment patterns resulted in this stage. It should be noted that checking the feasibility of a pattern, say $(\rho_{1,1},...,\rho_{K,1},...,\rho_{1,N},...,\rho_{K,N})$, is simply checking whether $M\sum_{n=1}^{N}\rho_{k,n} \geq R_k$ hold for every k.

Remark 3.4: The reduction of the search space of subcarrier assignments had been reduced from K^N (which is considered to be the worst case and may include the infeasible patterns) to $\prod_{n=1}^{N}\gamma_n$. Based on our simulation experience, a lot of γ_n's are 1.

3.2.2 Stage 2: Choose s Estimated Good Enough Subcarrier Assignment Patterns Using an Approximate Model

To evaluate the estimated performance of the feasible subcarrier assignment patterns obtained in Stage 1, we will employ an approximate model to estimate the total power consumption of each pattern as follows. For a given feasible pattern $(\rho_{1,1},...,\rho_{K,1},...,\rho_{1,N},...,\rho_{K,N})$, we let $\hat{N}_k = \sum_{n=1}^{N}\rho_{k,n}$ and $\hat{\alpha}_k = \frac{\sum_{n=1}^{N}\rho_{k,n}\alpha_{k,n}}{\hat{N}_k}$ denote the total number of subcarriers assigned to and the average power consumption coefficient of user *k*, respectively. We assume the requested data rate R_k of user *k* are distributed equally to the assigned subcarriers, then the approximate power consumed by user k, denoted by $\hat{P}_k$, can be calculated by $\hat{P}_k = \frac{\hat{N}_k}{\hat{\alpha}_k^2} f_k(R_k/\hat{N}_k)$. Consequently, the estimated total consumed power for the given pattern, denoted by $\hat{P}_T$, will be $\hat{P}_T = \sum_{k=1}^{K}\hat{P}_k$. We apply the above estimation process to each feasible pattern obtained in Stage 1 and pick the *s* (=50) patterns with smallest $\hat{P}_T$ to form the estimated good enough subset denoted by SS.

Remark 3.5: Based on the surrogate model with modeling noise *w*, the selected *s* patterns will contain at least *y* actual top *x*% patterns among the $\prod_{n=1}^{N}\gamma_n$ with probability 0.95. There exists a quantitative relationship between the values of *s*, *y*, *x*, *w* and $\prod_{n=1}^{N}\gamma_n$ as indicated in [12]. In general, larger *s* can have more *y* and smaller *x* but it will take more time for further evaluation. Therefore, for the real-time application consideration, we set $s = 50$.

3.2.3 Stage 3: Choose l Estimated Good Enough Subcarrier Assignment Patterns from the SS

To evaluate the s (=50) subcarrier assignment patterns using exact model is still too time consuming to meet the real-time application requirement. Therefore, we will employ a more refined model than the one used in Stage 2 to select l (=3) estimated good enough patterns from SS. The more refined model we employ here is a supervised learning Artificial Neural Network (ANN) used to evaluate the estimated consumed power of user k for a given subcarrier assignment pattern ($\rho_{1,1},\ldots,\rho_{K,1},\ldots,\rho_{1,N},\ldots,\rho_{K,N}$). This ANN model is constructed *off-line* using 5000 input/output pairs, and the details are described below.

The data associated with user k is the data rate request R_k, the power consumption coefficient $\alpha_{k,n}$, $n = 1,\ldots,N$, and the subcarrier assignment pattern regarding user k, i.e. $(\rho_{k,1},\ldots,\rho_{k,N})$. For such a given data vector $(R_k,\alpha_{k,1},\ldots,\alpha_{k,N},\rho_{k,1},\ldots,\rho_{k,N})^T$, an optimal bit allocation algorithm or the so called greedy algorithm [20] can be used to optimally allocate the bits to the assigned subcarrier to meet the data rate request while minimizing the power consumption of user k[3]. However, the dimension of the vector $(R_k,\alpha_{k,1},\ldots,\alpha_{k,N},\rho_{k,1},\ldots,\rho_{k,N})^T$ is too large. Thus, we will employ $(R_k,\hat{N}_k,\hat{\alpha}_k,d\alpha_k)^T$ to characterize user k, and serve as the input vector to the ANN, where $\hat{N}_k$ and $\hat{\alpha}_k$ had been defined in Stage 2, and $d\alpha_k \equiv \max_{n,\rho_{k,n}=1} \alpha_{k,n} - \min_{n,\rho_{k,n}=1} \alpha_{k,n}$. Consequently, the 5000 input/output pairs used to train the ANN can be obtained as follows. We uniformly select 5000 pairs of (R_k, $\hat{N}_k$) from the following ranges: $R_k \in [5,150]$, $\hat{N}_k \in [\frac{R_k}{M},\frac{2R_k}{M}]$. For each $\hat{N}_k$, we randomly select distinct $n_1,\ldots,n_{\hat{N}_k}$ from {1,…,N}, and randomly select α_{k,n_i} from the range [0,1.5] for each $i = 1,\ldots,\hat{N}_k$. The $\hat{\alpha}_k$

[3] This optimal bit allocation for each subcarrier assignment pattern is what we mean the exact model, and it is used for the ANN's off-line training.

and $d\alpha_k$ can be computed accordingly. Thus, we have 5000 input vectors of the form $(R_k, \hat{N}_k, \hat{\alpha}_k, d\alpha_k)^T$. Now for each $(R_k, \hat{N}_k, \alpha_{k,n_1}, \ldots, \alpha_{k,n_{\hat{N}_k}})$, we can use the greedy algorithm to compute the corresponding minimum consumed power of user k denoted by $\widetilde{P}_k$. Thus, $(R_k, \hat{N}_k, \hat{\alpha}_k, d\alpha_k)$ and $\widetilde{P}_k$ form an input/output pair. We then use the obtained 5000 input/output pairs to train a three-layer ANN. The input layer consists of four neurons, which correspond to R_k, $\hat{N}_k$, $\hat{\alpha}_k$, and $d\alpha_k$. We use 15 neurons in the hidden layer, and each neuron uses *hyperbolic tangent sigmoid* [17] as the transfer function to calculate the output from the summed value of its inputs. There is only 1 neuron in the output layer corresponding to the consumed power $\widetilde{P}_k$ of user k, and we use *linear* [17] as the transfer function. Using the above mentioned 5000 input/output pairs, we train this ANN by the Levenberg-Marquardt method proposed in [18] and [19].

Remark 3.6: In general, the more neurons, if not too many, in the hidden layer, the more accurate the ANN will be. Even though our ANN is trained off-line, more neurons in the hidden layer will increase the dimension of the arc weights thus increase the computation time for obtaining the output of ANN, which may hurt our purpose of real-time application. Since what we care here is the performance orders rather than the performance values of the tested input vectors, perfect accuracy is not necessary. Therefore, to save computation time, we select a moderate number of neurons, that is 15, in the hidden layer. In addition, there exist various activation functions such as *step* function, *hard limiter*, *ramp*, *hyperbolic tangent sigmoid*, *linear*,..., etc.[17]. Which function to be used really depends on the application. In our problem, we found *hyperbolic tangent sigmoid* and *linear* are the most suitable activation functions for the hidden layer and output layer, respectively. Furthermore, we found that the Levenberg-Marquardt method converges fast in training our ANN.

Once the ANN is trained, we can estimate $\widetilde{P}_k$ of each subcarrier assignment pattern in SS by setting up the corresponding input $(R_k, \hat{N}_k, \hat{\alpha}_k, d\alpha_k)$, feeding it into the ANN, and the output will be the estimated $\widetilde{P}_k$. Thus, the estimated total consumed power of a subcarrier assignment pattern, denoted by $\widetilde{P}_T$, in SS will be $\widetilde{P}_T = \sum_{k=1}^{K} \widetilde{P}_k$, and the l (=3) patterns with smallest estimated $\widetilde{P}_T$ among the s will be the estimated good enough subcarrier assignment patterns determined in this stage.

Remark 3.7: In [24], we have simulated that using a more accurate surrogate model to evaluate s (=50) candidate solutions, the top l (=3) solutions will contain the actual best among the s with probability 0.99.

3.2.4 Stage 4: Determine the Good Enough Subcarrier Assignment and Bit Allocation

Since there are only l (=3) subcarrier assignment patterns left, we can use the exact model, i.e. greedy algorithm, to calculate the optimal consumed power P_T for each pattern with very limited computation time. The subcarrier assignment associated with the optimal bit allocation corresponding to the smallest P_T among the three will be the good enough solution of (2.3) that we look for.

3.3 Test Results and Comparisons

In this section, we will test the performance of the proposed approach in the aspects of solution quality and computational efficiency. We will also compare with the existing subcarrier assignment and bit allocation algorithms such as the algorithm proposed by Wong *et al.* in [3], the linear programming approach proposed by Kim *et al.* in [5], the iterative algorithm proposed by Ergen *et al.* in [6], and Zhang's approach in [8].

As depicted in Figure 2.1, we assume the OFDM system has 128 subcarriers (i.e. $N = 128$) over a 5 MHz band. The system uses M-ary quadrature amplitude modulation (MQAM) such that the square signal constellations 4-QAM, 16-QAM, and 64-QAM carry two, four, and six bits/symbol, respectively; therefore in this system $D = \{0,2,4,6\}$ and $M = 6$. We adopt the approximate formula in (2.1) for the $f_k(c)$ in the transmission power $\frac{f_k(c)}{\alpha_{k,n}^2}$ shown in the objective function of (2.3).

Remark 3.8: The $f_k(c)$ in our hardware implementation is not limited to the formula given in (2.1), which is simply an example formula for the purpose of comparisons. However, we have to admit that once a function or a form of $f_k(c)$, which may correspond to certain *coding* and *modulation* schemes, is assumed, changing hardware is not as easy as the software.

