

國立交通大學

電信工程學系

碩士論文

多用戶無線多媒體通訊之節能式資源分配

Energy Efficient Radio Resource Allocation for  
Multiuser Multimedia Communications

研究生：邱泰翔

指導教授：蘇育德 博士

西元 2007 年 7 月

多用戶無線多媒體通訊之節能式資源分配  
Energy Efficient Radio Resource Allocation for  
Multiuser Multimedia Communications

研究生：邱泰翔

Student : Tai-Hsiang Chiu

指導教授：蘇育德 博士

Advisor : Dr. Yu T. Su

國立交通大學  
電信工程學系  
碩士論文

A Thesis

Submitted to Department of Communications Engineering  
College of Electrical and Computer Engineering  
National Chiao Tung University  
in Partial Fulfillment of the Requirements  
for the Degree of Master of Science

In

Communications Engineering

July 2007

Hsinchu, Taiwan, Republic of China

西元 2007 年 7 月

# 國立交通大學

## 博碩士論文全文電子檔著作權授權書

(提供授權人裝訂於紙本論文書名頁之次頁用)

本授權書所授權之學位論文，為本人於國立交通大學電信工程系所  
系統組組，95學年度第2學期取得碩士學位之論文。

論文題目：多用戶無線多媒體通訊之節能式資源分配

指導教授：蘇育德 教授

同意  不同意

本人茲將本著作，以非專屬、無償授權國立交通大學與台灣聯合大學系統圖書館：基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學及台灣聯合大學系統圖書館得不限地域、時間與次數，以紙本、光碟或數位化等各種方法收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行線上檢索、閱覽、下載或列印。

論文全文上載網路公開之範圍及時間：

本校及台灣聯合大學系統區域網路	<input checked="" type="checkbox"/> 立即公開
校外網際網路	<input checked="" type="checkbox"/> 立即公開

授權人：邱泰翔

親筆簽名：邱泰翔

中華民國 96 年 8 月 27 日

# 國立交通大學

## 博碩士紙本論文著作權授權書

(提供授權人裝訂於全文電子檔授權書之次頁用)

本授權書所授權之學位論文，為本人於國立交通大學電信工程系所  
系統組組，95學年度第2學期取得碩士學位之論文。

論文題目：多用戶無線多媒體通訊之節能式資源分配

指導教授：蘇育德 教授

### ■ 同意

本人茲將本著作，以非專屬、無償授權國立交通大學，基於推動讀者間「資源共享、互惠合作」之理念，與回饋社會與學術研究之目的，國立交通大學圖書館得以紙本收錄、重製與利用；於著作權法合理使用範圍內，讀者得進行閱覽或列印。

本論文為本人向經濟部智慧局申請專利(未申請者本條款請不予理會)的附件之一，申請文號為：\_\_\_\_\_，請將論文延至\_\_\_\_年\_\_\_\_月\_\_\_\_日再公開。

授權人：邱泰翔

親筆簽名： 邱泰翔

中華民國 96 年 8 月 27 日

# 國家圖書館博碩士論文電子檔案上網授權書

ID:GT009413505

本授權書所授權之論文為授權人在國立交通大學 電機 學院 電信工程 系所 系統組 組，95 學年度第 2 學期取得碩士學位之論文。

論文題目：多用戶無線多媒體通訊之節能式資源分配

指導教授：蘇育德 教授

茲同意將授權人擁有著作權之上列論文全文（含摘要），非專屬、無償授權國家圖書館，不限地域、時間與次數，以微縮、光碟或其他各種數位化方式將上列論文重製，並得將數位化之上列論文及論文電子檔以上載網路方式，提供讀者基於個人非營利性質之線上檢索、閱覽、下載或列印。

※ 讀者基於非營利性質之線上檢索、閱覽、下載或列印上列論文，應依著作權法相關規定辦理。

授權人：邱泰翔

親筆簽名：邱泰翔

民國 96 年 8 月 27 日

1. 本授權書請以黑筆撰寫，並列印二份，其中一份影印裝訂於附錄三之二(博碩士紙本論文著作權授權書)之次頁；另一份於辦理離校時繳交給系所助理，由圖書館彙總寄交國家圖書館。

國立交通大學  
電信工程學系碩士班

論文口試委員會審定書

本校 電信工程學系 碩士班 邱泰翔 君

所提論文(中文) 多用戶無線多媒體通訊之節能式資源分配

(英文) Energy Efficient Radio Resource Allocation for  
Multiuser Multimedia Communications

合於碩士資格水準、業經本委員會評審認可。

口試委員：

王世明

郭建

蘇賜麟

王薏君

指導教授：

李彥志

系主任：

李大嵩 教授

中華民國 96 年 9 月 25 日

# 多用戶無線多媒體通訊之節能式資源分配

研究生：邱泰翔

指導教授：蘇育德 博士

國立交通大學電信工程學系碩士班

## 中文摘要

本文旨在探討如何分配現有的無線電資源(傳輸功率、能量及通道數目)來滿足不同用戶之不同傳輸率之要求。我們考慮各通道之增益雜訊比(channel gain-to-noise ratio)之不同，推導出其功率、傳輸速率之最佳分配。這種分配可以最少功率或能量滿足各種多媒體不同傳輸速率的需求。我們也探討在固定總輸出能量或功率的限制下如何讓總傳輸率和(sum rate)最大。

在單一傳輸率要求的環境下，我們推導出最佳的公式解，根據這個公式我們提出兩種簡單的遞迴求解法來解決多種傳輸率的問題，第一種解法可得到近似最佳解，另一種則可保證得到最佳解。

在多傳輸率的情況，先前多媒體通訊資源分配的方法通常需要大量運算時間因此使得其在手持裝置上求最佳分配解變得不可行。我們的方法卻只要透過幾次迴圈就可以決定通道分配的矩陣，這個解法同時可用於單一和多重傳輸速率的多媒體傳輸應用上。除了運算簡單之外，更重要的是，這個演算法之性能和最佳解只有微小的誤差，使得此演算法深具吸引力。第二種解法稍為複雜，但可以保證找到最佳解。這兩種方法其複雜度皆遠低於現有求最佳解的方法，因此實用性極高。

我們並舉兩個應用的例子，第一個例子考慮了類似 IEEE 802.16e 的 OFDMA 下傳鏈，第二個應用例子則是區域合作式的上傳鏈。

# Energy Efficient Radio Resource Allocation for Wireless Multiuser Multimedia Communications

Student : Tai-Hsiang Chiu      Advisor : Yu T. Su

Department of Communications Engineering  
National Chiao Tung University

## Abstract

This thesis is concerned with a dynamic radio resource allocation problem. Given the availability of multiple orthogonal channels and transmission rate requirements from various wireless network users, we are interested in a joint channel, power and rate assignment scheme that satisfies the multimedia multi-rate requirements with the minimum total power.

For the mono-rate cases, we present a simple approach that gives a closed-form expression for the optimal solution. Based on this closed-form solution, we present two iterative algorithms that enable a transmitter to determine respectively a near-optimal and the optimal joint assignment scheme using the channel state information.

Earlier resource allocation schemes often require computational intensive and time-consuming procedures to find the optimum solution, if exists, or suboptimal ones that makes them not very practical for mobile devices to implement. In contrast, our proposals are very efficient in terms of computing load and convergence rate. Furthermore, our approaches can also be used to solve a dual problem: that of maximizing the sum rate while satisfying the total power or energy constraint.



## 誌 謝

對於論文能夠順利完成首先我要感謝指導教授 蘇育德 博士。老師的諄諄教誨讓我在這兩年的研究生生活中，無論是電信領域的專業亦或是生活上的待人處世，皆令我有很大收穫。也要感謝蒞臨的口試委員，他們提供的意見和補充資料使本文得以更加完整。此外，我尤其要感謝 林淵斌 學長不時地提供寶貴意見和幫助，還有實驗室其他的學長姐以及同學，讓我不僅在學習的過程中獲益匪淺，同時也為這兩年的生活增添了許多歡樂。

最後，我更要感謝我的家人，他們的關心與鼓勵讓我能沒有後顧之憂的專心於學業，沒有他們在背後的支持，我沒有辦法這麼順利的完成論文，謹以此論文獻上我最深的敬意！



# Contents

English Abstract	i
Contents	ii
List of Figures	iv
List of Tables	vi
<b>1 Introduction</b>	<b>1</b>
<b>2 Scenario and Problem Formulation</b>	<b>4</b>
2.1 Downlink Scenario and Assumptions . . . . .	4
2.2 Uplink Scenario and Assumptions . . . . .	5
2.3 Problem Formulation . . . . .	7
<b>3 General Tree Searching</b>	<b>10</b>
3.1 Algorithm Based on Dynamic Programming . . . . .	10
3.1.1 The basic problem . . . . .	10
3.1.2 The dynamic programming algorithm . . . . .	12
3.2 Algorithm Based on Branch and Bound . . . . .	13
3.2.1 Terminology and general description . . . . .	14
3.2.2 Bounding function . . . . .	15
3.2.3 Branching rule . . . . .	16
3.2.4 Strategy for selecting next subproblem . . . . .	17

<b>4</b>	<b>Mono-Rate Wireless Resource Allocation</b>	<b>20</b>
4.1	Conventional Greedy Algorithm . . . . .	20
4.2	An Iterative Optimization Algorithm . . . . .	22
4.2.1	Computational complexity . . . . .	29
<b>5</b>	<b>Multi-Rate Wireless Resource Allocation</b>	<b>31</b>
5.1	A Greedy Algorithm . . . . .	31
5.2	Dynamic Programming Based Resource Allocation Algorithm . . . . .	32
5.2.1	Dynamic programming formulation . . . . .	32
5.2.2	A dynamic programming approach . . . . .	34
5.2.3	Numerical behavior . . . . .	35
5.3	A Branch and Bound Approach for Resource Allocation . . . . .	41
5.3.1	Bounding function . . . . .	41
5.3.2	Branching rule . . . . .	41
5.3.3	Strategy for selecting next subproblem . . . . .	42
5.3.4	Complexity reduction techniques for the B&B based algorithm . . . . .	43
5.4	Complexity Consideration . . . . .	56
<b>6</b>	<b>Simulation Results</b>	<b>60</b>
6.1	Resource Allocation for an OFDMA Downlink System . . . . .	61
6.2	Resource Allocation for a Locally Cooperative Uplink System . . . . .	64
<b>7</b>	<b>Conclusion</b>	<b>67</b>
	<b>Bibliography</b>	<b>68</b>

# List of Figures

3.1	Illustration of the search space of B&B. . . . .	15
3.2	The relation between the bounding function $g$ and the objective function $f$ on the sets $S$ and $G$ of feasible and potential solutions of a problem. . . . .	16
3.3	Search strategies in B&B: the Best First Search. . . . .	18
3.4	Search strategies in B&B: the Breath First Search. . . . .	19
3.5	Search strategies in B&B: the Depth First Search. . . . .	19
4.1	The iterative optimization algorithm for fixed mono-data rate . . . . .	27
4.2	Numerical results obtained in the course of applying the algorithm of Fig. 4.1 to solve Example 4.1. . . . .	28
4.3	Average complexities of the proposed iterative power allocation algorithm for mono-rate transmission with different total transmit power constraints . . . . .	30
5.1	The problem formulation modified to DP format . . . . .	34
5.2	Probabilities of <i>DPRA</i> algorithm achieving the optimum solution in a downlink scenario. . . . .	37
5.3	Average performance loss (with respect to the optimum solution) of <i>DPRA</i> algorithm in a downlink scenario. . . . .	37
5.4	Probabilities of <i>DPRA</i> algorithm achieving the optimum solution in an uplink scenario. . . . .	38
5.5	Average performance loss (with respect to the optimum solution) of <i>DPRA</i> algorithm in an uplink scenario. . . . .	38

5.6	Trellis associated with the <i>DPRA</i> algorithm and the corresponding numerical values in solving Example 5.1. . . . .	39
5.7	The B&B format complied searching tree for multiuser channel assignment.	44
5.8	Illustration for subsets of all child nodes from the initial node . . . . .	49
5.9	The searching tree associated with Example 5.2. . . . .	54
5.10	Average complexities of <i>BBRA</i> and <i>DPRA</i> schemes in a 64-VOCs downlink scenario. . . . .	58
5.11	Average complexities of <i>BBRA</i> and <i>DPRA</i> schemes in a 128-VOCs downlink scenario. . . . .	58
5.12	Average complexities of <i>BBRA</i> and <i>DPRA</i> schemes in a 10-VOCs uplink scenario. . . . .	59
5.13	Average complexities of <i>BBRA</i> and <i>DPRA</i> schemes in a 15-VOCs uplink scenario. . . . .	59
6.1	The probability that the <i>DPRA</i> method yields the optimum performance in an OFDMA downlink system. . . . .	62
6.2	The average performance degradation (with respect to the optimum performance) of the <i>DPRA</i> algorithm in an OFDMA downlink system. . . .	63
6.3	The average complexity of the <i>BBRA</i> and <i>DPRA</i> schemes when used in an OFDMA downlink system. . . . .	63
6.4	The probability of the <i>DPRA</i> method achieving the optimum in a locally cooperative uplink scenario . . . . .	65
6.5	The average performance loss from the optimum by the <i>DPRA</i> algorithm in a locally cooperative uplink scenario . . . . .	66
6.6	The average complexity of the <i>BBRA</i> and <i>DPRA</i> schemes in a locally cooperative uplink scenario . . . . .	66

# List of Tables

5.1	Numerical results obtained in solving Example 5.1. . . . .	40
5.2	The procedure of allocating the order of tree's level. . . . .	47
5.3	The procedure of reusing the results of repetitious calculations. . . . .	50
5.4	Effects of different techniques on the performance of the <i>BBRA</i> approach. . . . .	52
5.5	A greedy search procedure for solving Example 5.2 . . . . .	53
5.6	The step-by-step searching procedure for solving Example 5.2. . . . .	55
6.1	Transmission rate requirements for video, audio, voice and data services . . . . .	60
6.2	Statistical characterizations of two independent multimedia sources. . . . .	60
6.3	Simulation parameters of the OFDMA system . . . . .	61
6.4	System parameters of the locally cooperative uplink system under consideration. . . . .	64

# Chapter 1

## Introduction

As the demand for higher data rate multi-media wireless communications increases, it also becomes more and more important that one takes into account the energy-efficiency factor in designing an anti-fading transmission scheme for mobile terminals. Resource allocation such as dynamic power control has long been regarded as an effective means to reduce the average power consumption, co-channel interferences and maintain the link quality in a wireless network. In some applications such as wireless sensor networks, in which terminals are powered by batteries, efficient power management is essential in meeting the network's life expectancy.