In all simulations presented in this section, we set $P_e = 10^{-4}$ for each user, and the wireless channel is modeled as a frequency-selective channel consisting of six independent Rayleigh multipaths. Each multipath is modeled by Clark's flat fading model [25]. We assumed the delays and the corresponding gains of the six paths are $100 \cdot p$ nanosecond and e^{-2p} (exponentially decay), respectively, where p =0, 1, 2, 3, 4 and 5 denote the multipath index. Hence, the relative power of the six multipath components are 0dB, -8.69 dB, -17.37 dB, -26.06 dB, -34.74 dB, and -43.43dB. We also assume the average subcarrier channel gain $E|\alpha_{k,n}|^2$ is unity for all k and n. Based on the above assumptions, we generate power consumption coefficients $\alpha_{k,n}, k=1,...,K, n=1,...,N,$ using MATLAB for our simulations.

We consider various number of users by setting K=2, 4, 8, 16, and 32. For each K, we randomly generate 500 sets of $\alpha_{k,n}$, $k=1,...,K$, $n=1,...,N$, based on the above mentioned power consumption coefficient generation process and denote α^i as the ith set in the 500. We assume a fixed total requested data rate R_T (=512 bits/symbol) and randomly generate R_k, $k=1,...,K$, based on the constraint $\sum_{k=1}^{K} R_k = R_T$. By the above test setup, we have run our approach for each K, each set of $\alpha_{k,n}$, $k=1,...,K$, $n=1,...,N$, and each set of R_k, $k=1,...,K$. We also apply the four methods mentioned at the beginning of this section to the same test.

For each K associated with the data rate request R_k, $k=1,...,K$, we denote $abSNR(\alpha^i)$ as the **a**verage **b**it **SNR**[4] when α^i is used and calculate $\frac{\sum_{i=1}^{500} abSNR(\alpha^i)}{500}$, the average $abSNR$,

[4] It is noted that the average required transmit power (in energy per bit) is defined as the ratio of the overall transmit energy per OFDM symbol, the P_T in (2.1), to the total number of bits transmitted per OFDM symbol, which consists of 512 bits in our test case. Moreover, we define the average bit Signal-to-Noise Ratio (SNR) as the ratio of the average transmit power, $\frac{P_T}{512}$, to the noise PSD level N_0. As we have assumed that all the data rates per symbol are fixed at 512, and the N_0 is just a constant, thus the P_T is proportional to the average bit SNR. Therefore, for the purpose of comparison, we can use the average bit SNR to replace P_T.

resulted from the 500 α^i's using our approach and the other four methods and report them in Figure 3.4. We can see that the $\frac{\sum_{i=1}^{500} abSNR(\alpha^i)}{500}$ obtained by our approach, which are marked by "△" in Figure 3.4, is smallest among all methods. Moreover, the performance of our approach is even better as the number of users increases as can be observed from Figure 3.4.

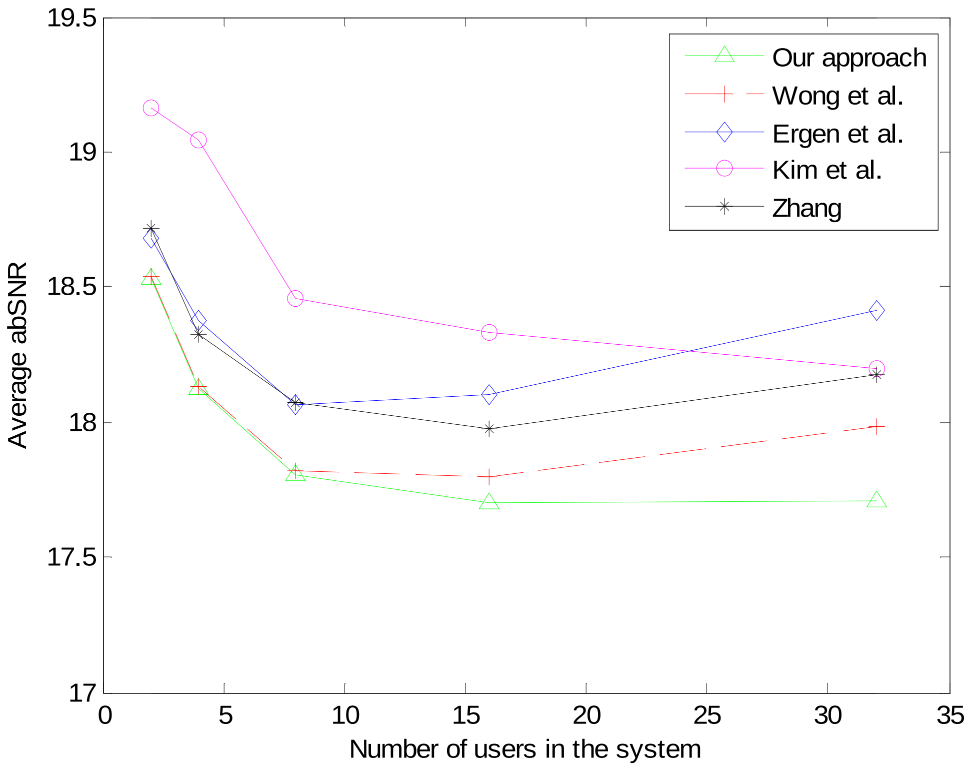


Figure 3.4. The $\overline{abSNR}$ for K=2,4,8,16, and 32 obtained by the five methods.

In previous comparisons, we have set $P_e = 10^{-4}$. It would be interesting to know how will the QoS requirement affect the performance of our approach. Therefore, we have tested the performance of the five methods for various K and various P_e ranging from 10^{-1} to 10^{-6} using randomly generated 500 sets of $\alpha_{k,n}, k = 1,\ldots,K, n = 1,\ldots,128,$ for each K and each P_e. The conclusions on the performance for various K are similar. A typical one is shown in Figure 3.5, which corresponds to K=7. The average of the 500 average bit SNR for various P_e obtained by our approach is marked by "△" in Figure 3.5. We can see that the performance of our approach is the best among the five, and when the QoS level is required higher (i.e. the value of P_e is smaller), the performance of our approach is even better (i.e. smaller average of the average bit SNR compared with the other four methods).

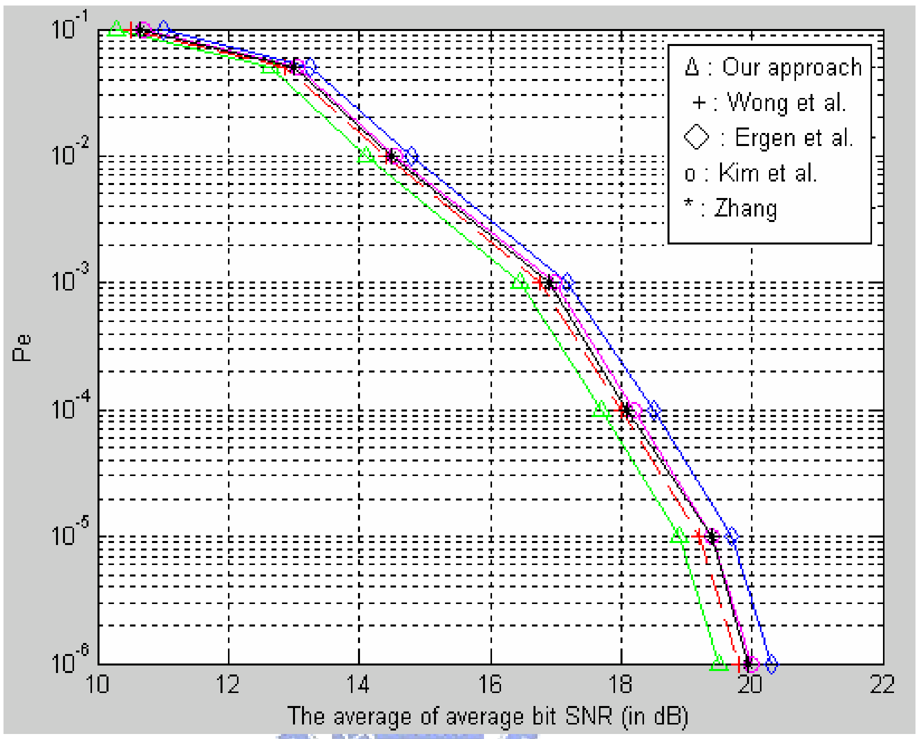


Figure 3.5. Comparison of the performance of the five methods with respect to various P_e for the case of *K*=7.

To investigate the computational efficiency of our approach and the other four methods, we need to report the average computation time for obtaining the $abSNR(\alpha^i)$. However, as we have previously indicated that the DPG method will be implemented by integrated circuits, the computation time of our approach is partly real and partly estimated, and its details are stated in the following.

All the computation time of our OO theory based four-stage approach except for the DPG algorithm are recorded in the employed Pentium 2.4 GHz processor and 512 Mbytes RAM PC, and we denote it by T_R. For *K*=2, 4, 8, 16, and 32 in the test results shown in Figure 3.4, the corresponding average T_R for obtaining an $abSNR(\alpha^i)$ are 1.214 ms, 1.686 ms, 3.386 ms, 5.366 ms, and 11.136 ms, respectively. To estimate the computation time of the DPG method, we base on 90-nm CMOS integrated circuit technology and denote this estimated computation time as T_{E}. We let $\otimes$, $\oplus$ and ROM denote the operations of a multiplication, addition, and accessing the data of a ROM, respectively, and define $T_{(\cdot)}$ as the computation time for performing the operation $(\cdot)$. Referring to the work of Hsu, S.K. *et al.* [26] and Kanan R. *et al.* [27], $T_{\otimes} = 1.0$ ns for a 16x16 bit multiplication[5], and it takes 1.2 ns for accessing the data of a ROM. In practical designs, the circuit complexity of a 16x16 bit multiplication is five times greater than a 16+16 bit addition [28, Ch. 5, p. 113, and Ch. 13, p. 433], thus we can set $T_{\oplus} \approx 0.2$ ns. Then, based on the last column of Table 3.1, we have $T_{\mathrm{PE}_{\mathrm{n}}^{1}} = 6.6$ ns, $T_{\mathrm{PE}_{\mathrm{n}}^{2}} = 2.2$ ns, $T_{\mathrm{PE}^{4}} = 1.6$ ns, $T_{\mathrm{PE}^{5}} = 1.2$ ns, and $T_{\mathrm{PE}^{7}} = 1.0$ ns. In our simulations, the values of $t_{\max} \times j_{\max}$ are set to be 8000, 10000, 12000, 15000, and 18000 for cases of $K = 2$, 4, 8, 16, and 32, respectively. Thus based on (3.15) and (3.16), the estimated computation time, T_{E}, of the DPG method is 0.208 ms, 0.52 ms, 1.248 ms, 3.12 ms, and 7.488 ms for cases of $K = 2$, 4, 8, 16, and 32, respectively. Summing up T_{R} and T_{E}, the estimated average computation time of our approach for obtaining an $abSNR(\alpha^{i})$ for various K are reported in the second row of Table 3.2. We also report the average computation time of the other four methods on the same test case in rows 3-6 of Table 3.2. The method proposed by Wong *et al.* is most computation-time consuming as have been indicated in [5]-[9]. Considering that the frame length of a wideband OFDM is 20 ms [29], the proposed approach can meet the real-time application requirement for high mobility circumstances.

Table 3.2
Average computation time (ms) for obtaining an $abSNR(\alpha^{i})$ for various number of users

Computation time (ms) \ K / Method	2	4	8	16	32
Our approach	1.42	2.21	4.63	8.49	18.63
Wong *et al.*	103.32	185.3	371.3	701.2	1507.1
Ergen *et al.*	10.18	14.9	18.8	31.1	53.2
Kim *et al.*	24.95	30.5	40.6	96.6	225.9
Zhang	26.81	42.5	45.3	60.3	88.1

[5] A question may be raised that whether 16-bit data type has enough precision for implementation. The answer is yes, because the resulted $\rho_{k,n}^{*}$ that we need from the hardware computation is whether $\rho_{k,n}^{*}$ is zero or nonzero but not how accurate the nonzero value is.

As demonstrated above, our approach outperforms the other four methods in the aspect of power consumption, and we can obtain the results in real-time.

Remark 3.9: It seems not fair that the computation time of our algorithm are partly estimated from hardware performance, while the computation time of other algorithms are entirely from the computer simulation. In fact, what we want to assert is we can achieve the best performance among all methods (the comparisons resulted in Figure 3.4) in real-time (the data shown in the second row of Table 3.2).

Remark 3.10: As we have indicated in *Remark* 3.2 that the DPG method is simple, hence it takes more iterations to converge. However, the key that it can help speed up our approach is its hardware implementability, and this is the reason why we estimate its computation time based on the hardware architecture rather than the commonly adopted expression.

Remark 3.11: In the deep submicron technology, the effect of the wire delay is prominent, especially in the design of large area or complicated routing. There are two types of wire delay in our hardware architecture, the intra and inter wire delay. The intra wire delay is the wire delay inside the hardware component of a PE, such as the multiplier. The inter wire delay is the wire delay between PEs and registers of the hardware architecture shown in Figure 3.3. The intra wire delay plays a dominant role in the overall wire delay, however they had been taken into account in our estimation of computation time. Since our hardware architecture is very regular and modular, the inter wire delay can at most be a small fraction of T_{E}, the estimated computation time of the DPG algorithm, and will not affect the real-time application.

To evaluate the actual goodness of the obtained good enough solutions, we should compare with the optimal solution of (2.3) using extensive simulations. To cover more system conditions in our simulations, we consider the cases of *N*=32, 64, and 128 and take four various *K* for each *N*. We define Average Bit Per Subcarrier (ABPS) as $\frac{\sum_{k=1}^{K} R_k}{N}$ to denote the congestion condition of the system. We set *M*=6 and consider three cases of ABPS, ABPS=3, 4 and 5, for each *N* and each *K*. For each (*N*, *K*, ABPS), we randomly generate 250 sets of $R_k, k = 1,...,K,$ based on the constraint on ABPS, and randomly generate a set of

$\alpha_{k,n}, k = 1,\ldots,K, n = 1,\ldots,N,$ for each set of $R_k, k = 1,\ldots,K$. We employ (2.1) for the $f(c)$ but set $P_e = 10^{-4}$ and $N_0 = 1$. Table 3.3 shows the average of 250 $\frac{d-D}{d}\times 100\%$ for each (*N*, *K*, APBS), where *D* and *d* denote the actual optimal power consumption of (2.3) and the power consumption of good enough solution obtained by our approach, respectively. For each ABPS, the average of the average $\frac{d-D}{d}\times 100\%$ of various *N* and *K* is shown in the last row of Table 3.3, which indicates the average deviation of *d* from *D* is around 1.0% in various congestion condition of the system. This shows that the good enough solutions we obtained are really good enough.

Table3.3

The average $\frac{d-D}{d}\times 100\%$ for each set of N, K and ABPS

N	K	Average $\frac{d-D}{d}\times 100\%$		
		ABPS =3	ABPS =4	ABPS =5
32	4	0.246	0.300	0.283
32	6	0.646	1.260	0.874
32	8	1.606	1.634	1.115
32	10	1.837	1.778	1.568
64	4	0.198	0.114	0.131
64	8	0.557	0.422	0.241
64	12	1.872	1.663	1.295
64	16	2.227	2.328	2.090
128	4	0.056	0.081	0.080
128	8	0.131	0.110	0.144
128	16	0.501	0.863	0.852
128	32	2.572	2.842	2.793
Average		1.037	1.116	0.956

3.4 Concluding Remarks

In this chapter, we have proposed an OO theory based four-stage approach to solve the adaptive subcarrier assignment and bit allocation problem of multiuser OFDM system for a good enough feasible solution. To resolve the computational complexity problem caused by the DPG method in our approach, we propose a hardware architecture to implement the DPG method so as to exploit deep submicron technology. Comparing with some existing methods, the quality of the good enough feasible solution obtained by our approach is excellent, and the estimated computation time meets the real-time application requirement.

Chapter 4

A Computationally Efficient Method for Large Dimension Subcarrier Assignment and Bit Allocation Problem of Multiuser OFDM System

In multiuser OFDM communication system, the increasing dimension will (i) adverse the computational complexity of the already time consuming mathematical programming based approaches [3], [5] and (ii) enlarge the discrete solution space, which will degrade the quality of the solutions obtained by the more local-like approaches [6]-[9] as well as the corresponding computation time. Although the method presented in chapter 3 can result a good enough solution and can meet real-time application requirement, however, implementing the first stage in hardware is almost impossible for area concern due to the large-dimension ASABA problem. Thus, dealing with large-dimension ASABA problem based on software-like method is a challenging issue in wireless communication, and the *purpose* of our proposed second method is proposing a computationally efficient method to solve the considered problem for a *good enough* solution for large-dimension ASABA problem.

The quality of the solution obtained by the mathematical programming based approach [3] is considered to be one of the best so far. However, they arbitrarily round the optimal *continuous* subcarrier assignment pattern off to the closest discrete values may cause *infeasibility* problem and *not* guarantee to be a good solution, if feasible. To avoid the undesirable effect caused by rounding off, we will handle the discrete solution space directly and use a global-like approach. However, the global searching techniques [30] such as *Genetic Algorithm* (GA), *Simulated Annealing* method, *Tabu Search* method and *Evolutionary Programming* are not adequate here because of their tremendous computation time, which is even worse than the mathematical programming based approach due to (i) evaluating the *objective value* of a feasible subcarrier assignment pattern is time consuming, (ii) handling the constraints is not an easy task and (iii) the size of the discrete solution space is huge. Evaluating the exact performance (i. e. the objective value) of a feasible subcarrier assignment pattern is a conventional "*value*" concept. However, it is indicated in OO theory that the performance order of discrete solutions is likely preserved even evaluated by a *surrogate* *model*. In other words, the OO theory claims that there is high probability that we can find the *actual* good discrete solutions if we limit ourselves to the top *n*% of the estimated good discrete solutions evaluated by a surrogate model [12]. Thus, to retain the merit of global searching technique while avoiding the cumbersome conventional performance evaluation of a discrete solution, our approach is based on OO theory to solve the considered problem for a *good enough* solution with *high probability* using *limited* computation time.

The approach consists of three OO stages. First of all, we will reformulate the considered problem to separate it into subcarrier assignment and bit allocation problem such that the objective function of a feasible subcarrier assignment pattern is the corresponding optimal bit allocation for minimizing the total consumed power. Then, in the first stage, we will develop an *easy-to-evaluate approximate* objective function to *estimate* the objective value of a subcarrier assignment pattern and employ a GA to search through the huge discrete solution space to find the top s subcarrier assignment patterns based on the estimated objective values. In the meantime, a subtle *representation scheme* and a *repair operator* for GA need be designed to handle the constraints of the considered problem. In the second stage, we use an off-line trained ANN to estimate the objective values of the s subcarrier assignment patterns obtained in stage 1 and pick the top l patterns based on the estimated objective value. In the third stage, we use the exact objective function to evaluate the l subcarrier assignment patterns obtained in stage 2, and the best one associated with the corresponding optimal bit allocation is the good enough solution that we seek. In the proposed three-stage approach, the models employed to evaluate a solution are varying from very rough (stage 1) to exact (stage 3). In the meantime, the candidate solution space is reduced from the *original* huge solution space (stage 1) to only *l candidate solutions* (stage 3). In general, a more accurate approximate objective function will take more time to evaluate a solution; however as can be seen from our three-stage approach, when a more accurate approximate objective function is used, the search space is already reduced considerably, and the computation time is largely reduced accordingly.