By using an optimal dynamic power and rate allocation strategy, Goldsmith *et al.* [1] derived the ergodic (Shannon) capacity of a single-user wideband fading channel when channel state information (CSI) is available at both the transmit and the receive sides. The ergodic capacity is the maximum long-term achievable rate averaged over all states of a time-varying channel. The corresponding optimal power allocation strategy is obtained via a water-filling procedure over time or, equivalently, over the fading states. For a fading multiple-access channel (MAC) with perfect CSI at both sides of the MA link, Knopp and Humblet [2] derived the optimal power control policy that maximizes the total ergodic rates of all users.

This thesis is concerned with efficient channel, power and rate assignment schemes that fulfill either a mono-rate or multi-rate requirements with minimum total transmitted

power. We generalize the conventional problem setting by defining virtual orthogonal channels (VOCs) and present very efficient iterative algorithms that is guaranteed to yield the optimal solution for mono-rate transmissions. For multiple rate applications, we suggest two efficient procedures that give optimal and near-optimal solutions of the problem.

The proposed iterative algorithms are then applied to two operation scenarios. The first scenario is a single cell downlink Orthogonal Frequency Division Multiple Access (OFDMA) system in which the base station needs to meet multiple rate requirements simultaneously. The second application example we considered is an MA uplink with locally cooperative communication links. This scenario occurs when dual-mode (e.g., WLAN/WCDMA or WLAN/GPRS) user terminals located within a small neighborhood form an opportunistic local network so that active users can forward their messages to neighboring idle terminals for cooperative transmissions. It is reasonable to assume that the inter-user distance within the local network is far smaller than the terminal-to-base-station distance, hence an active mobile terminal uses the local transmission (e.g., WLAN) mode for inter-user communication, which is almost error-free and consumes negligible power, and the cellular (e.g., WCDMA or GPRS) mode for uplink transmissions.

Note that such a cooperative communication scheme is a special case of the opportunistic diversity first introduced by Tse and Hanly [3] and [4]. Many later investigations focus on the transmission and protocol issues related to cooperative space diversity and relays. Sendonaris *et al.* [5] [6] proposed a Code Division Multiple Access (CDMA)-based two-user cooperative communication scheme that allows two users to act as relays to retransmit the estimated data of their partner's information at the highest possible uplink transmission rate. This simple relay technique was extended by [7] and [8] to a variety of cooperative strategies. Maric and Yates [9] considered a cooperative broadcast strategy with an objective to maximize the network lifetime.



Most of previous works on cooperative communications focus on the improvement of the peer-to-peer link quality and present some scheduling schemes without considering the implementation complexity. Zhu *et al.* [10] proposed a method to solve the problems about who should help whom and how to cooperate over a multiuser Orthogonal Frequency Division Multiplexing (OFDM) network. Although their algorithm leads to optimal performance, the associated computing complexity is relatively high for a mobile device. Water-filling-like algorithms for optimal power allocation with prescribed error tolerance for various systems have been proposed [10]-[13]. These algorithms are basically greedy (exhaustive) searches and there is no guarantee that the optimal solution is to be found. By contrast, our proposals are much simpler and the optimal allocation is obtainable.

The rest of this thesis is organized as follows. The ensuing chapter describes potential operation scenarios for radio resource allocation needs and gives an abstract problem statement. In Chapter 3, we review the general problem of tree searches and discuss possible solutions based on dynamic programming and branch-and-bound algorithms. Chapter 4 presents an efficient iterative algorithm that is guaranteed to yield the optimal solution for mono-rate transmissions. In the following chapter, we propose two efficient algorithms to find suboptimal and the optimal channel assignment matrices for multiple rate applications. Finally, we provide numerical performance of these algorithms when they are used in the two application scenarios discussed in Chapter 2.

# Chapter 2

## Scenario and Problem Formulation

### 2.1 Downlink Scenario and Assumptions

A base station in a cellular system needs to transmit multiple data streams to mobile stations simultaneously. To eliminate or suppress mutual interference, these data streams are often transmitted through channels that are orthogonal in time, frequency, code or hybrid domains. For example, the IEEE 802.16e adapts an OFDMA scheme so that a user data stream can be allocated in a fixed number of orthogonal subcarriers and time slots. Multiple orthogonal channels are also available in the 3G-Cdma2000 system via Walsh-Hadamard coding. If multiple antennas are installed at one side or both sides of the link and if the channel state is known to both sides, the resulting multiple-input multiple-output (MIMO) channel can be decomposed into parallel orthogonal channels. Hence, without loss of generality, we shall consider a general scenario under which  $N$  virtual orthogonal channels (VOCs) are available for multirate downlink transmission and the instantaneous channel conditions of all VOCs are available to the transmitter(s). Therefore, the number of VOCs includes the number of the orthogonal subcarriers [10], time-slots which are unused or any other forms of orthogonal channels like the number of eigen-channels in a MIMO wireless link [15].

It is assumed that each VOC is independently and frequency non-selectively Rayleigh-

faded and the bandwidth of the  $i$ th channel is  $W_i$  during a fixed transmission interval. The candidate data types might include voice, image, video, data, etc., each has a distinct rate requirement, and the number of the requested data types is given by

$$d = \sum_{j=1}^{K_d} n_j \quad (2.1)$$

where  $K_d$  is the number of active destination terminals and  $n_j$  is the number of the  $j$ th user's request data types. For convenience, the same data type, like voices, from different users will be regarded as different types.

The maximum transmission rate (capacity)  $C_{ij}$  offered by the  $i$ th channel to serve the  $j$ th data type with transmitted power  $p_{ij}$  is

$$C_{ij} = W_i \log_2 \left( 1 + \frac{p_{ij} h_{ij}}{\sigma_{ij}^2} \right), \quad 1 \leq i \leq N, \quad 1 \leq j \leq d \quad (2.2)$$

where  $h_{ij}$  and  $\sigma_{ij}^2$  denote the channel gain and noise power of the  $i$ th VOC which serves the  $j$ th data type. For simplicity, it is assumed that the noise power ( $\sigma_{ij}^2$ ) and bandwidths ( $W_i$ ) are the same for all VOCs and are given by ( $\sigma^2, W$ ). Further, each channel's gain-to-noise ratio (GNR) ( $h_{ij}/\sigma^2$ ) is known to the base station and the normalized capacity (rate)  $r_{ij}$  for the  $i$ th channel to serve the  $j$ th data type is

$$r_{ij} = \frac{C_{ij}}{W \log_2 e} = \ln \left( 1 + \frac{h_{ij} p_{ij}}{\sigma^2} \right) = \ln (1 + a_{ij} p_{ij}), \quad (2.3)$$

where  $a_{ij} = h_{ij}/\sigma^2$ .

## 2.2 Uplink Scenario and Assumptions

Another scenario we are interested in is a locally cooperative uplink system in which a group of neighboring dual-mode mobile terminals forms a local area network. We assume that the mobile terminals are located within a small neighborhood so that the inter-user distance is far smaller than the uplink (terminal-to-base-station) distance. The users who are willing to share their resources form a cooperative communication

network. When a user terminal decides to solicit for transmission aid, it broadcasts its requirement to its peers in the network. Upon receiving the request for cooperation, other network users will send their replies to inform the requester(s) of the resources to be offered and their conditions (gain-to-noise ratios). For such a locally cooperative network, the available uplink channels are converted into VOCs which might include various multiple access channels such as FDMA, TDMA, CDMA or SDMA system and other equivalent orthogonal or near-orthogonal channels for accessing the base station of interest.

As in the downlink case, we assume that each VOC is independently Rayleigh-faded and the  $i$ th VOC has bandwidth  $W_i$  Hz during a fixed transmission interval. Consequently, the number of VOCs is given by

$$N = \sum_{i=1}^{K_u} m_i,$$

where  $m_i$  denotes the number of VOCs offered by the  $i$ th user and  $K_u$  represents the number of active users in the cooperative network. The number of the transmission rates (data types)  $d$  is simply

$$d = \sum_{j=1}^{\hat{K}_u} n_j,$$

where  $n_j$  is the number of data types requested by the  $j$ th user and  $\hat{K}_u$  is the number of users (help-seekers) who have announced their transmission cooperation requests. Obviously,  $\hat{K}_u \leq K_u$ . It is assumed that the mobiles who join the locally cooperative communication network can perfectly obtain the related rates and data types information through local area network (LAN) communications, e.g., IEEE 802.11.

Note that for the downlink case, each physical channel has different GNR values when assigned to transmitting to different receive terminals. By contrast, for the uplink transmission example considered here, each data type is carried by the source terminal itself or by a mobile station who offers unused VOCs, a channel's GNR remains unchanged no matter whose message is transmitted. Thus the maximum transmission rate

$r_{ij}$  supplied by the  $i$ th VOC with transmitted power  $p_{ij}$  is

$$r_{ij} = \frac{C_{ij}}{W \log_2 e} = \ln \left( 1 + \frac{h_i p_{ij}}{\sigma^2} \right) = \ln (1 + a_i p_{ij}), \quad a_i = \frac{h_i}{\sigma^2}. \quad (2.4)$$

## 2.3 Problem Formulation

Given the multiple rate requirements and channel state information (i.e.,  $a_i$ 's), one would like to find the channel assignment and power allocation that minimize the total transmitted power. We first consider the downlink case and define the  $N \times d$  channel assignment matrix  $\mathbf{A}_{N \times d} = [\mathbf{A}_{ij}]$  by  $A_{ij} = 1$  if the  $i$ th VOC is used to transmit the  $j$ th data type; otherwise,  $A_{ij} = 0$ . Assuming a VOC can only serve one data type at a given time interval, then  $A_{ij}$  is either 1 or 0 and a legitimate channel assignment matrix  $\mathbf{A}_{N \times d}$  must satisfy

$$\sum_{j=1}^d A_{ij} \leq 1, \quad \sum_{i=1}^N A_{ij} \geq 1, \quad 1 \leq i \leq N, \quad 1 \leq j \leq d \quad (2.5)$$

For the downlink case, all signals are transmitted from the same base station, hence only the total transmitter power will be considered.

Mathematically, the problem of concern is [10]

$$\begin{aligned} & \min_{\mathbf{P}, \mathbf{A}} \sum_{i=1}^N \sum_{j=1}^d p_{ij} \\ \text{s.t. } & \sum_{i=1}^N A_{ij} r_{ij} = R_j, \quad \sum_{i=1}^N \sum_{j=1}^d p_{ij} \leq P_c, \quad 1 \leq j \leq d, \quad 1 \leq i \leq N \end{aligned} \quad (2.6)$$

where  $\mathbf{A}$  is the channel assignment matrix defined above and  $\mathbf{P}$  denotes the power allocation matrix whose  $(i, j)$ th entry  $p_{ij}$  represents the  $i$ th VOC's transmitted power for sending the  $j$ th data type and  $r_{ij}$  is the corresponding transmission rate.  $R_j$  is the required rate of the  $j$ th data type and  $P_c$  is the total transmitter power constraint.

Although in reality there is a peak total power constraint  $\sum_{i=1}^N \sum_{j=1}^d p_{ij} \leq P_c$ , we shall not consider this constraint to begin with. Solving the problem with the total power constraint follows a two-step procedure. In the first step we solve the unconstrained problem to obtain the required optimal total power and then check if the solution meets

the peak power constraint. The problem is solved if the constraint is satisfied; otherwise the problem does not have an admissible solution and one is forced to go to Step 2. In the second step we can modify (decrease) the rate requirements, deny some data types, or settle with a suboptimal channel/power allocation to accommodate the peak total power constraint. Which of these three options is chosen depends on other system design considerations and the final solution is likely to be obtained by an *outer* iterative process. As far as this thesis is concerned, however, the total transmit power constraint shall not be discussed henceforth.

Recall that for the uplink scenario, the GNR associated with a VOC is independent of the channel assignment. Furthermore, as each VOC may belong to different users in the LAN, it will have its own peak power constraint, and the problem of the locally cooperative uplink scenario becomes

$$\begin{aligned}
 & \min_{\mathbf{P}, \mathbf{A}} \sum_{i=1}^N \sum_{j=1}^d p_{ij} \\
 \text{s.t. } & \sum_{i=1}^N A_{ij} r_{ij} = R_j, \quad 0 \leq p_{ij} \leq \bar{p}_i, \quad 1 \leq j \leq d, \quad 1 \leq i \leq N
 \end{aligned} \tag{2.7}$$

where  $\mathbf{P}$ ,  $\mathbf{A}$ ,  $p_{ij}$ ,  $r_{ij}$  and  $R_j$  are the same as those of (2.6) and  $\bar{p}_i$  is the power constraint of the  $i$ th VOC. This constraint can be transformed into the rate constraint  $\bar{r}_i = \ln(1 + a_i \bar{p}_i)$ . In other words, the constraint  $0 \leq p_{ij} \leq \bar{p}_i$  is equivalent to  $0 \leq r_{ij} \leq \bar{r}_i$ .

By removing the constraint  $\sum_{i=1}^N \sum_{j=1}^d p_{ij} \leq P_c$ , the two problems (2.6) and (2.7) can be unified as

$$\begin{aligned}
 & \min_{\mathbf{P}, \mathbf{A}} \sum_{i=1}^N \sum_{j=1}^d p_{ij} \\
 \text{s.t. } & \sum_{i=1}^N A_{ij} r_{ij} = R_j, \quad 0 \leq r_{ij} \leq \bar{r}_{ij}, \quad 1 \leq j \leq d, \quad 1 \leq i \leq N
 \end{aligned} \tag{2.8}$$

where  $\bar{r}_{ij} = \ln(1 + a_{ij} \bar{p}_i)$ .

It will become clear after our discourse in the following chapters that the dual problem

of maximizing the data throughput subject to a total power constraint

$$\begin{aligned} & \max_{\mathbf{P}, \mathbf{A}} \sum_{i=1}^N \sum_{j=1}^d r_{ij} \\ \text{s.t. } & \sum_{i=1}^N \sum_{j=1}^d A_{ij} p_{ij} = P_c, \quad 0 \leq p_{ij} \leq \bar{p}_i, \quad 1 \leq j \leq d, \quad 1 \leq i \leq N \end{aligned} \quad (2.9)$$

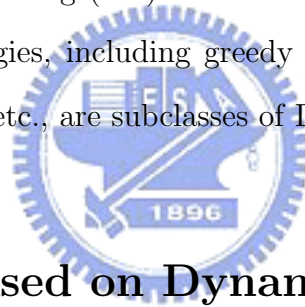
can be solved by the algorithms proposed in ensuing two chapters. As a result, for brevity, we shall not deal with (2.9) in this thesis.



# Chapter 3

## General Tree Searching

Tree representations are used in statistical inference, database structures and many other scientific applications. We are more interested in the special scenario that relates the tree searching to an optimal problem. This chapter introduces two techniques we used, namely dynamic programming (DP) and branch-and-bound. In a more general sense, all tree searching strategies, including greedy methods, the divide-and-conquer strategy, prune-and-search, ... etc., are subclasses of DP algorithms for they all involve sequences of decisions.



### 3.1 Algorithm Based on Dynamic Programming

The *dynamic programming* was coined by Bellman [14] to describe the techniques which he brought together to study a class optimization problems involving sequences of decisions. There have been many applications and further developments since that time. In this paper, we focus on the situations where decisions are made in stages.

#### 3.1.1 The basic problem

We now formulate a general problem of decision under stochastic uncertainty over a finite number of stages. This problem, which we call *basic*. The basic problem is very general. In particular, we will not require that the state, control, or random parameter take a finite number of values or belong to a space of  $n$ -dimensional vectors. A surprising



aspect of dynamic programming is that its applicability depends very little on the nature of the state, control, and random parameter spaces. For this reason, it is convenient to proceed without any assumptions on the structure of these spaces.

We are given a discrete-time dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k), \quad k = 0, 1, \dots, N - 1 \quad (3.1)$$

where

$k$  = the discrete time index,

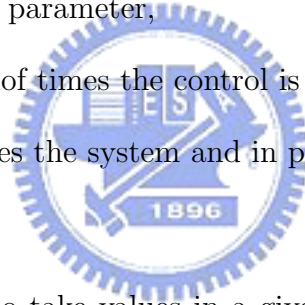
$x_k$  = the state of the system and summarized past information that is relevant for future optimization,  $x_k \in S_k$ ,

$u_k$  = the control or decision variable to be selected at time  $k$ ,  $u_k \in C_k$ ,

$w_k$  = a random perturbation parameter,

$N$  = the horizon or number of times the control is applied,

$f_k$  = a function that describes the system and in particular the mechanism by which the state is updated.



The control  $u_k$  is constrained to take values in a given nonempty subset  $U(x_k) \subset C_k$ , which depends on the current state  $x_k$ ; that is,  $u_k \in U_k(x_k)$  for all  $x_k \in S_k$  and  $k$ .

We consider the class of policies (also called control laws) that consist of a sequence of functions

$$\pi = \{\mu_0, \dots, \mu_{N-1}\} \quad (3.2)$$

where  $\mu_k$  maps states  $x_k$  into controls  $\mu_k = \mu_k(x_k)$  and is such that  $\mu_k(x_k) \in U_k(x_k)$  for all  $x_k \in S_k$ . Such policies will be called *admissible*.

Given an initial state  $x_0$  and an admissible policy  $\pi = \{\mu_0, \dots, \mu_{N-1}\}$ , the states  $\{x_k\}$  and disturbances  $\{w_k\}$  are random variables with distributions defined through the system equation

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k), \quad k = 0, 1, \dots, N - 1 \quad (3.3)$$

Thus, for given functions  $g_k$ ,  $k = 0, 1, \dots, N$ , the expected cost of  $\pi$  starting at  $x_0$  is

$$J_\pi(x_0) = E \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\} \quad (3.4)$$

where the expectation is taken over the random variables  $w_k$  and  $x_k$ . An optimal policy  $\pi^*$  is one that minimizes the cost; that is,

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0) \quad (3.5)$$

where  $\Pi$  is the set of all admissible policies.

### 3.1.2 The dynamic programming algorithm

The dynamic programming technique rests on a very simple idea, the *principle of optimality*, which we described below.

#### Principle of Optimality

Let  $\pi^* = \{\mu_0^*, \mu_1^*, \dots, \mu_{N-1}^*\}$  be an optimal policy for the basic problem, and assume that when using  $\pi^*$ , a given state  $x_i$  occurs at time  $i$  with positive probability. Consider the subproblem whereby we are at  $x_i$  at time  $i$  and wish to minimize the “cost-to-go” from time  $i$  to time  $N$

$$E \left\{ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

Then the truncated policy  $\{\mu_i^*, \mu_{i+1}^*, \dots, \mu_{N-1}^*\}$  is optimal for this subproblem. ■

The principle of optimality suggests that an optimal policy can be constructed in piecemeal fashion, first constructing an optimal policy for the “tail problem” involving the last stage, then extending the optimal policy to the “tail problem” involving the last two stages, and continuing in this manner until an optimal policy for entire problem is constructed. The dynamic programming algorithm is based on this idea: it proceeds sequentially, by solving all the tail subproblems of a given time length.

We now state the dynamic programming algorithm for the basic problem.

## The Dynamic Programming Algorithm

For every initial state  $x_0$ , the optimal cost  $J_\pi^*(x_0)$  of the basic problem is equal to  $J_0(x_0)$ , given by the first step of the following algorithm, which proceeds forward in time from period 1 to period  $N$ :

$$J_0(x_0) = g_0(x_0),$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k), w_k} E\{g_k(x_k, u_k, w_k) + J_{k-1}(f_x(x_k, u_k, w_k))\}, \quad k = 1, \dots, N \quad (3.6)$$

where the expectation is taken with respect to the probability distribution of  $w_k$ , which depends on  $x_k$  and  $u_k$ . Furthermore, if  $u_k^* = \mu_k^*(x_k)$  minimizes the right side of (3.6) for each  $x_k$  and  $k$ , the policy  $\pi^* = \{\mu_0^*, \mu_1^*, \dots, \mu_{N-1}^*\}$  is optimal. ■

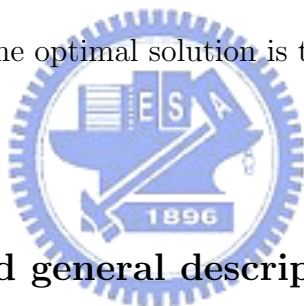
## 3.2 Algorithm Based on Branch and Bound

Solving NP-hard discrete and combinatorial optimization problems is often an immense job requiring very efficient algorithms, and the Branch and Bound (B&B) paradigm, first proposed by A. H. Land and A. G. Doig in 1960 for linear programming, is one of the main tools in the construction of these. The B&B method is basically an enumeration approach; it searches for the best solution in the complete space of solutions associated a given problem. However, explicit enumeration is normally impossible due to the exponentially increasing number of potential solutions. The use of bounds for the function to be optimized combined with the value of the current best solution enables the algorithm to prune the non-promising search space and search parts of the solution space only.

At any point during the course of search for the solution, the status of the solution with respect to the search of the solution space is described by a pool of yet unexplored subset of this and the best solution found so far. Initially only one subset exists, namely the complete solution space, and the best solution found so far is  $\infty$ . The unexplored subspaces are represented as nodes in a dynamically generated search tree, which initially

only contains the root, and each iteration of a classical B&B algorithm processes one such node. The iteration has three main components: selection of the node to process, bound calculation, and branching. In Fig. 3.1, the initial situation and the first step of the process are illustrated.

The sequence of these may vary according to the strategy chosen for selecting the next node to process. If the selection of next subproblem is based on the bound value of the subproblems, then the first operation of an iteration after choosing the node is branching. For each of these, it is checked whether the subspace consists of a single solution, in which case it is compared to the current best solution keeping the best of these. Otherwise the bounding function for the subspace is calculated and compared to the current best solution. If the subspace cannot contain the optimal solution, the whole subspace is discarded. The search terminates when there are no unexplored parts of the solution space left, and the optimal solution is then the one recorded as "current best"



### 3.2.1 Terminology and general description

In the following subsection, we consider minimization problems - the case of maximization problems can be dealt with similarly. The problem is to minimize a function  $f(x)$  of variables  $(x_1 \dots x_n)$  over a region of *feasible solutions*,  $S$ :

$$\min_{x \in S} f(x)$$

The function  $f$  is called the *objective function* and may be of any type. The set of feasible solutions is usually determined by general conditions on the variables, e.g. that these must be non-negative integers or binary, and special constraints determining the structure of the feasible set. In many cases, a set of *potential solutions*,  $G$ , containing  $S$ , for which  $f$  is still well defined. A function  $g(x)$  often defined on  $G$  (or  $S$ ) with the property that  $g(x) \leq f(x)$  for all  $x$  in  $S$  arises naturally. Both  $S$  and  $G$  are very useful

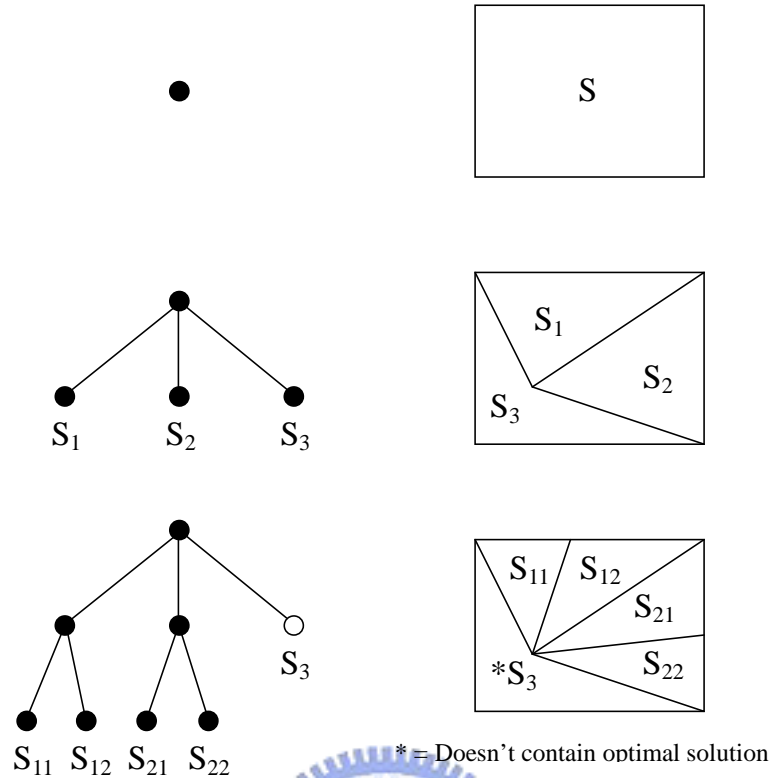


Figure 3.1: Illustration of the search space of B&B.

in the B&B context. Fig. 3.2 illustrates the situation where  $S$  and  $G$  are intervals of real numbers.

### 3.2.2 Bounding function

The bounding function is the key component of any B&B algorithm in the sense that a low quality bounding function cannot be compensated for through good choices of branching and selection strategies. Ideally the value of a bounding function for a given subproblem should equal the value of the best feasible solution to the problem, but on account of obtaining this value is usually in itself NP-hard, the goal is to come as close as possible using only a limited amount of computational effort. A bounding function is called *strong*, if it in general gives values close to the optimal value for the subproblem bounded, and *weak* if the values produced are far from the optimum. One often experiences a trade off between quality and time when dealing with bounding

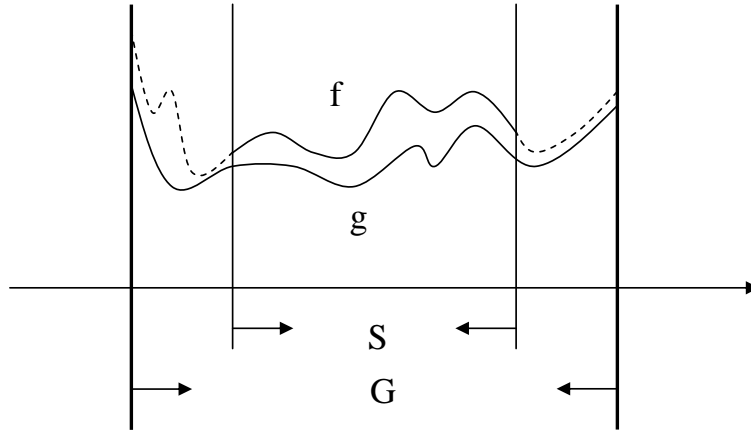


Figure 3.2: The relation between the bounding function  $g$  and the objective function  $f$  on the sets  $S$  and  $G$  of feasible and potential solutions of a problem.

functions: The more time spent on calculating the bound, the better the bound value usually is. It is normally considered beneficial to use as strong a bounding function as possible in order to keep the size of the search tree as small as possible.