We organize Chapter 4 in the following manner. In Section 4.1, we will reformulate the considered problem. In Section 4.2, we will present our three OO stages to solve the considered problem. In Section 4.3, we will apply our algorithm to numerous large-dimension ASABA cases and compare with some existing algorithms in the aspects of solution quality and computation time. Finally, we will draw a conclusion in Section 4.4.

4.1 Reformulation

We *assume* the solution of (2.3) exists, which is equivalent to the following *assumption*: there are enough subcarriers to meet the data rate request of all users, i.e. the following inequality hold

$$\sum_{k=1}^{K} \left\lceil \frac{R_k}{M} \right\rceil \le N \tag{4.1}$$

because a subcarrier cannot be shared by more than one user. Notation $\lceil y \rceil$ in (4.1) denotes the integer closest to y on the right-hand side.

Remark 4.1: (i) The situation that the solution of (2.3) does not exist, i.e. assumption (4.1) does not hold is beyond the scope of this dissertation. (ii) For the extreme case that $\sum_{k=1}^{K} \left\lceil \frac{R_k}{M} \right\rceil = N$ and the spectrum is *fixed* for each user, (2.3) becomes a very simple optimal bit allocation problem and can be readily solved by the existing greedy algorithm [20]. However, methods proposed in this dissertation and [3-9] aim to solve ASABA problem, (2.3), which is much more complicated than the above mentioned extreme case.

To develop an approximate objective function for a subcarrier assignment pattern, we need to reformulate (2.3). We let ρ , c , α and R denote the vectors $[\rho_{k,n}],[c_{k,n}],[\alpha_{k,n}], k=1,\ldots,K, n=1,\ldots,N,$ and $[R_k], k=1,\ldots,K,$ respectively, and define $C(\rho)$, *the feasible set of bit allocation c for a given* ρ , as

$$C(\rho)=\{c \mid 0 \le c_{k,n} \le M \text{ and } R_k = \sum_{n=1}^{N} \rho_{k,n} c_{k,n}, k=1,\ldots,K, n=1,\ldots,N, \text{for a given } \rho\} \tag{4.2}$$

Now for a given subcarrier assignment pattern ρ that satisfies $\sum_{k=1}^{K} \rho_{k,n} = 1,\ n=1,\ldots,N$ and $\rho_{k,n} \in \{0,1\}$ for all k and n, (2.3) becomes an *optimal bit allocation problem* under the given ρ that is to find the optimal c of the following problem:

$$\min_{c} \sum_{n=1}^{N}\sum_{k=1}^{K} \frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k(c_{k,n})$$

subject to $$c \in C(\rho)$$

$$C(\rho) \neq \phi \qquad (4.3)$$

where the constraint $C(\rho) \neq \phi$ represents the existence of feasible bit allocation c for the given ρ. Note that the assumption, (4.1), of our problem does not imply $C(\rho) \neq \phi$ for any ρ that satisfies $\sum_{k=1}^{K} \rho_{k,n} = 1,\ n = 1,\ldots,N$ and $\rho_{k,n} \in \{0,1\}$; for example, in an extreme case that $\rho_{1,1} = \rho_{1,2} = \ldots = \rho_{1,N} = 1$ and the rest of $\rho_{k,n} = 0$, then if (4.1) is satisfied but $R_k \neq 0$ for any $k \neq 1$, we have $C(\rho) = \phi$ for the given ρ. However, the assumption (4.1) guarantees that there exists ρ such that $C(\rho) \neq \phi$. In fact, the constraint $C(\rho) \neq \phi$ is equivalent to the inequality constraints $R_k \leq M \sum_{n=1}^{N} \rho_{k,n},\quad k = 1,\ldots,K$. Thus, we can rewrite (2.3) into the following form:

$$\min_{\rho}\{\min_{c} \sum_{n=1}^{N}\sum_{k=1}^{K} \frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k(c_{k,n}) \mid c \in C(\rho)\}$$

subject to $$R_k \leq M \sum_{n=1}^{N} \rho_{k,n}, k = 1,\ldots,K\,,$$

$$\sum_{k=1}^{K} \rho_{k,n} = 1,\ n = 1,\ldots,N\,,$$

$$\rho_{k,n} \in \{0,1\}\text{, for all } k \text{ and } n \qquad (4.4)$$

(4.4) can be viewed as a *separation* of *subcarrier assignment* and *bit allocation* problem, because the optimization problem inside the big bracket is the *optimal bit allocation* problem for a given *feasible* ρ that satisfies all the constraints in (4.4), and the overall problem is finding the best feasible ρ associated with an optimal bit allocation. Furthermore, the optimal bit allocation problem is separable for a given feasible ρ, because its objective function $\sum_{n=1}^{N}\sum_{k=1}^{K} \frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k(c_{k,n})$, as well as the constraints $c \in C(\rho)$ are separable. Thus we can decompose it into the following K independent subproblems: For k=1,…,K,

$$\min_{c_{k,1},\ldots,c_{k,N}} \sum_{n=1}^{N} \frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k(c_{k,n})$$

subject to $$0 \le c_{k,n} \le M, n = 1,\ldots,N,$$

$$R_k = \sum_{n=1}^{N} c_{k,n} \qquad (4.5)$$

Clearly (4.4) is a constrained combinatorial optimization problem with (i) hard to evaluate objective function, because the objective function $\{\min \sum_{n=1}^{N}\sum_{k=1}^{K} \frac{\rho_{k,n}}{\alpha_{k,n}^2} f_k(c_{k,n}) \mid c \in C(\rho)\}$ itself is an optimization problem, (ii) equality and inequality constraints involving integers and (iii) huge discrete solution space for ρ as long as K and N are large.

Remark 4.2: As indicated previously, the assumption (4.1) is to assume that the solution of (2.3) exists. Since (4.4) is equivalent to (2.3), the assumption (4.1) should also apply to (4.4) to assume the solution of (4.4) exists. It should be noted that no additional assumption is needed to derive (4.4) from (2.3). Furthermore, the decomposition of the objective function of (4.4), i.e. the term inside the big bracket of (4.4), into (4.5) has nothing to do with the assumption (4.1). This decomposition simply says that for a given feasible ρ, the optimal bit allocation for individual user is independent of each other.

4.2. The Three-Stage Ordinal Optimization (OO) Approach

The proposed three-stage OO based approach for solving (4.4), or (2.3) equivalently, consists of three OO stages as stated in the following.

4.2.1 Stage 1: Using GA to Select Top *s* Subcarrier Assignment Patterns Based on an Easy-to-Evaluate Approximate Objective Function

As shown in (4.5) that the objective function of (4.4) for a given feasible ρ can be decomposed into K independent optimal bit allocation subproblems. We let $N_k = \sum_{n=1}^{N} \rho_{k,n}$ and $\hat{\alpha}_k = \frac{\sum_{n=1}^{N} \rho_{k,n}\alpha_{k,n}}{N_k}$ denote the total number of subcarriers assigned to and the average power consumption coefficient of user k, respectively. Then, we use the following to *approximate* the optimal power consumed by user k, i.e. the optimal objective value of (4.5), for the given ρ.

We assume the total data rate request R_k of user k are distributed *equally* to the assigned N_k subcarriers, i.e. setting $c_{k,n} = \frac{R_k}{N_k}$ for each assigned subcarrier, and assume each of the N_k subcarriers has the same power consumption coefficient $\hat{\alpha}_k$ defined above, then the power consumed by user k, denoted by $\hat{P}_k$, can be computed by $\hat{P}_k = \frac{N_k}{\hat{\alpha}_k^2} f_k(\frac{R_k}{N_k})$. Consequently, we can obtain the *approximate* total consumed power for the given ρ, denoted by $\hat{P}_T$, by calculating $\hat{P}_T = \sum_{k=1}^{K} \hat{P}_k$, which will serve as the *approximate* objective function of (4.4).

Then, to use GA as a global searching technique, we need to define a *representation scheme* to map all ρ that satisfy $\rho_{k,n} \in \{0,1\}$ and $\sum_{k=1}^{K} \rho_{k,n} = 1$ into a set of *chromosomes* first [30], [31, Ch.14]. Let the *alphabet* of the representation scheme be the set $\{1,2,\ldots,K\}$. We define the chromosome u as a string of N symbols, $u_1, u_2, \ldots, u_N$, such that the nth symbol u_n, which takes an element from the alphabet, indicates the user that subcarrier n is assigned to. In other words, $u_n = k_1$ means $\rho_{k_1,n} = 1$ and $\rho_{k,n} = 0$ for all $k \neq k_1$. This representation scheme ensures that $\rho_{k,n} \in \{0,1\}$ and satisfies $\sum_{k=1}^{K} \rho_{k,n} = 1$, because u_n taking only one element from the alphabet implies that the nth subcarrier can at most be assigned to one user. However not all chromosomes u can satisfy the inequality constraints $M \sum_{n=1}^{N} \rho_{k,n} \geq R_k$ required in (4.4). The required number of subcarriers for user k to meet the inequality constraint is at least $\left\lceil \frac{R_k}{M} \right\rceil$. We define $\delta_k(u_n) = 1$ if $u_n = k$ and 0, otherwise. Thus, the number of subcarriers assigned to user k in a chromosome u is $\sum_{n=1}^{N} \delta_k(u_n)$. To meet the inequality constraint, the following has to hold

$$\sum_{n=1}^{N} \delta_k(u_n) \geq \left\lceil \frac{R_k}{M} \right\rceil \qquad (4.6)$$

We define $\sigma_k(u) = \sum_{n=1}^{N} \delta_k(u_n) - \left\lceil \frac{R_k}{M} \right\rceil$, then $\sigma_k(u) \geq 0$ implies the number of subcarriers assigned to user k is enough or surplus, and $\sigma_k(u) < 0$ implies the other way. Therefore, for a given u we can compute $\sigma_k(u)$ for $k = 1,\ldots,K$ and order them in an ascending sequence $\sigma_{k_1}(u) \leq \sigma_{k_2}(u) \leq \ldots \leq \sigma_{k_K}(u)$, where the ordered indices $k_1,\ldots,k_K \in \{1,\ldots,K\}$, and we have no order preference for the k's with same values of $\sigma_k(u)$. Thus $\sigma_{k_1}(u) \geq 0$ implies u satisfies (4.6) for all k and is feasible.