Bounding functions naturally arise in connection with the set of potential solutions  $G$  and the function  $g$  mentioned in above. Due to the fact that  $S \subseteq G$ , and that  $g(x) \leq f(x)$  on  $G$ , the following is easily seen to hold:

$$\min_{x \in G} g(x) \leq \left\{ \begin{array}{l} \min_{x \in G} f(x) \\ \min_{x \in S} g(x) \end{array} \right\} \leq \min_{x \in S} f(x) \quad (3.7)$$

If both of  $G$  and  $g$  exist there are now choices between three optimization problems, for each of which the optimal solution will provide a lower bound for the given objective function. The "skill" here is of course to chose  $G$  and/or  $g$  so that one of these is easy to solve and provides tight bounds.

### 3.2.3 Branching rule

All branching rules in the context of B&B can be seen as subdivision of a part of the search space through the addition of constraints, often in the form of assigning values to variables. Convergence of B&B is ensured if the size of each generated subproblem is smaller than the original problem, and the number of feasible solutions to the original

problem is finite. Normally, the subproblems generated are disjoint - in this way the problem of the same feasible solution appearing in different subspaces of the search tree is avoided.

### 3.2.4 Strategy for selecting next subproblem

The strategy for selecting the next live subproblem to investigate usually reflects a trade off between keeping the number of explored nodes in the search tree low, and staying within the memory capacity of the computer used.

If one always selects among the live subproblems one of those with the lowest bound, called the *best first search* strategy, *BeFS*. Fig. 3.3 shows a small search tree -the numbers in each node corresponds to the sequence. A subproblem P is called critical if the given bounding function when applied to P results in a value strictly less than the optimal solution of the problem in question. Nodes in the search tree corresponding to critical subproblems have to be partitioned by the B&B algorithm no matter when the optimal solution is identified - they can never be discarded by means of the bounding function. Since the lower bound of any subspace containing an optimal solution must be less than or equal to the optimum value, only nodes of the search tree with lower bound less than or equal to this will be explored.

Even though the choice of the subproblem with the current lowest lower bound makes good sense also regarding the possibility of producing a good feasible solution, memory problems arise if the number of critical subproblems of a given problem becomes too large. The situation more or less corresponds to a *breath first search* strategy, *BFS*, in which all nodes at one level of the search tree are processed before any node at a higher level. Fig. 3.4 shows the search tree with the numbers in each node corresponding to the BFS processing sequence. The number of nodes at each level of the search tree grows exponentially with the level making it infeasible to do breadth first search for larger problems.

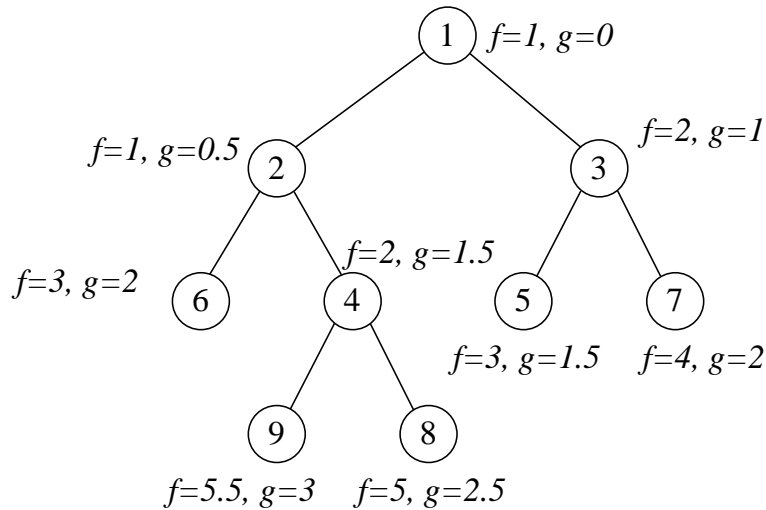


Figure 3.3: Search strategies in B&B: the Best First Search.

The alternative used is a *depth first search* strategy, *DFS*. Here a live node with largest level in the search tree is chosen for exploration. Fig. 3.5 shows the DFS processing sequence number of the nodes. The memory requirement in terms of number of subproblems to store at the same time is now bounded above by the number of levels in the search tree multiplied by the maximum number of children of any node, which is usually a quite manageable number. An advantage from the programming point of view is the use of recursion to search the tree - this enables one to store the information about the current subproblem in an incremental way, so only the constraints added in connection with the creation of each subproblem need to be stored. The drawback is that if the incumbent is far from the optimal solution, large amounts of unnecessary bounding computations may take place. In order to avoid this, DFS is often combined with a selection strategy which is that exploring the node with the small lower bound first hopefully leads to a good feasible solution.



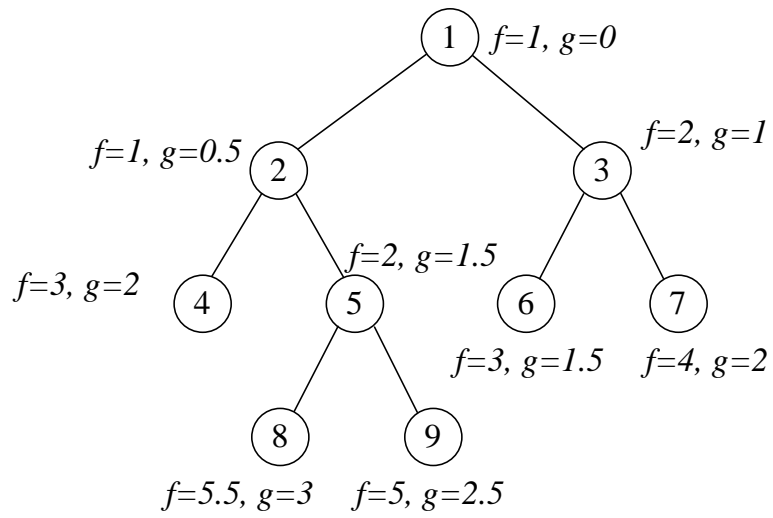


Figure 3.4: Search strategies in B&B: the Breath First Search.

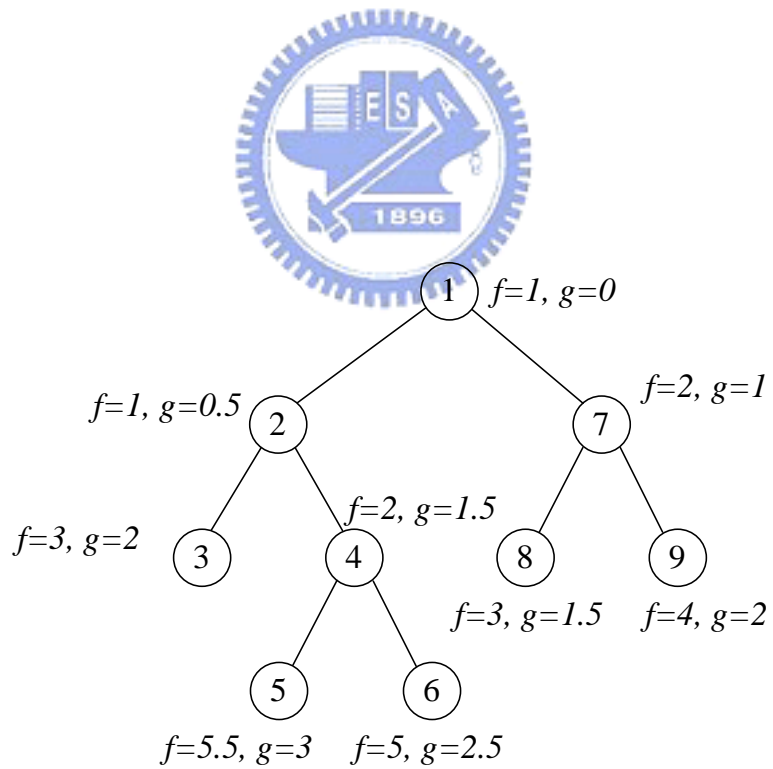


Figure 3.5: Search strategies in B&B: the Depth First Search.

## Chapter 4

# Mono-Rate Wireless Resource Allocation

We begin with the simplest case that there is only one rate (user) requirement. Although the mono-rate problem seems easy, the solution we obtained for this case will be used for the more complex and realistic situation when there are multiple rate requests.

Since there is only one request, all channels are used to carry the same data type and the channel allocation matrix  $\mathbf{A}$  is fixed and (2.8) is reduced to

$$\min_{\mathbf{P}} \sum_{i=1}^N p_i \quad s.t. \quad \sum_{i=1}^N r_i = R, \quad 0 \leq p_i \leq \bar{p}_i, \quad 1 \leq i \leq N. \quad (4.1)$$

As mentioned before, the peak power constraint  $0 \leq p_i \leq \bar{p}_i, 1 \leq i \leq N$  can be replaced by the peak rate constraint  $0 \leq r_i \leq \bar{r}_i, 1 \leq i \leq N$ . Moreover, as there is only one data type, the GNR  $(a_{ij})$  can be replaced by a simpler notation  $(a_i)$  for obvious reason. Two algorithms are introduced in this chapter. One is the conventional greedy approach and the second one is a new iterative optimization algorithm.

### 4.1 Conventional Greedy Algorithm

Finding the minimum required transmitted power for transmitting a single specific rate request over multiple orthogonal channels is a nonlinear programming problem. For a solution in nonlinear programming to be optimal, it must satisfy the Karush-Kuhn-

Tucker (KKT) conditions which can be regarded as a generalization of the method of Lagrange multipliers.

**Theorem 4.1.1. Karush-Kuhn-Tucker (KKT) Theorem** *Let  $x^*$  be a regular point and a local minimizer for the problem of minimizing  $f$  subject to  $h(x) = 0$ ,  $g(x) \leq 0$ . Then, there exist  $\lambda^* \in \mathbb{R}^m$  and  $\mu^* \in \mathbb{R}^m$  such that*

(a)  $\mu^* \geq 0$ ;

(b)  $Df(x^*) + \lambda^{*T} Dh(x^*) + \mu^{*T} Dg(x^*) = 0^T$ ;

(c)  $\mu^{*T} g(x^*) = 0$ . ■

Note that  $\mu_j^* \geq 0$  (by (a)) and  $g_j(x^*) \leq 0$ , thus the condition

$$\mu^{*T} \mathbf{g}(x^*) = \mu_1^* \mathbf{g}_1(x^*) + \dots + \mu_p^* \mathbf{g}_p(x^*) = \mathbf{0} \quad (4.2)$$

implies that if  $g_j(x^*) < 0$ ,  $\mu_j^*$  must be zero. Then, the KKT conditions of the mono-rate problem are

$$\begin{aligned} f(p_i) &= \sum_{i=1}^N p_i - \lambda \left( \sum_{i=1}^N \ln(1 + a_i p_i) - R \right) - \mu_i^0 p_i + \mu_i^P (p_i - \bar{p}_i) \\ h(p_i) &= \sum_{i=1}^N \ln(1 + a_i p_i) - R \\ g_1(p_i) &= -p_i \\ g_2(p_i) &= p_i - \bar{p}_i \end{aligned} \quad (4.3)$$

From the *KKT Theorem*, we arrive at the following equivalent necessary conditions for all  $1 \leq i \leq N$ ,

Condition 1:

$$\frac{\partial f}{\partial p_i} = \frac{\partial}{\partial p_i} \left[ \sum_{i=1}^N p_i - \lambda \left( \sum_{i=1}^N \ln(1 + a_i p_i) - R \right) - \mu_i^0 p_i + \mu_i^P (p_i - \bar{p}_i) \right] = 0 \quad (4.4)$$

Condition 2:

$$-(\mu_i^0 p_i) = 0, \text{ if } p_i = 0, \mu_i^0 \geq 0; \quad \text{otherwise } -p_i < 0, \mu_i^0 = 0 \quad (4.5)$$

Condition 3:

$$\mu_i^P(p_i - \bar{p}_i) = 0, \text{ if } (p_i - \bar{p}_i) = 0, \mu_i^P \geq 0; \quad \text{otherwise } (p_i - \bar{p}_i) < 0, \mu_i^P = 0 \quad (4.6)$$

Condition 4:

$$\sum_{i=1}^N r_i = \sum_{i=1}^N \ln(1 + a_i p_i) = R \quad (4.7)$$

Obviously, there is no closed-form expressions for the optimal power allocation  $p_i$  of the  $i$ th VOC and the Lagrange multipliers  $\lambda, \mu_i$ , except for the degenerate case. However, one can use some numerical methods, ex. Newton method, Gradient method to search for  $\lambda$  and compute the optimal power distribution, both of which are subject to the same set of constraints. As the range of  $\lambda$  is a continuous region any of conventional numerical approach requires high computing complexity.

## 4.2 An Iterative Optimization Algorithm

Even with an exhaustive search, the true optimal  $\lambda$  is hard to come by. We propose an iterative Lagrange-type algorithm. Unlike earlier proposals in which the channels to be excluded (i.e., those  $i$  with  $p_i = 0$ ) are found only after the optimal level  $\lambda$  is known, our approach first selects some “bad” channels to be excluded and then perform power allocation on the remaining channels. The optimal solution can be easily found within a few iterations.

We sort all available channels in descending order of channel GNR, i.e., the channel indexing is such that  $a_1 \geq a_2 \geq \dots \geq a_N$  and assume that  $0 < p_i < \bar{p}_i$  for all  $i$ . The later assumption, by Conditions 2 and 3, then imply that  $\mu_i^0 = 0$  and  $\mu_i^P = 0$  for all  $i$ . Suppose we use the first  $x$  channels and denote by  $p_i(x), r_i(x)$  the allocated transmission power and normalized rate for the  $i$ th channel, then  $p_i(x) = r_i(x) = 0$ , for  $x < i \leq N$  and, furthermore, Condition 1 (4.4) becomes

$$f'(\mathbf{p}(x)) = \frac{\partial}{\partial p_i} \left[ \sum_{i=1}^N p_i - \lambda \left( \sum_{i=1}^N \ln(1 + a_i p_i) - R \right) \right] = 0 \quad (4.8)$$

where  $\mathbf{p}(x) \stackrel{\text{def}}{=} (p_1(x), p_2(x), \dots, p_x(x))$ , we obtain

$$1 - \frac{\lambda a_i}{1 + a_i p_i} = 0 \implies \lambda a_i = (1 + a_i p_i) \quad (4.9)$$

Multiplying (4.9) for  $1 \leq i \leq x$ , we obtain

$$\begin{aligned} \lambda^x \prod_{i=1}^x a_i &= \prod_{i=1}^x (1 + a_i p_i) \\ x \ln \left[ \lambda \left( \prod_{i=1}^x a_i \right)^{\frac{1}{x}} \right] &= \ln \left[ \prod_{i=1}^x (1 + a_i p_i) \right] \\ &= \sum_{i=1}^x \ln(1 + a_i p_i) \\ &= R \\ \implies \lambda &= \frac{e^{R/x}}{\hat{a}(x)}, \quad \hat{a}(x) \stackrel{\text{def}}{=} \left[ \prod_{i=1}^x a_i \right]^{1/x} \end{aligned} \quad (4.10)$$

and

$$\begin{aligned} r_i(x) &= \ln(1 + a_i p_i) = \ln(\lambda a_i) = \ln \left[ e^{R/x} \cdot \frac{a_i}{\hat{a}(x)} \right] \\ &= \frac{R}{x} + \ln \left( \frac{a_i}{\hat{a}(x)} \right), \quad i = 1, 2, \dots, x \end{aligned} \quad (4.11)$$

It is clear that, for a fixed  $x$ ,  $r_i(x)$  is a decreasing function of  $i$ , and the above equation is the unconstrained solution  $\mathbf{r}(x; N) \stackrel{\text{def}}{=} (\mathbf{r}(x), \mathbf{0}_{1 \times (N-x)})$  to the problem

$$\min_P \sum_{i=1}^N p_i \quad s.t. \quad \sum_{i=1}^N r_i = R \quad (4.12)$$

One of these unconstrained solutions  $\mathbf{r}(x; N), x = N, N-1, \dots, 1$  is the solution to (4.1). The first step to establish this conclusion is

**Lemma 4.2.1.** *The sequence  $\{r_x(x), x = 1, 2, \dots, N\}$  is monotonically decreasing. ■*

**Proof.** Lemma follows directly from the relation

$$\begin{aligned} r_x(x) &= \frac{R}{x} + \ln \left[ \frac{a_x}{\hat{a}(x)} \right] \\ &= \frac{R}{x} - \frac{1}{x} \sum_{i=1}^{x-1} \ln a_i + \frac{x-1}{x} \ln a_x \\ &= \frac{x-1}{x} \left[ r_{x-1}(x-1) + \ln \left( \frac{a_x}{a_{x-1}} \right) \right] \end{aligned} \quad (4.13)$$

and the fact that  $a_1 \geq a_2 \geq \dots \geq a_N$ . □

We also need the definition

**Definition 4.2.1.** An unconstrained solution  $\mathbf{r}(x, N)$  is said to be admissible if  $r_x(x) > 0$ . The admissible active channel number sets for the problem defined by (4.1) is defined by  $\mathbf{F} = \{x | r_x(x) > 0, 1 \leq x \leq N\}$ , where  $r_x(x)$  is given by (4.11). ■

With this definition we can show that

**Lemma 4.2.2.** The total transmitted power associated with the admissible unconstrained optimal rate assignment (4.11) is also a decreasing function of the number of channels used. In other words,  $N_1 < N_2 \implies \sum_{i=1}^{N_1} p_i(N_1) > \sum_{i=1}^{N_2} p_i(N_2)$ , for  $N_1, N_2 \in \mathbf{F}$ . ■

**Proof.** To begin with, let us assume that  $N_1 = N$  and  $N_2 = N + 1$ . If the above Lemma is true in this case, it will also be true for the other case  $N_2 - N_1 > 1$ .

$$p_i(x) = \frac{e^{r_i(x)} - 1}{a_i}, \quad 1 \leq i \leq x, \quad (4.14)$$

The minimum power of the case  $x = N$  is given by

$$\begin{aligned} \tilde{P}_N &= \sum_{i=1}^N \frac{e^{r_i(N)} - 1}{a_i} = \sum_{i=1}^N \left[ \frac{e^{R/N}}{\hat{a}(N)} - \frac{1}{a_i} \right] \\ &= N \cdot \frac{e^{R/N}}{\hat{a}(N)} - \sum_{i=1}^N \frac{1}{a_i} \end{aligned}$$

where  $\hat{a}(N) = \left[ \prod_{i=1}^N a_i \right]^{\frac{1}{N}}$ . The minimum power of the case  $x = N + 1$  can be described as the function of  $r_{N+1}$ .

$$\begin{aligned} \tilde{P}_{N+1} &= \tilde{P}'_N + p_{N+1} = \sum_{i=1}^N \frac{e^{r'_i(N)} - 1}{a_i} + p_{N+1} \\ &= \sum_{i=1}^N \left[ \frac{e^{(R-r_{N+1}(N+1))/N}}{\hat{a}(N)} - \frac{1}{a_i} \right] + p_{N+1} \\ &= N \cdot \frac{e^{(R-r_{N+1}(N+1))/N}}{\hat{a}(N)} - \sum_{i=1}^N \frac{1}{a_i} + \frac{e^{r_{N+1}(N+1)} - 1}{a_{N+1}} \end{aligned} \quad (4.15)$$

The difference between  $\tilde{P}_N$  and  $\tilde{P}_{N+1}$  can be expressed as the function of  $r_{N+1}(N+1)$

$$\begin{aligned} f(r_{N+1}(N+1)) &= \tilde{P}_N - \tilde{P}_{N+1} \\ &= \frac{N}{\hat{a}(N)} \left( e^{R/N} - e^{(R-r_{N+1}(N+1))/N} \right) - \frac{e^{r_{N+1}(N+1)} - 1}{a_{N+1}} \end{aligned} \quad (4.16)$$

$$f'(r_{N+1}(N+1)) = \frac{\partial f(r_{N+1}(N+1))}{\partial r_{N+1}(N+1)} = \frac{e^{(R-r_{N+1}(N+1))/N}}{\hat{a}(N)} - \frac{e^{r_{N+1}(N+1)}}{a_{N+1}} \quad (4.17)$$

Therefore, the solution of  $f'(r_{N+1}(N+1)) = 0$ ,  $r_{N+1}^*(N+1)$ , is given by

$$\begin{aligned} r_{N+1}^*(N+1) &= \frac{R}{N+1} + \frac{N}{N+1} \cdot \ln \left[ \frac{a_{N+1}}{\hat{a}(N)} \right] \\ &= \frac{R}{N+1} + \ln \left[ \frac{a_{N+1}}{\hat{a}(N+1)} \right] \end{aligned} \quad (4.18)$$

The second derivative for  $f(r_{N+1}(N+1))$  reads

$$\begin{aligned} f^{(2)}(r_{N+1}(N+1)) &= \frac{\partial f'(r_{N+1}(N+1))}{\partial r_{N+1}(N+1)} \\ &= \frac{-1}{\hat{a}(N)a_{N+1}} \left\{ \frac{a_{N+1}}{N} e^{[R-r_{N+1}(N+1)]/N} + \hat{a}(N)e^{r_{N+1}(N+1)} \right\} \end{aligned} \quad (4.19)$$

which implies that  $f^{(2)}(r_{N+1}(N+1)) < 0$ , for  $0 \leq r_{N+1}(N+1) < R$ . Since  $f'(r_{N+1}^*(N+1)) = 0$  and  $f(0) = 0$ , we have  $f'(r_{N+1}(N+1)) > 0$ , for  $0 \leq r_{N+1}(N+1) < R$  and  $f(r_{N+1}^*(N+1)) > 0$ .

Hence, the minimum power for the case  $x = N$  is larger than that for the case  $x = N+1$ , which can be achieved with  $r_{N+1}(N+1) = r_{N+1}^*(N+1)$ .  $\square$

Based on the above Lemma and the fact that the optimal  $x^*$  is such that  $r_{x^*}(x^*) > 0$  and  $r_{x^*+1}(x^*+1) < 0$ , we conclude that the optimal solution to (4.1) can be obtained by repeating the following steps

$$\begin{aligned} x^* &= \max_{y \in \mathbf{F}} y \\ r_i(x^*) &= \frac{R}{x^*} + \ln \left[ \frac{a_i}{\hat{a}(x^*)} \right], \quad 1 \leq i \leq x^* \\ \mathbf{r}^* &= (\mathbf{r}(x^*), \mathbf{0}_{1 \times (N-x^*)}) \end{aligned} \quad (4.20)$$

and modifying the number of channels involved  $x$  until  $r_i \geq 0$ , for  $1 \leq i \leq N$ . This algorithm obtains the optimal rate allocation vector ( $\mathbf{r}^*$ ) with the rate constraint  $0 \leq r_i$  for all  $i$ .

However, there is still the rate constraint,  $r_i \leq \bar{r}_i$ , for all  $i$ , to be met. We redefine the set for the rate requirements and the components of the rate allocation vector (4.20)

$$\begin{aligned}
C_{\bar{r}} &= \{i \mid 1 \leq i \leq N, r_i \geq \bar{r}_i\}, \\
C^* &= C^* \cup C_{\bar{r}}, \\
C_r &= \{i \mid 1 \leq i \leq N, i \notin C^*\}, \\
r_i &= \bar{r}_i, \quad i \in C_{\bar{r}}
\end{aligned} \tag{4.21}$$

and modify the constraint rates

$$R' = R - \sum_{i \in C_{\bar{r}}} r_i \tag{4.22}$$

The optimal solution with the additional rate constraint  $r_i \leq \bar{r}_i$ , for all  $i$  can be found by repeatedly applying the above iterative algorithm (4.20) to solve

$$\min_{\mathbf{P}} \sum_{i \in C_r} p_i \quad s.t. \quad \sum_{i \in C_r} r_i = R', \quad 0 \leq p_i, \quad i \in C_r \tag{4.23}$$

until  $C_{\bar{r}} = \{\emptyset\}$ . Combining the two searching phases, we obtain the optimal rate allocation vector that satisfies the rate constraint  $0 \leq r_i \leq \bar{r}_i$ , for all  $i$ . The flow chart of this iterative algorithm is illustrated in Fig. 4.1.

**Example 4.1.** *We wish to minimize the required power to achieve a desired normalized data rate of 3 over five channels. The GNR's and power constraints of these channels are (1.0, 0.8, 0.6, 0.4, 0.2) and (2w, 2w, 2w, 2w, 2w), respectively. Applying the proposed algorithm, we obtain the corresponding numerical results and the final solution as shown in Fig. 4.2.*