Suppose $\sigma_{k_1}(u) < 0$, then u is infeasible. Such an *infeasibility* problem may occur to any newly generated chromosomes, resulted from *initial population* generation, *crossover* operations, and *mutation* operations, and can be resolved by a *repair operator*, which is designed to recover the infeasible chromosome to a feasible one as stated in the following. Under the *assumption* that we have enough subcarriers to meet all users' data rate request, (4.1), if $\sigma_{k_1}(u) < 0$, there must exist i such that $\sigma_{k_1}(u) \leq \ldots \leq \sigma_{k_i}(u) < 0$, $\sigma_{k_K}(u) \geq \ldots \geq \sigma_{k_{i+1}}(u) \geq 0$, and $\sigma_{k_{i+1}}(u) + \ldots + \sigma_{k_K}(u) \geq -(\sigma_{k_1}(u) + \ldots + \sigma_{k_i}(u))$. Thus, we can reassign the surplus subcarriers of users $k_{i+1}, \ldots, k_K$ to users $k_1, \ldots, k_i$ in the following manner. Randomly pick the surplus subcarriers that were assigned to user k_K to make up the insufficient subcarriers required by k_1, that is randomly pick an u_m from all u_m's with $u_m = k_K$, and reset the picked u_m as $u_m = k_1$. When all surplus subcarriers of user k_K are reassigned, we proceed with picking the surplus subcarriers of user k_{K-1} and so forth. Similarly, when the number of insufficient subcarriers of user k_1 is made up, we proceed with making up the insufficient subcarriers of user k_2 and so forth. The above process will continue until the number of insufficient subcarriers of user k_i is made up. Then the resulting u will be feasible. Based on this repair operator, we may describe the employed GA to solve (4.4) in the following.

We randomly generate I, say 200, chromosomes such that each symbol of each chromosome is assigned by an element randomly selected from the alphabet, $\{1, \ldots, K\}$, and apply the repair operator to them. The resulting I feasible chromosomes will serve as the initial population of the employed GA. To evaluate the *fitness* of a chromosome u based on the above mentioned *approximate* total power consumption, we first compute $N_k(u) = \sum_{n=1}^{N} \delta_k(u_n)$, $\hat{\alpha}_k(u) = \frac{\sum_{n=1}^{N} \delta_k(u_n)\alpha_{k,n}}{N_k(u)}$, and $\hat{P}_k(u) = \frac{N_k(u)}{\hat{\alpha}_k^2} f_k(\frac{R_k}{N_k(u)})$ for every $k=1,\ldots,K$. Then, the fitness of u will be $\frac{1}{\hat{P}_T(u)}$, where $\hat{P}_T(u) = \sum_{k=1}^{K} \hat{P}_k(u)$. Based on the fitness values of all chromosomes in the *population* pool, we use *roulette wheel selection scheme* to select chromosomes into the *mating* pool, from which we select chromosomes to serve as the *parents* for crossover. The probability that a chromosome is selected as a parent is p_c, say 0.7.

We apply a *single point crossover scheme* to the selected parents, and the generated *offsprings* may be infeasible as indicated previously. Therefore we will apply the repair operator to each generated offspring, and the resulting feasible offsprings will replace the corresponding parents in the mating pool. Subsequently, we will apply the *mutation* operation to each chromosome in the mating pool with *mutation probability* p_m, say 0.02. Any changed chromosome after mutation operation may also be infeasible, and we will apply the repair operator to it. Consequently, the resulting chromosomes in the mating pool after the above *evolution process* will be the population pool of next iteration.

The above process completes one iteration of our GA. We stop the GA when the number of iterations exceeds 60. After the applied GA converges, we rank the final I chromosomes (i.e. u's) based on their fitness values and pick the best s (=50) u's. We can then convert these s chromosomes into the subcarrier assignment patterns ρ's in the following manner: $\rho_{k,n} = \delta_k(u_n)$ for all k and n. Then these converted ρ's are the s subcarrier assignment patterns determined in this stage, and they are feasible for (4.4).

Remark 4.3: Based on [12], larger s will consist of more actual good subcarrier assignment patterns. However, larger s may cause more computation time for further evaluation. Thus, the value of s should be determined based on the available computation budget to obtain and the required goodness of the good enough solution. In the current application, the computation time is of more concern.

4.2.2 Stage 2: Choose Top l Subcarrier Assignment Patterns From the s Based on an ANN Model

Since evaluating the s ρ's obtained in Stage 1 using the exact objective function is still too time consuming, based on [10], we can trim the candidate solution set further using a more accurate approximate objective function. Therefore, we will employ a *supervised learning* ANN [17] to estimate the optimal power consumed by user k and select top l (=3) ρ's from the s.

Remark 4.4: The value of l is also determined based on a tradeoff between the computation time required to obtain and the goodness of the good enough solution.

This ANN is trained *off-line* using 5000 input/output pairs of data. The input data associated with user *k* is the data rate request R_k, the number of subcarriers assigned to user *k*, N_k, and the power consumption coefficient $\alpha_{k,n_i}, i=1,...,N_k$, for the assigned N_k subcarriers $n_1,...,n_{N_k}$. However, the dimension of the vector $(R_k, \alpha_{k,n_1},..,\alpha_{k,n_{N_k}})$ is large provided that N_k is large. A large ANN, i.e. an ANN consisting of large number of neurons in both input and hidden layers, will consume more computation time to obtain the output even if it is trained off-line. Although larger ANN can serve as a more accurate function approximator, what we care here is the performance order rather than the performance value. Therefore, for computation-time concern, we favor a simpler ANN. Since the values of $\alpha_{k,n_1},..,\alpha_{k,n_{N_k}}$ may have some kind of distribution, to characterize these values without using the details, we may use the corresponding mean, $\hat{\alpha}_k$, and variance, $\text{var}(\alpha_k)$ [32]. Thus, to design a simple ANN, we will employ $(R_k, N_k, \hat{\alpha}_k, \text{var}(\alpha_k))$ to characterize the input data of user *k*.

Consequently, the 5000 input/output pairs used to train the ANN can be obtained as follows. We uniformly select 5000 sets of (R_k, N_k) from the following ranges: $R_k \in [5,150]$, $N_k \in [\frac{R_k}{M}, \frac{2R_k}{M}]$, which makes $R_k \le MN_k$ to ensure that there are enough subcarriers to meet the data rate request. For each N_k, we randomly select $n_1,..,n_{N_k}$ from {1,…,*N*}, then randomly generate α_{k,n_i} from the range [0,2.0] for each $i=1,...,N_k$. The $\hat{\alpha}_k$ and $\text{var}(\alpha_k)$ can be computed accordingly. Thus, we have 5000 input vectors of $(R_k, N_k, \hat{\alpha}_k, \text{var}(\alpha_k))$. Now for each $(R_k, N_k, \alpha_{k,n_1},...,\alpha_{k,n_{N_k}})$, the corresponding output data for training ANN is the *actual* optimal power consumed by user *k* denoted by P_k. To compute P_k, we can use the greedy algorithm [20] to solve (4.5) by optimally distributing R_k bits to the assigned N_k subcarriers one at a time based on the *least incremental power consumption criteria*. We then use the obtained 5000 input/output pairs, $((R_k, N_k, \hat{\alpha}_k, \text{var}(\alpha_k)), P_k)$, of data to train a three-layer ANN whose structure is described in the following. The input layer consists of four neurons corresponding to R_k, N_k, $\hat{\alpha}_k$, and $\text{var}(\alpha_k)$. The hidden layer consists of 15 neurons, and each neuron uses *hyperbolic tangent sigmoid* as the activation function. There is only 1 neuron in the output layer corresponding to P_k, and we use *linear* as the activation function. Using the 5000 input/output pairs of data, we train this ANN by adjusting its arc weights using the Levenberg-Marquardt method proposed in [18] and [19]. Based on this off-line trained ANN, we can estimate the total consumed power corresponding to a ρ as follows. For a given ρ, we can compute $N_k = \sum_{n=1}^{N} \rho_{k,n}$ and determine $\alpha_{k,n_1},...,\alpha_{k,n_{N_k}}$ from the gain α for each k=1,…,K. Subsequently, we can set up the input $(R_k, N_k, \hat{\alpha}_k, \mathrm{var}(\alpha_k))$, feed into the off-line trained ANN, and obtain the *estimated* P_k, denoted by $\tilde{P}_k$, from the output of ANN for each k=1,…,K. Then we can compute the estimated total consumed power, denoted by $\tilde{P}_T$, for the given ρ by $\tilde{P}_T = \sum_{k=1}^{K} \tilde{P}_k$. Using this off-line trained ANN, the l (=3) ρ's with smallest $\tilde{P}_T$ among the s feasible ρ's obtained in Stage 1 are the subcarrier assignment patterns determined in this stage.

4.2.3 Stage 3: Determine the Good Enough Subcarrier Assignment and Bit Allocation

Since there are only l (=3) candidate feasible-ρ's left, we can use the exact objective function of (4.4) to calculate the objective value of each ρ. That is to solve the optimal bit allocation problem (4.5) for the given ρ using the greedy algorithm mentioned above to obtain the optimal power consumption P_k for user k=1,…,K. Then we calculate $P_T = \sum_{k=1}^{K} P_k$ for the given ρ. Consequently, the ρ associated with the optimal bit allocation corresponding to the smallest P_T among the l feasible ρ's will be the good enough solution of (2.3) that we look for.