□



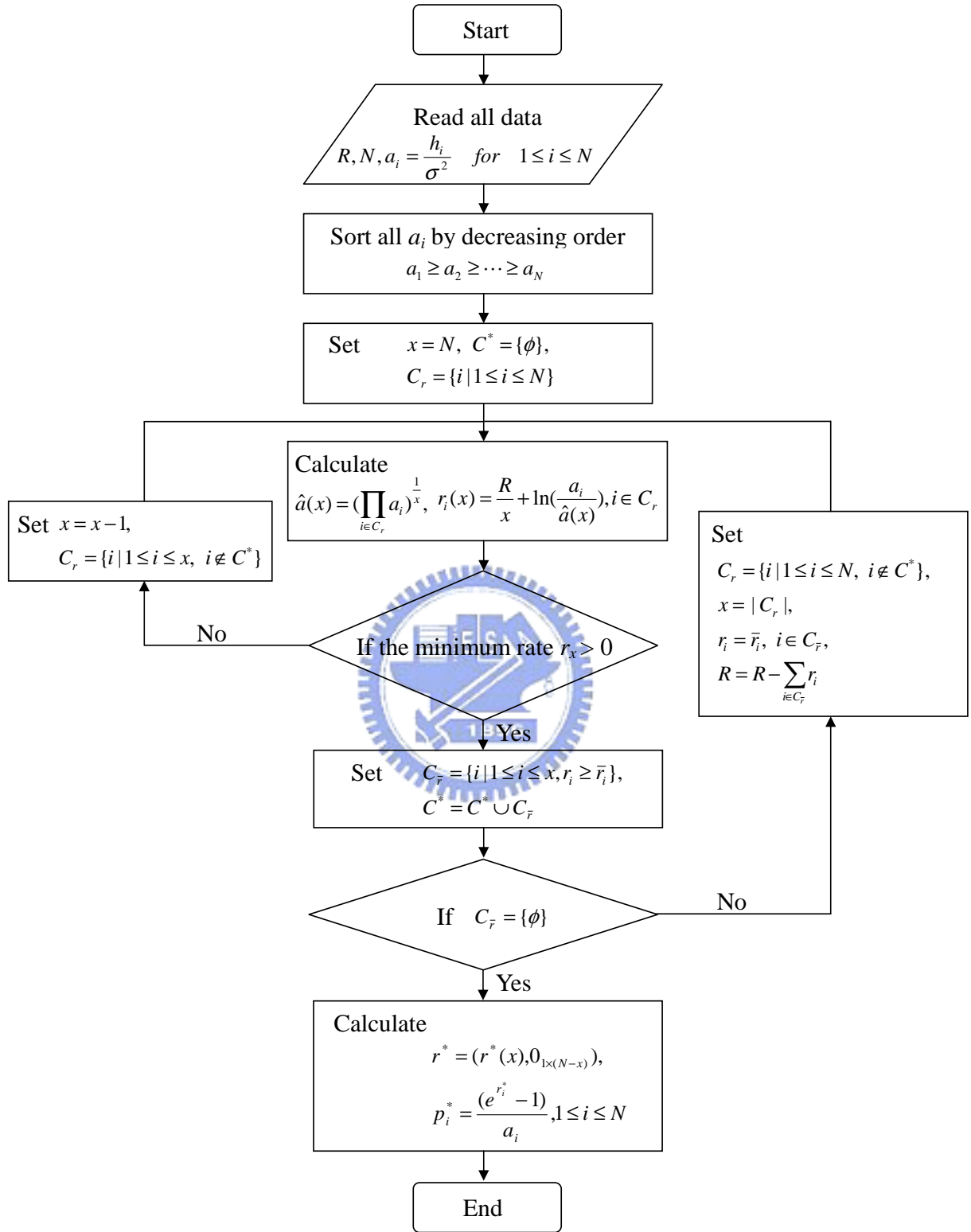


Figure 4.1: The iterative optimization algorithm for fixed mono-data rate

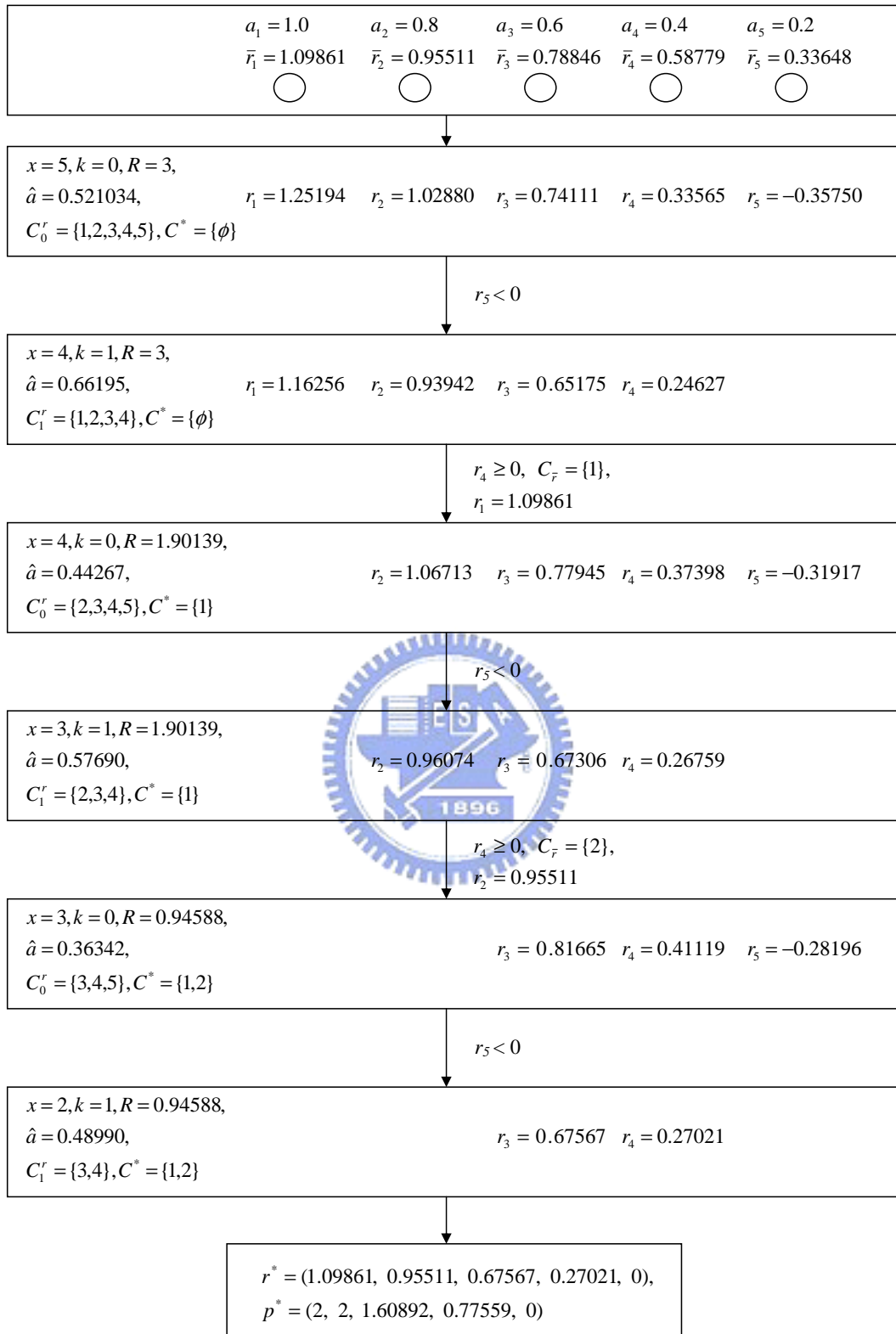


Figure 4.2: Numerical results obtained in the course of applying the algorithm of Fig. 4.1 to solve Example 4.1.

### 4.2.1 Computational complexity

To analyze the complexity of the proposed iterative optimization algorithm, the complexity for computing the closed-form formula

$$r_x(x) = \frac{R}{x} + \ln \left[ \frac{a_i}{\hat{a}(x)} \right] \quad (4.24)$$

is used as a reference. (4.20) shows that for finding the  $x^*$  the binary search can be applied to reduce the computing time of (4.24). After  $x^*$  is obtained one still needs to compute the remaining candidate rates  $r_i(x^*)$ ,  $1 \leq i \leq x^*$ , the overall complexity for (4.20) is thus given by  $O(\log_2 N + x^*) \leq O(\log_2 N + N)$ .

Next, if some VOCs have allocated power exceeding their respective constraints, the formulas (4.21) and (4.22) have to be considered. However, we cannot perfectly predict how many times the formula (4.20) will be repeated. For this reason, some cases are simulated in Fig. 4.3 to show the complexity of the mono-rate iterative algorithm.

In this simulation, the data rate and the noise power are assumed to be 5 and 0.01. To avoid the capacity of the transmitted power increases with the number of VOCs, the total power constraint is assumed in Fig. 4.3. Thus, the power constraint of each VOC is equal to the total power constraint divided by the number of VOCs.

Fig. 4.3 reveals that the required computing times of the closed-form based approach is an increasing function of the number of VOCs. The tighter the total power constraint becomes, the more VOCs will reach their power constraints. For this reason, the computing load for (4.20) increases accordingly which is shown in Fig. 4.3. Finally, we notice that when the total power constraint equals 1 w, the power constraint on each VOC is seldom to be violated, so the complexity given by Fig. 4.3 is approximately  $O(\log_2 N + x^*) \leq O(\log_2 N + N)$ .

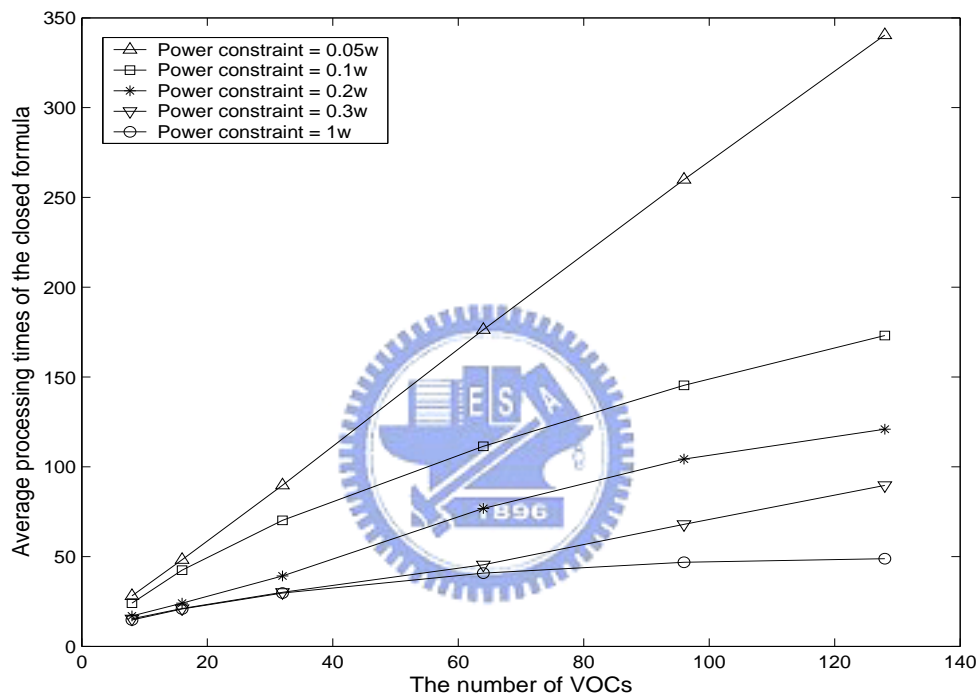


Figure 4.3: Average complexities of the proposed iterative power allocation algorithm for mono-rate transmission with different total transmit power constraints

# Chapter 5

## Multi-Rate Wireless Resource Allocation

Based on the mono-rate algorithm of Fig. 4.1, the following algorithm is designed for multi-rate applications. The first algorithm is a conventional greedy approach that conducts an exhaustive search over all possible channel assignment matrices  $\mathbf{A}_{\mathbf{N} \times \mathbf{d}}$ . Although this algorithm is guaranteed to yield the optimal  $\mathbf{A}_{\mathbf{N} \times \mathbf{d}}$ , the searching process is both complicated and time-consuming, especially if the numbers of data rates (types) and/or VOCs are large. We apply the DP technique to derive a simple and practical solution which requires much reduced complexity at the cost of minor performance loss. To recover the potential performance loss, we further present a solution based on the B&B principle which entails modest computing complexity but gives the optimal solution with certainty.

### 5.1 A Greedy Algorithm

The optimal solution can be obtained by a conventional two-step procedure like that described in [10]. One first finds the optimal  $P$  for a fixed  $\mathbf{A}_{\mathbf{N} \times \mathbf{d}}$  by using the iterative water-filling procedure of Fig. 4.1 and then conducting an exhaustive search for the optimal  $\mathbf{A}_{\mathbf{N} \times \mathbf{d}}$  that gives the minimum total transmitted power while meeting all constraints. For a given  $\mathbf{A}_{\mathbf{N} \times \mathbf{d}}$ , the problem of computing the optimal power allocation

vector can be divided into  $d$  simple optimization subproblems defined by

$$C(j) = \{ i \mid 1 \leq i \leq N, A_{ij} = 1 \}$$

$$\min \sum_{i=1}^N \sum_{j=1}^d p_{ij}, \quad s.t. \quad \sum_{i \in C(j)} r_{ij} = R_j, \quad 0 \leq p_{ij} \leq \bar{p}_i, \quad 1 \leq i \leq N, \quad 1 \leq j \leq d \quad (5.1)$$

A fully search is performed to find the optimal assignment matrix  $\mathbf{A}_{N \times d}$ . We compute the minimum transmitted power for each desired rate by running the above iterative water-filling method within  $d$  subproblems and calculate their sum as the minimum total power with respect to a given  $\mathbf{A}_{N \times d}$ . To acquire this solution, we have to compute the optimal channel assignment matrix and power allocation matrix. However, the exhaustive-searching process is too much complicated and time-consuming to be implemented, especially if the numbers of data rates (types) and virtual channels are large. In the next section, we propose a simple and practical algorithm by exploring the dynamic programming method which can reduce a lot of computed complexity with only minor performance loss.



## 5.2 Dynamic Programming Based Resource Allocation Algorithm

### 5.2.1 Dynamic programming formulation

For introducing the dynamic programming optimization algorithm to simplify searching the assignment matrix  $\mathbf{A}_{N \times d}$ , we must modify the problem formulation (2.8) into the dynamic programming format (3.6).

First, for the initial problem of each data type subset, in opposition to the intuitional idea that each data type subset is initialized as an empty subset and VOCs are assigned at each stage, we propose an innovative thought which all VOCs are initialized to each data type and VOCs are deleted at each stage. There are two advantages for this method:

- Easy to have initial values:

Since each data type subset has already been allocated all VOCs, we can easily to calculate the transmitted power of each data type subset. However, for the common idea, there is an empty subset of each data type, so the initial value is hard to set.

- Simple to apply power constraints of the VOCs:

The number of VOCs in each data type subset is decreasing stage by stage, so simultaneously, the transmitted power of each data type is increasing. Thus, if the deleted VOC will cause this data type subset cannot afford its desired rate, this VOC will be chosen to stay in this data type and the implement of power constraint can be achieved easily. Nevertheless, if the VOC is added at each stage, we cannot guarantee if the final channel allocation can satisfy all power constraints.