4.3 Test Results and Comparisons

In this section, we will demonstrate the performance of the proposed algorithm on solving the large-dimension ASABA problem (4.4), which is equivalent to (2.3), in the aspects of solution quality and computational efficiency by comparing with other algorithms. We assume the OFDM system has 256 subcarriers (i.e. N=256), which can carry two, four, and six bits/symbol; therefore in this system M=6. We adopt the approximate formula in (2.1) for the $f_k(c)$ in the transmission power $\frac{f_k(c)}{\alpha_{k,n}^2}$ shown in the objective function of (2.3), and we set $P_e = 10^{-4}$ and $N_0 = 10^{-12}$ watt in the following simulations.

We use a frequency-selective channel consisting of six independent Rayleigh multipaths to model the wireless transmission channel, and each multipath is modeled by Clark's flat fading model [25]. We assumed that the power delay profile is exponentially decaying with e^{-2p}, where p =0, 1, 2, 3, 4 and 5 denote the multipath index. Hence, the related power of the six multipath components are 0dB, -8.69 dB, -17.37 dB, -26.06 dB, -34.74 dB, and -43.43dB. We also assume the average subcarrier channel gain $E|\alpha_{k,n}|^2$ is unity for all k and n. Based on the above assumptions, we can generate power consumption coefficients $\alpha_{k,n}$, k=1,…,K, n=1,…,N, using MATLAB for our simulations.

We consider cases of various number of users for K=10, 20, 30, 40, and 50. For each K, we assume a fixed total data rate request R_T =1024 bits/symbol and randomly generate R_k, $k=1,\ldots,K$, based on the constraint $\sum_{k=1}^{K} R_k = R_T$. For each K and the associated R, we randomly generate 5000 sets of $\alpha_{k,n}$, $k=1,\ldots,K$, $n=1,\ldots,N$, based on the above mentioned power consumption coefficient generation process and denote α^i as the ith set in the 5000. With the above test setup, we apply our algorithm to solve (2.3) on a Pentium 2.4 GHz processor and 512 Mbytes RAM PC. We also apply the more global-like mathematical programming based approaches proposed by Wong *et al.* and Kim *et al.* in [3] and [5], respectively, and the more local-like two-module scheme and two-step subcarrier assignment approaches proposed by Ergen *et al.* and Zhang in [6] and [8], respectively, to the same test cases on the same PC. For the purpose of comparison, we can use the average bit SNR (*abSNR*) to replace P_T, because *abSNR* is defined as the ratio of the average transmit power, $\frac{P_T}{R_T}$, to the noise PSD level N_0. As we have assumed that all the data rates per symbol are fixed at R_T, and the N_0 is just a constant, thus P_T is proportional to *abSNR*.

Remark 4.5: As shown in (2.1), P_T consists of the term N_0. Therefore, the magnitude of N_0 employed in our tests is not relevant to the results of *abSNR*, because the term N_0 will be cancelled out as noted in the definition of *abSNR*.

For each K with the associated vector R, we denote $abSNR(\alpha^i)$ as the resulted $abSNR$ when α^i is used and calculate $\overline{abSNR} = \frac{\sum_{i=1}^{5000} abSNR(\alpha^i)}{5000}$, where $\overline{abSNR}$ denotes the average of the 5000 $abSNR$'s for a given K. The resulted $\overline{abSNR}$ for each K and each algorithm are shown in Figure 4.1.

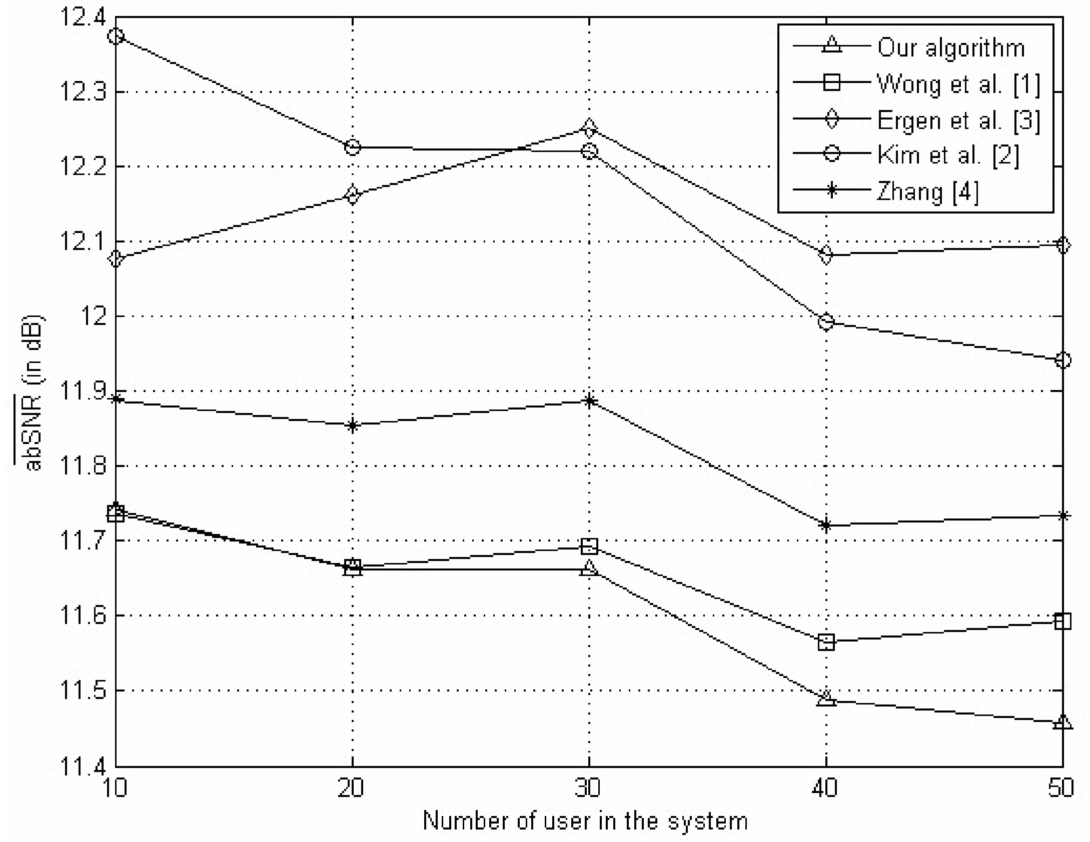


Figure 4.1. The $\overline{abSNR}$ for K=10, 20, 30, 40 and 50 obtained by the five algorithms.

Form Figure 4.1, we see that the $\overline{abSNR}$ obtained by our algorithm, which are marked by "△", is smallest among all algorithms. Moreover, the result obtained by our algorithm is even better when the number of users increases as can be observed from Figure 4.1.

Remark 4.6: The quality of the solution obtained by the approach proposed by Wong *et al.* in [3] is excellent and has been used as a comparing standard in most of the literature regarding ASABA problems [5], [7], [8]. We also manifest the quality of their solution in our simulations as shown in Figure 4.1. The reason that supports their solution's excellent quality is their global-like mathematical programming based approach as indicated previously. They first employed a Lagrangian relaxation method to solve the continuous version of the ASABA problem then rounded the optimal continuous subcarrier assignment solution off to the closest integer solution. Such an arbitrarily rounding off may cause possible *infeasibility* and not *theoretically* guarantee to obtain a good solution, especially when the dimension of the ASABA problem is large. Dislike their approach, we handle the discrete solution space directly. In the first stage of our approach, our specially designed GA, which associates with a surrogate model for fast fitness evaluation, search through the whole feasible solution space to find some good feasible subcarrier assignment patterns. Thus, our approach is also global-like and will not cause any infeasibility problem. Then in the second and third stages, we use the ANN and exact models, respectively, to help pinpoint a good enough subcarrier assignment pattern associated with optimal bit allocation among the feasible solutions resulted in Stage 1. The arbitrarily rounding off technique employed in [3] is lack of theoretical support. However, the foundation of our approach is OO theory, which is a theoretically sound general methodology [10] and has several successful applications on the combinational optimization problems with huge discrete solution space [24], [33]-[34].

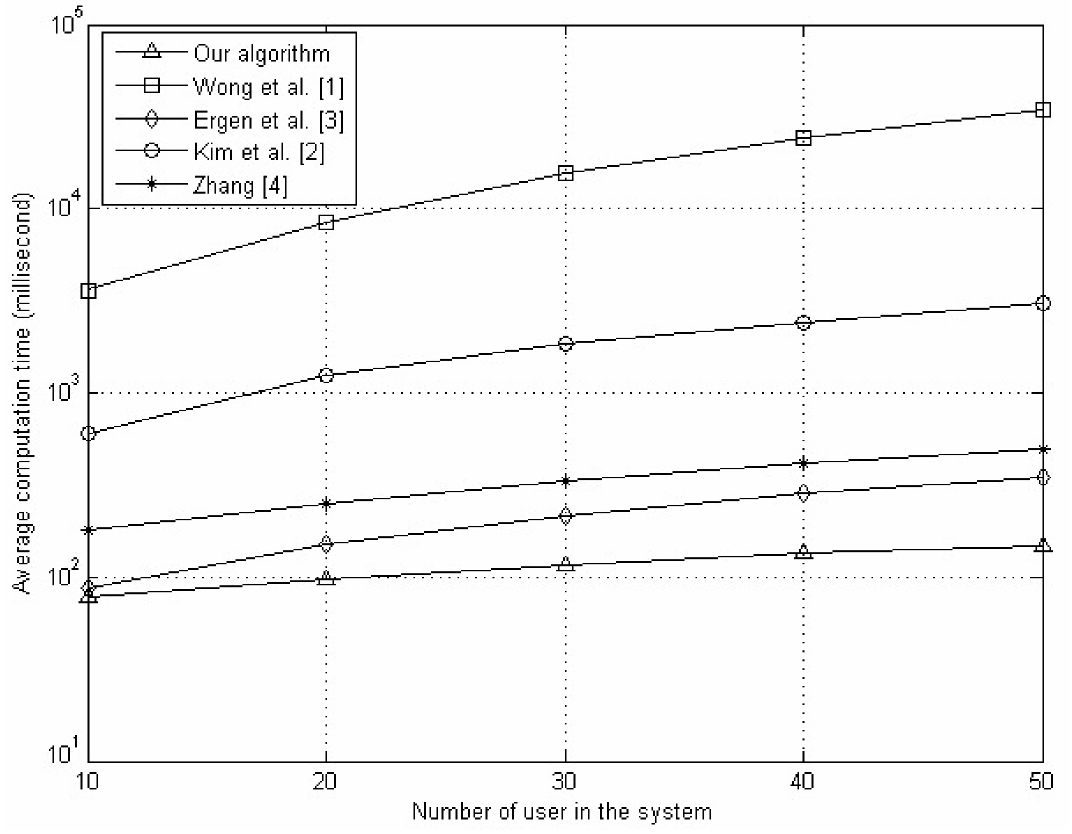


Figure 4.2. The average computation time for obtaining an *abSNR* by the five algorithms in cases of K=10, 20, 30, 40 and 50.

We also show the average computation time for obtaining an *abSNR* for each *K* and each algorithm in Figure 4.2. From this figure, we see that the average computation time obtained by our algorithm, which are around 100 milliseconds as marked by "△", is also smallest among all algorithms. These results show that our algorithm outperforms the other four in both aspects of solution quality and computational efficiency. More importantly, when the number of users increases, the performance of our algorithm is even better. This demonstrates that our algorithm is most suitable for large-dimension ASABA problems.

Remark 4.7: The methods in [5], [6], [8] are proposed to overcome the computational complexity of the method in [3]. Indeed, the methods in [6], [8] are more computationally efficient than the methods in [3], [5] as shown in Figure 4.2, because the former are local-like heuristic methods while the latter are global-like mathematical programming based approaches. In fact, the authors of [6] and [8] did not compare the computational efficiency of their methods with the method in [3] in their papers, because they take their methods being conceptually faster for granted. However, since the methods in [6], [8] are local-like methods, the computation time of each solution adjustment step is very short, but the improvement of the solution is limited. Hence their convergence rate will be degraded especially when the dimension of the ASABA problem is large. On the contrary, the computational complexity of our approach is less relevant to the size of the ASABA problem, because (i) the population size and number of iterations of the employed GA in stage 1 are fixed, (ii) the parameters *s* and *l* in stages 1 and 2, respectively, are fixed, and (iii) the structure of the ANN is also fixed. This is the reason why the computational efficiency of our algorithm can compete with the methods in [6], [8] in solving large-dimension ASABA problems. It is commonly understood that the comparisons based on CPU times may not be objective enough, however we can hardly obtain any analytical expression of the total consumed number of multiplications and additions of the methods in [3], [5], [6], [8]. In fact, the CPU time is a commonly used tool for the comparisons of computational efficiency in similar subjects appearing in [7], [35], [36].

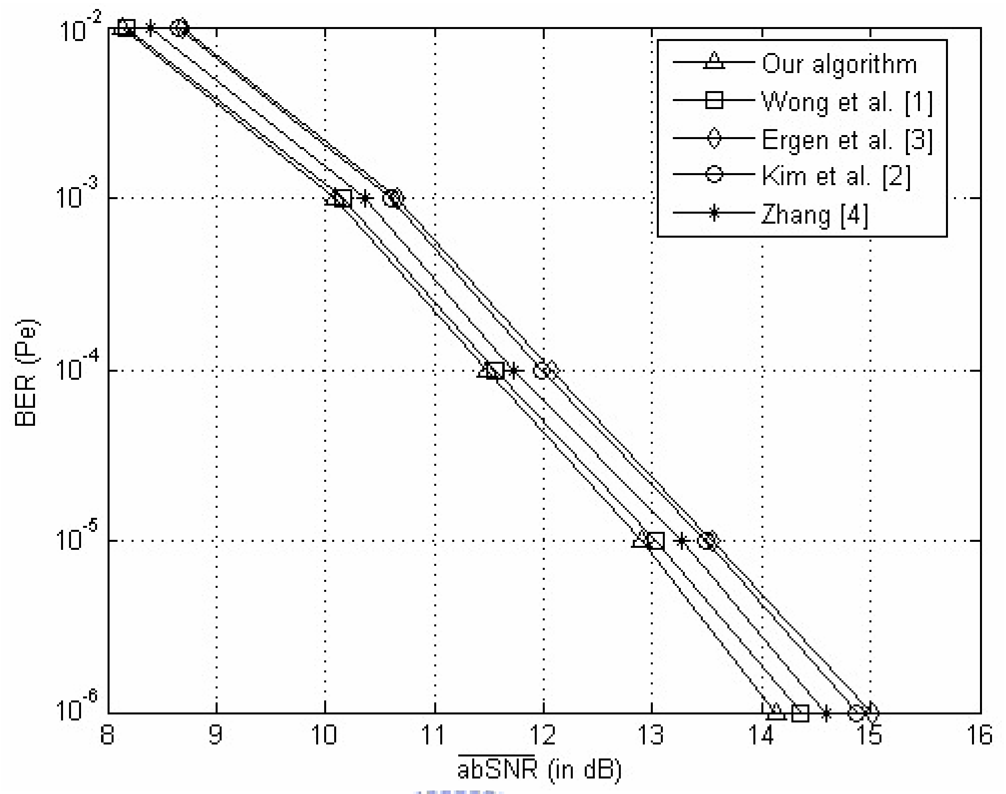


Figure 4.3. Comparison of the five algorithms for various P_e in the case of K=40.

In previous comparisons, we have set the BER, $P_e = 10^{-4}$. It would be interesting to know how will the Quality-of-Service (QoS) requirement, i.e. various BER, affect the performance of our algorithm. Therefore, we have tested the five algorithms for K=10, 20, 30, 40 and 50 with various P_e ranging from 10^{-2} to 10^{-6} using randomly generated 5000 sets of $\alpha_{k,n}$, k=1,…,K, n=1,…,256, for each K. The conclusions on the performance for the five algorithms for various K are similar. A typical one is shown in Figure 4.3, which corresponds to K=40. The $\overline{abSNR}$ obtained by our algorithm is marked by "△" in Figure 4.3. We see that the performance of our algorithm is the best among the five in all cases of P_e, and when the QoS level is required higher (i.e. the value of P_e is smaller), the performance of our algorithm is even better (i.e. smaller $\overline{abSNR}$ compared with the other four algorithms). This further demonstrates the superiority of the solution quality achieved by our algorithm.

4.4 Concluding Remarks

In this chapter, we have proposed a computationally efficient three-stage OO approach to solve the large-dimension ASABA problem of multiuser OFDM system for a good enough solution. By looking into the insight of the ASABA problem (2.3), we reformulate it into (4.4) and develop an approximate objective function as well as a subtle representation scheme and a repair operator for the GA employed in Stage 1, which makes our OO approach possible in handling the huge discrete solution space as well as the constraints. The easily computed surrogate models employed in Stages 1 and 2 help resolve the computation complexity caused by the hard-to-evaluate objective function. These factors contribute most to the computational efficiency of our algorithm. Furthermore, we have demonstrated the superiority of our algorithm by comparing with four existing algorithms through numerous test cases in the aspects of solution quality and computational efficiency. More importantly, our approach has wide range of applications in resource allocation problems of wireless network and communication.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Two multi-stage OO based methods to solve the ASABA problem of the multiuser OFDM system for a good enough solution have been presented and discussed. The first method can meet the real-time application requirement while with the assistance of hardware. The second method is computationally efficient for solving the large dimension ASABA problem of multiuser OFDM system.

The first method presented in Chapter 3 consists of four OO stages to find a good enough solution to the ASABA problem. In the first three stages, we use surrogate models to quickly evaluate the estimated performance of a solution so as to select an *estimated good enough subset* from the *candidate solution set* using limited computation time. When the size of the solution space is huge, the reduction of the search space can be done in several stages. The surrogate models in the stages can range from very rough to more refined ones, and the exact model will be employed in the last stage when there are only few solutions left in the candidate solution set. The four-stage OO approach ensures the quality of the obtained solution, however at the cost of solving a continuous version of the considered problem in the first stage. To resolve this computational complexity problem, we propose a hardware implementable DPG method to exploit deep submicron technology so as to obtain the optimal continuous solution extremely fast.

Due to the large dimension of the ASABA problem, implementing the first stage in hardware is almost impossible for area concern. Therefore in the first stage of our second method presented in Chapter 4, we develop an approximate objective function to evaluate the performance of a subcarrier assignment pattern and use a genetic algorithm to efficiently search through the huge solution space to find I (=200) good solutions.

Numerical results and comparisons with various existing algorithms are provided to demonstrate the potential of our proposed techniques. It is shown that the proposed resource allocation methods substantially improve the system power efficiencies. In the meantime, the proposed resource allocation algorithms are more computationally efficient. Moreover, the first method can meet the real-time application requirement and the second method is suitable for large dimensional ASABA problems.

5.2 Future Work

The proposed algorithms are based on the assumption that perfect channel information is available for adaptive resource allocation. In practice, the estimated channel information may be not very accurate either because of the estimation error or because of the delay between the estimation and the transmission instances. It is therefore worth studying adaptive resource allocation schemes while considering channel mismatch.

The circuit of the proposed hardware architecture is based on equations (2.1) and (2.3), however, it is not general enough to hold for all practical systems. For example, the required power $f_k(c)$ in $\frac{f_k(c)}{\alpha_{k,n}^2}$ may correspond to certain *coding* and *modulation* schemes. Thus, we need to modify hardware circuit of the DPG method for other specific $f_k(c)$. Because changing hardware is not as easy as the software, it is therefore worth studying the easily implemented hardware architecture for different communication system to meet real-time application requirement.