Next, as the assumption that each VOC can serve only one data type, so which data type should be served by this VOC has to be determined stage by stage. Additionally, to save the cost of memory and reduce the computational complexity, only the state with the minimum sum of all data types' transmitted power is survived at each stage. Therefore, the data type corresponding to the state with the minimum power will keep the VOC which is allocated at this stage and the VOC will be removed from all other data type subsets.

Third, the order of VOC to be determined also affects the performance of DP algorithm much. By the intuition, the VOC with better channel response needs to be allocated carefully to avoid large performance loss from the optimal solution. For this reason, we choose the maximum channel response of the VOC to each data type to stand for this VOC and the order of VOC to be allocated depends on this maximum value. This method is shown as the following formula,

$$Sort_i [ max_j (a_{ij}) ], 1 \leq i \leq N, 1 \leq j \leq d \quad (5.2)$$

Finally, the stage by stage searching trellis with the DP format is illustrated below

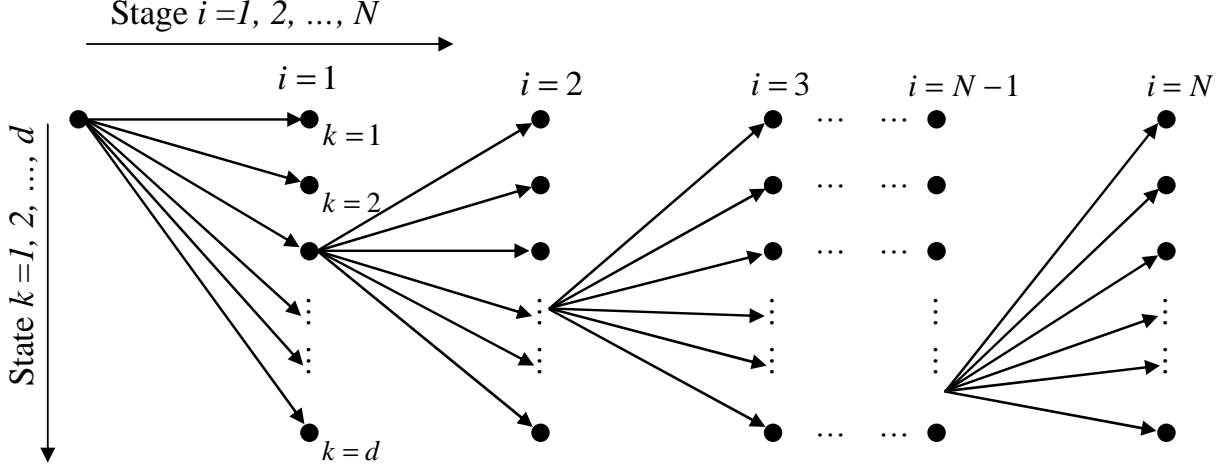


Figure 5.1: The problem formulation modified to DP format

## 5.2.2 A dynamic programming approach

To apply the DP technique to solve (5.1) stage by stage, we have to define the following subsets and functions,

$C_t^r$  the survived VOC's subset vector at the  $t$  stage, where  $C_t^r = (C_t^r(1), \dots, C_t^r(d))$ ,

$C_t^r(j)$  the survived VOC's subset of the  $j$ th data type at the  $t$  stage,

$g(R, C)$  the total consuming power by the iterative optimization algorithm of Fig. 4.1 with the require normalize rate  $\mathbf{R}$  and the VOC's subset  $\mathbf{C}$ ,

$f_k(C_t^r(j))$  a function that describes the subset by which the state is updated,

$$\text{where } f_k(C_t^r(j)) = \begin{cases} C_{t-1}^r(j) & , \quad j = k \\ C_{t-1}^r(j) \setminus t & , \quad j \neq k \end{cases}$$

The VOC's subset of each data type is initialized as  $C_0^r(j) = \{i \mid 1 \leq i \leq N\}$ ,  $1 \leq j \leq d$ . Then, the cost formula,

$$J_t(C_t^r) = \min_{k, 1 \leq k \leq d} \left\{ \sum_{j=1}^d g(R_j, f_k(C_t^r(j))) \right\}, \quad 1 \leq t \leq N \quad (5.3)$$



and the VOC's subset of each data type are updated each stage to implement DP method for solving the problem of searching a near optimal channel assignment matrix  $\mathbf{A}_{N \times d}$  and this algorithm will be referred as **DPRA** algorithm in the following chapter.

### 5.2.3 Numerical behavior

As we describe above, the solution provided by *DPRA* approach is not always optimal. The following simulated results show the performance of *DPRA* method. To avoid the sum of the normalized rates increases with the number of data types, we define *the normalized sum rate*  $= \sum_{j=1}^d R_j$ . Various numbers of data types with the same normalized sum rate are simulated and the normalized rate of each data type is randomly assigned. For simplicity, the normalized noise power ( $\sigma^2$ ) is assumed to be *0.01*. The results are the average of the *100000* simulated times.

From Fig. 5.2, the simulated results have shown the probability of *DPRA* algorithm achieving the optimal solution. Compared 64 VOCs with 128 VOCs, since the number of better VOCs in each data type subset increases and fewer VOCs are better in more than one data type subset, the correct probability of the decision at each stage by *DPRA* method will raise. On the opposite, if the number of data types increases, it means selections at each stage become more, so the probability of reaching the optimal solution should degrade as the results.

As for the uplink scenario in Fig. 5.4, since the GNR of the VOC is the same to each data type, VOCs are contended by every data type. Better VOCs are important to all data types, so it becomes hard to decide these VOCs belong to which data type. For this reason, the probability of achieving the optimum among the uplink scenario decays much fast than it among the downlink scenario. Opposite to the downlink, more number of VOCs means more VOCs used in every data type in the initial state, so more probability of wrong decision occurring.

Except for the probability of achieving the optimal solution, the average performance

loss is an useful evaluation to a sub-optimal algorithm, so Fig. 5.3 illustrates the average performance loss from the optimal solution by *DPRA* approach. With fewer VOCs, a wrong decision will introduce more performance loss than it with more VOCs. Additionally, the probability of achieving the optimum with fewer VOCs is lower than more VOCs, so the average performance with fewer VOCs is obviously worst than it with more VOCs.

From Figs. 5.4 and 5.5, the probability among the uplink scenario decreases much fast than it among the downlink scenario. In addition, if one good VOC is assigned to some data type, it means other data types all lose this good VOC. Thus, the performance loss of the uplink scenario is obviously worse than the downlink scenario. Beyond that, compared  $N = 10$  with  $N = 15$ , since there are more useful VOCs among 15 VOCs than 10 VOCs, wrong decision by *DPRA* method will introduce less performance loss. As for the effect from the number of data types, the uplink scenario appears like the downlink scenario. The more number of data types, the more performance loss.

Consequently, even though the probability of achieving the optimum solution is not very high when the normalized sum rate is high, the average performance loss is still kept within a tolerable range. In other words, it indicates that the performance *DPRA* algorithm is very close to the optimal.

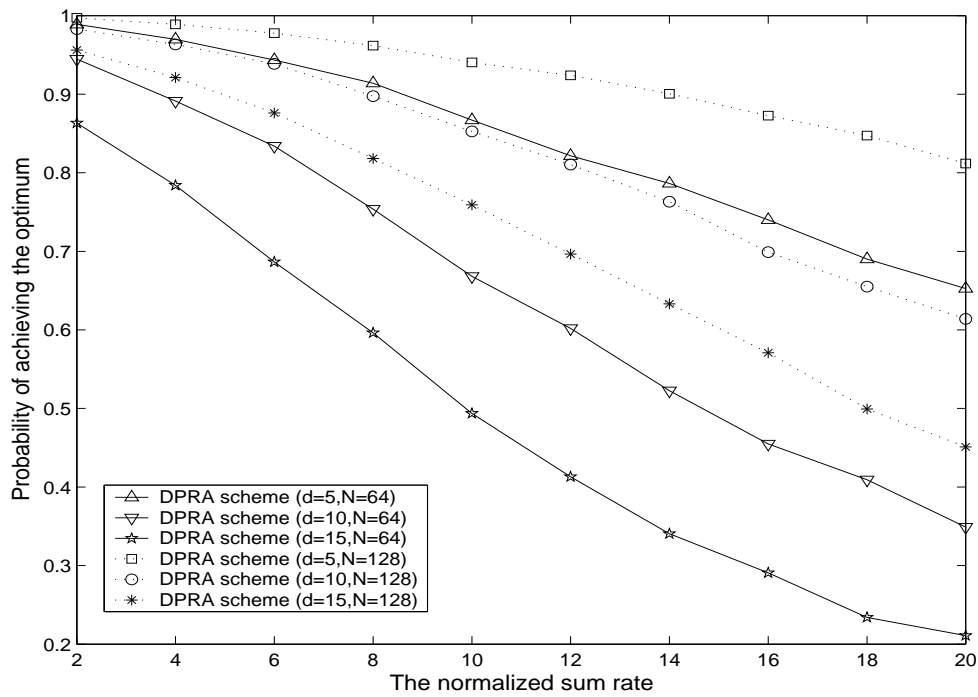


Figure 5.2: Probabilities of *DPRA* algorithm achieving the optimum solution in a downlink scenario.

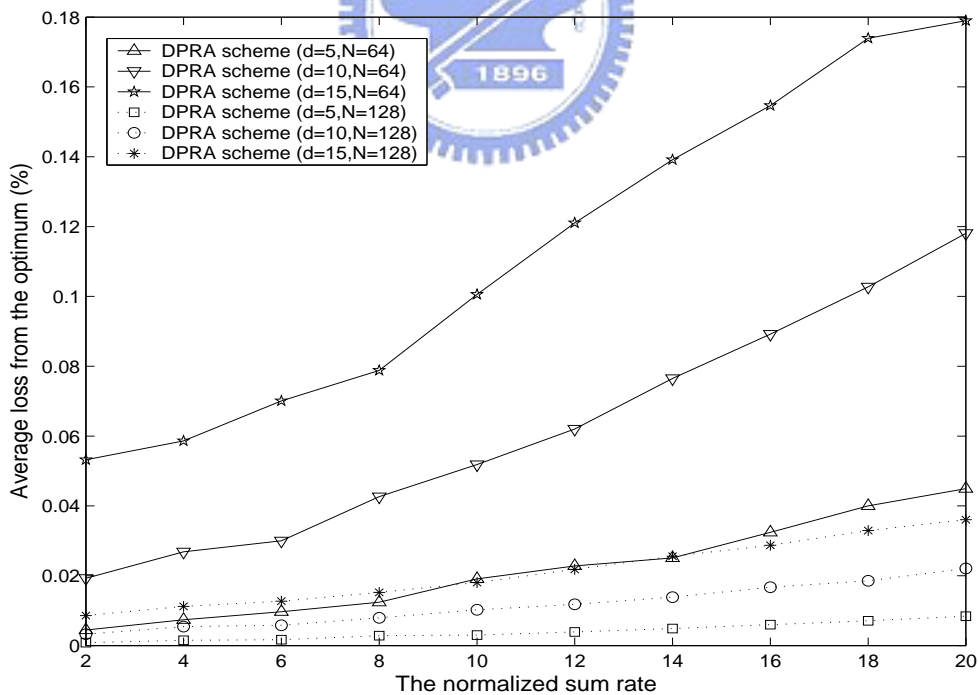


Figure 5.3: Average performance loss (with respect to the optimum solution) of *DPRA* algorithm in a downlink scenario.

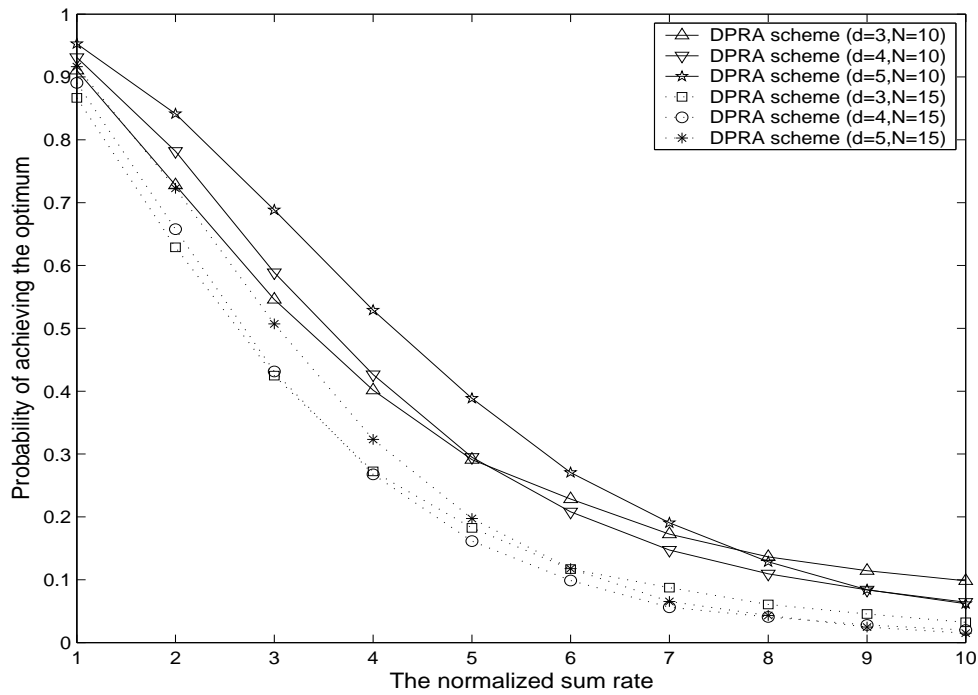


Figure 5.4: Probabilities of *DPRA* algorithm achieving the optimum solution in an uplink scenario.