References

[1] J. Blogh, P. Cherriman, and L. Hanzo, ”Comparative study of adaptive beam-steering and adaptive modulation-assisted dynamic channel allocation algorithms,” *IEEE Trans. Veh. Technol.*, vol. 50, no. 2, pp. 398-415, March 2001.

[2] L. Xu, X. Shen, and JW Mark, “Fair resource allocation with guaranteed statistical QoS for multimedia traffic in Wideband CDMA cellular network,” *IEEE Trans. Mobile Computing*, vol. 4, no. 2, pp. 166-177, 2005.

[3] C. Y. Wong, R. S. Cheng, K. B. Letaief, and R. D. Murch, “Multiuser OFDM with 33 adaptive subcarrier, bit, and power allocation,” *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1747–1758, Oct. 1999.

[4] W. Rhee and J. M. Cioffi, “Increase in capacity of multiuser OFDM system using dynamic subchannel allocation,” in *Proc. IEEE VTC*, Spring, Tokyo, Japan, May 2000, vol. 2, pp. 1085–1089.

[5] I. Kim, H. L. Lee, B. Kim, and Y. H. Lee, ”Use of linear programming for dynamic subcarrier and bit allocation in multiuser OFDM,” *IEEE Trans. Veh. Technol.*, vol. 55, no. 4, pp. 1195-1207, March 2006.

[6] M. Ergen, S. Coleri, and P. Varaiya, “QoS aware adaptive resource allocation techniques for fair scheduling in OFDMA based broadband wireless access systems,” *IEEE Trans. Broadcasting*, vol. 49, no. 4, pp. 362-370, Dec. 2003.

[7] D. Kivanc, G. Li, and H. Liu, “Computationally efficient bandwidth allocation and power control for OFDMA,” *IEEE Trans. Wirel. Commun.*, vol. 2, no. 6, pp. 1150–1158, Oct. 2003.

[8] G. Zhang, “Subcarrier and bit allocation for real-time services in multiuser OFDM systems,” in *Proc. IEEE International Conf. Communications*, vol. 5, pp. 2985 – 2989, June 2004.

[9] Z. Han, Z. Ji, and KJR Liu, “Low-complexity OFDMA channel allocation with Nash bargaining solution fairness,” in *Proc. IEEE Global Telecommunication Conf.*, vol. 6, pp. 3726-3731, Dec. 2004.

[10] Y. C. Ho, *Soft Optimization for Hard Problem*. Cambridge, MA: Harvard Univ., 1996.

34 Lecture Notes.

[11] Y. C. Ho, C. C. Cassandras, C. H. Chen, and L. Dai, “Ordinal optimization and simulation,” *J. Oper. Res. Soc.*, vol. 21, pp. 490–500, 2000.

[12] T. W. E. Lau and Y. C. Ho, “Universal alignment probability and subset selection for ordinal optimization,” *J. Optim. Theory Appl.*, vol. 39, no. 3, June 1997.

[13] D. Whitley, “A Genetic Algorithm Tutorial,” Colorado State Univ., Tech. Rep. CS-93-103, Mar. 1993.

[14] D. Beasley, D. R. Bull, and R. R. Martin, “An overview of genetic algorithms: Part 1, fundamentals,” *University Computing*, vol. 15, no. 2, pp. 58–69, 1993.

[15] D. Beasley, D. R. Bull, and R. R. Martin, “An overview of genetic algorithms: Part 2, research topics,” *University Computing*, vol. 15, no. 4, pp. 170–181, 1993.

[16] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming: An Introduction*. San Mateo, CA: Morgan Kaufmann, 1998.

[17] C. T. Lin and C. S. George Lee, *Neural Fuzzy System: A Neuro-Fuzzy Synergism to Intelligent Systems*, Englewood Cliffs, NJ: Prentic-Hall, 1996.

[18] Hagan, MT, and M. Menhaj, “Training feedforward networks with Marquardt algorithm,” *IEEE Trans. Neural networks*, vol. 5, no. 6, pp. 989-993, Nov. 1994.

[19] G. Lera and M. Pinzolas, “Neighborhood based Levenberg–Marquardt algorithm for neural network training,” *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1200–1203, Sep. 2002.

[20] S. K. Lai, R. S. Cheng, K. Ben Letaief, and R. D. Murch, “Adaptive trellis coded MQAM and power optimization for OFDM transmission,” in *Proc. IEEE Vehicular Technology Conf. (VTC’99)*, Houston, TX, May 1999.

[21] D. Luenberger, *Linear and Nonlinear Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1984.

[22] C. Lin and S. Lin, “A new dual-type method used in solving optimal power flowproblems,” *IEEE Trans. Power Syst.*, vol. 12, pp. 1667–1675, Nov. 1997.

[23] S. Lin and C. Lin, “A computationally efficient method for nonlinear multicommodity network flow problems,” *Networks*, pp. 225–244, July 1997.

[24] S. Y. Lin, Y. C. Ho and C. H. Lin, “An ordinal optimization theory based algorithm for solving the optimal power flow problem with discrete control variables,” *IEEE Trans. Power Systems*, vol. 19, no. 1, pp. 276-286, Feb. 2004.

[25] T. S. Rappaport, Wireless Communications: Principle and Practice, Prentice Hall, 2nd edition, 2002.

[26] S. K. Hsu, S. K. Mathew, M. A. Anders, B. R. Zeydel, V. G. Oklobdzija, R. K. Krishnamurthy, and S. Y. Borkar, “A 110 GOPS/W 16-bit multiplier and reconfigurable PLA loop in 90-nm CMOS,” *IEEE J. Solid-State Circuits*, Vol. 41, pp. 256 – 264, Jan. 2006.

[27] R. Kanan, B. Hochet, M. Declercq, and A. Guyot, “A Low-Power High Storage Capacity Structure for GaAs MESFET ROM,” in *Proc. International Workshop Memory Technology, Design and Testing*, San Jose, pp. 58-63, Aug. 1997.

[28] K. C. Chang, Digital Systems Design with VHDL and Synthesis: an Integrated Approach, *IEEE Computer. Society Press, Los Alamitos, California*, USA, 1999.

[29] J. Chuang and N. Sollenberger, “Beyond 3G: Wideband Wireless Data Access Based on OFDM and Dynamic Packet Assignment,” *IEEE Communications Magazine,* vol. 7, no. 38, pp. 78-87, July 2000.

[30] S. M. Sait and H. Youssef, Iterative Computer Algorithms With Applications in Engineering: Solving Combinatorial Optimization Problems. Los Alamitos, CA: *IEEE Comput. Soc.*, Aug. 1999.

[31] E. K. P. Chong and S. H. Żak, An Introduction to Optimization, 2nd ed. New York: Wiley, 2001

[32] L. G. Alberto, Probability and Random Processes for Electrical Engineering, Second Edition, Addison Wesley, 1994.

[33] S. Y. Lin and S. C. Horng, “Application of an ordinal optimization algorithm to the wafer testing process,” *IEEE Trans. Systems, Man, and Cybernetics*, Part A, Vol. 36, No. 6, pp. 1229-1234, Nov. 2006.

[34] S. C. Horng and S. Y. Lin, “Ordinal optimization of G/G/1/K polling systems with k-limited service discipline,” accepted by *Journal of Optimization Theory and*

Applications.

[35] L. Y. Ou and Y. F. Chen, ”An iterative multi-user bit and power allocation algorithm for DMT-based Systems”, *IEICE Trans. Commun*., vol. E88-B, no. 11, pp. 4259-4265, Nov. 2005.

[36] S. M. Lee and D. J. Park “Fast optimal bit and power allocation based on the Lagrangian method for OFDM systems,” *IEICE Trans. Commun*., vol. E89-B, no. 4, pp. 1346-1353, Apr. 2006.

List of Publication
著作目錄

姓名：黃榮壽 (Jung-Shou Huang**)**

期刊論文著作：

1. Shin-Yeu Lin and Jung-Shou Huang, “Adaptive Subcarrier Assignment and Bit Allocation for Multiuser OFDM System Using Ordinal Optimization Approach,” IEEE Transactions on Vehicular Technology, vol. 57, no. 5, pp. 2907-2919, Sep. 2008. (EI, SCI)
2. Shin-Yeu Lin and Jung-Shou Huang, “A Computationally Efficient Method for Large Dimension Subcarrier Assignment and Bit Allocation Problem of Multiuser OFDM System,” accepted to appear in IEICE Transactions on Communications. (EI, SCI)

研討會論文著作：

1.林心宇、黃榮壽，“一個用於多重使用者正交分頻多工系統中有效率的次載波指定與位元分配方法”，2007 年資訊科技應用學術研討會，新竹，八月三十一日，2007。

博士候選人學經歷資料

姓名：黃榮壽

性別：男

生日：中華民國 59 年 3 月 30 日

籍貫：台灣省台中縣

論文題目：中文：以序的最佳化理論為基礎的多重使用者正交分頻多工系統的適應性次載波指定與位元分配方法

英文：Adaptive Subcarrier Assignment and Bit Allocation Methods for Multiuser OFDM System Using Ordinal Optimization Approach

學歷：

1. 私立淡江大學電機工程學系學士，民國 83 年 9 月~85 年 6 月
2. 國立交通大學電機與控制工程學系碩士，民國 85 年 9 月~87 年 6 月
3. 國立交通大學電機與控制工程學系博士班，民國 91 年 9 月~97 年 9 月

經歷：

1. 泰威科技電子工程師，民國 87 年~88 年
2. 世紀民生設計工程師，民國 88 年~89 年
3. 義隆電子設計工程師，民國 89 年~97 年

Vita

Jung-Shou Huang was born in Taiwan, R.O.C.. He received the B.S. degree in electrical engineering from Tamkang University in 1996 and the M.S. degree in electrical and control engineering from National Chiao Tung University in 1998. He has been studying for his Ph. D. degree in electrical and control engineering from National Chiao Tung University since September 2002.

Currently, he also works as a digital IC designer in Elan Microelectronics Corporation. His major research interests include optimization theory with applications, image processing and digital IC design.