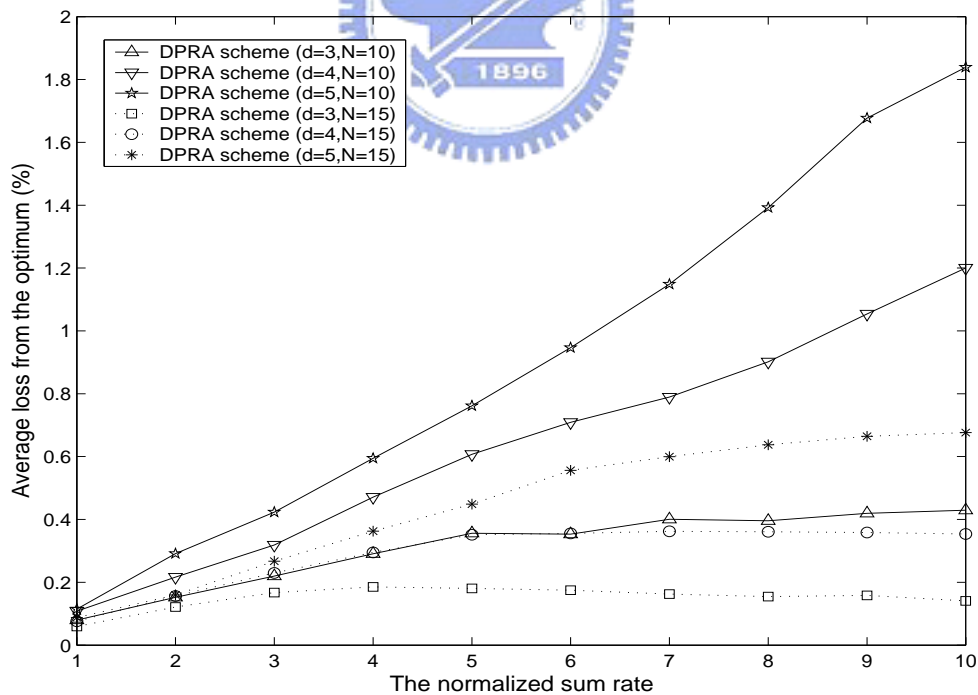


Figure 5.5: Average performance loss (with respect to the optimum solution) of *DPRA* algorithm in an uplink scenario.

**Example 5.1.** Consider the situation where there are 5 VOCs available for serving 3 different data types with the required rates given by

$$[d_1 \ d_2 \ d_3] = [1 \ 3 \ 2]$$

and the GNRs given by

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ a_{51} & a_{52} & a_{53} \end{bmatrix} = \begin{bmatrix} 0.3 & 2.0 & 1.5 \\ 0.7 & 0.2 & 0.4 \\ 0.4 & 0.5 & 0.9 \\ 1.5 & 1.0 & 0.7 \\ 0.1 & 0.7 & 1.0 \end{bmatrix}$$

The trellis and the associated parameters' values of the DPRA algorithm are given in Figs. 5.6 and 5.1.

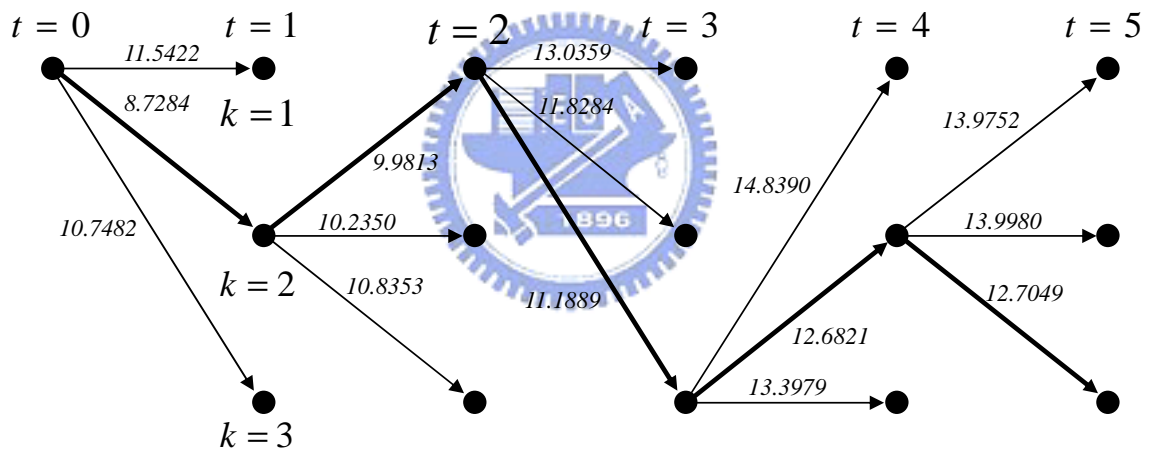


Figure 5.6: Trellis associated with the DPRA algorithm and the corresponding numerical values in solving Example 5.1.

Table 5.1: Numerical results obtained in solving Example 5.1.

Initial value ( $t = 0$ ) and ( $R_1 = 1, R_2 = 3, R_3 = 2$ )						
$C_0^r(1)$	$g(R_1, C_0^r(1))$	$C_0^r(2)$	$g(R_2, C_0^r(2))$	$C_0^r(3)$	$g(R_3, C_0^r(3))$	
{1,2,3,4,5}	1.12274	{1,2,3,4,5}	4.3292	{1,2,3,4,5}	2.48247	
		$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$
State 1 ( $k = 1$ )	$f_1(C_t^r(1))$	{1,2,3,4,5}	{2,3,4,5}	{2,3,4,5}	{2,3,4}	{2,4}
	$g(R_1, f_1(C_t^r(1)))$	1.1227	1.1227	1.1227	1.1227	1.1227
	$f_1(C_t^r(2))$	{2,3,4,5}	{1,2,3,5}	{1,2,3}	{1,2}	{1,3}
	$g(R_2, f_1(C_t^r(2)))$	7.1430	5.2558	6.4634	8.6724	6.4634
	$f_1(C_t^r(3))$	{2,3,4,5}	{2,3,5}	{2,3}	{2,5}	{5}
	$g(R_3, f_1(C_t^r(3)))$	3.2764	3.6028	5.4498	5.0960	6.3891
	$\sum_{j=1}^3 g(R_j, f_1(C_t^r(j)))$	11.5422	9.9813	13.0359	14.8910	13.9752
State 2 ( $k = 2$ )	$f_2(C_t^r(1))$	{2,3,4,5}	{2,3,5}	{2,3,4}	{2,4}	{4}
	$g(R_1, f_2(C_t^r(1)))$	1.1227	2.3030	1.1227	1.1227	1.1455
	$f_2(C_t^r(2))$	{1,2,3,4,5}	{1,2,3,4,5}	{1,2,3,5}	{1,2,3}	{1,2,3}
	$g(R_2, f_2(C_t^r(2)))$	4.3292	4.3292	5.2558	6.4634	6.4634
	$f_2(C_t^r(3))$	{2,3,4,5}	{2,3,5}	{2,3}	{2,5}	{5}
	$g(R_3, f_2(C_t^r(3)))$	3.2764	3.6028	5.4498	5.0960	6.3891
	$\sum_{j=1}^3 g(R_j, f_2(C_t^r(j)))$	8.7284	10.2350	11.8284	12.6821	13.9980
State 3 ( $k = 3$ )	$f_3(C_t^r(1))$	{2,3,4,5}	{2,3,5}	{2,3,4}	{2,4}	{4}
	$g(R_1, f_3(C_t^r(1)))$	1.1227	2.3030	1.1227	1.1227	1.1455
	$f_3(C_t^r(2))$	{2,3,4,5}	{1,2,3,5}	{1,2,3}	{1,2}	{1,3}
	$g(R_2, f_3(C_t^r(2)))$	7.1430	5.2558	6.4634	8.6724	6.4634
	$f_3(C_t^r(3))$	{1,2,3,4,5}	{2,3,4,5}	{2,3,5}	{2,3,5}	{2,5}
	$g(R_3, f_3(C_t^r(3)))$	2.4825	3.2764	3.6028	3.6028	5.0960
	$\sum_{j=1}^3 g(R_j, f_3(C_t^r(j)))$	10.7482	10.8353	11.1889	13.3979	12.7049

## 5.3 A Branch and Bound Approach for Resource Allocation

### 5.3.1 Bounding function

The bounding function is the most important component of the B&B algorithm. A weak bounding function is not able to reduce much complexity from the exhaustive-searching algorithm in finding the optimal channel assignment matrix. Since the proposed *DPRA* algorithm, as shown in Figs. 5.3 and 5.5, is very efficient in finding a near-optimal solution (i.e., channel assignment matrix), it can be used to obtain a very strong bounding function.

### 5.3.2 Branching rule

We need to build a searching tree that contains all possible channel assignments, to begin with. Since each VOC serves one data type only, the corresponding tree must have  $N$  levels and each parent node should have  $d$  child nodes to include all feasible solutions; see Fig. 5.7.

To start our searching process, we have to determine which channels will serve which data type initially. For the same reasons as those mentioned in the previous chapter, we shall assume that all VOCs are serving all data types initially. VOCs are deleted as we proceed along the tree and arrive at higher levels, i.e., at the  $k$ th level in the searching tree, we decide which data type is to be served by the  $m_k$ th VOC. Therefore, a decision sequence  $\{m_1, m_2, \dots, m_N\}$  has to be given before our tree-searching. In addition to two advantages discussed before, such an initialization assumption makes it more convenient for us to construct the searching tree for the B&B algorithm.

In using a B&B method to solve a minimized problem, an upper bound which close to the optimum solution is needed to block as much as possible the search of paths which does not lead to the optimal solution. If the power consumed by child nodes can be guaranteed to be equal or larger than its parent one, there can be no optimal

solution in the following child nodes when their parents' transmitted power is greater than the upper bound. The proposed innovative initialized method is consistent with this criterion and thus, it is very convenient to build the tree for the B&B algorithm with such an initialization.

As we assume that each data type can be served by all VOCs initially, i.e., it has all VOCs as the set of its *potential solution*,  $G$ , we have

$$A_{ij} = 1, 1 \leq i \leq N, 1 \leq j \leq d$$

With such an  $\mathbf{A}_{N \times d}$ , one then use the iterative algorithm (Fig. 4.1) for each data type to compute the virtual minimum consuming power  $g(x)$ . As each VOC is allowed to serve only one data type, at the  $i$ th level, every path from a parent node to the  $j$ th child node means the  $i$ th VOC will serve the  $j$ th data type. Note that the VOC subset  $C(j)$  of the child node  $j$  is a part of the parent node's VOC subset, the virtual minimum power of the child node must be equal or larger than that of the parent node which is proved in Lemma 4.2.2. Thus, if the virtual minimum power of the parent is not smaller than the bounding value, we are sure that there is no optimal solution in its child nodes.

### 5.3.3 Strategy for selecting next subproblem

Since the path of a channel assignment search has to arrive at a child node in the final level in order that the corresponding solution,  $f(x)$ , is feasible, the *depth first search (DFS)* strategy is suitable for this criterion. Initially, the bounding value is calculated by the bounding function derived from the *DPRA* approach. A DFS-based searching procedure then tries to continuously separate the parent space into the subproblem (child) space. If the virtual minimum consuming power of a child node is not smaller than the bounding value, there is no need to search the remaining subtree and one proceeds to search other candidate child nodes. If the searching reaches the final level and the feasible solution  $f(x)$  is less than the bounding value, the bounding value will be updated to  $f(x)$ . The complete searching tree of the B&B algorithm for solving



the problem of multi-rate wireless resource allocation is illustrated in Fig. 5.7. For convenience, this algorithm is called the **BBRA** algorithm henceforth.

### 5.3.4 Complexity reduction techniques for the B&B based algorithm

#### (1) The VOC order allocation of the tree's level

The order of the decision sequence  $\mathcal{D} = (m_1, m_2, \dots, m_N)$  that decides which data type is served by the  $m_k$ th VOC at the  $k$ th level is a prominent factor that affects the searching speed. If a bad VOC, which is not even used in the final optimal solution, is in the early part of the decision sequence, the virtual consuming power of the corresponding level's child nodes will not be altered and we will waste much more time searching in its subtree. Intuitively, the decision sequence  $\mathcal{D}$  should be arranged according to the VOC's GNR. But for the downlink scenario, the GNR of a VOC is a function of the data type (user terminal) it serves. Hence, we suggest the following systematic method to determine the decision sequence.

1. Apply the mono-rate channel/rate assignment algorithm for each data type.
2. Compute the sum rate of each VOC and denote by  $r_s(i)$  and  $D(i)$  the sum rate and the set of data types (as was determined by running the water-filling algorithm  $d$  times for the  $d$  data types) to be served by the  $i$ th VOC.
3.  $m_k = \arg \max\{r_s(j) | j \in I_N \setminus \{m_1, \dots, m_{k-1}\}\}$ , where  $I_N = \{1, 2, \dots, N\}$ .

Recall that the  $j$ th child node in the  $i$ th level, no matter to which parent node it belongs, represents the decision that the  $i$ th VOC is to serve the  $j$ th data type. Once the  $i$ th VOC is assigned to serve a data type, it must be released the duty of serving other data types in  $D(i)$ . These data types will have to seek the services of other VOCs that will demand larger power to satisfy the required transmission service. With such an ordering of the decision sequence, as we proceed from one level to the next, the

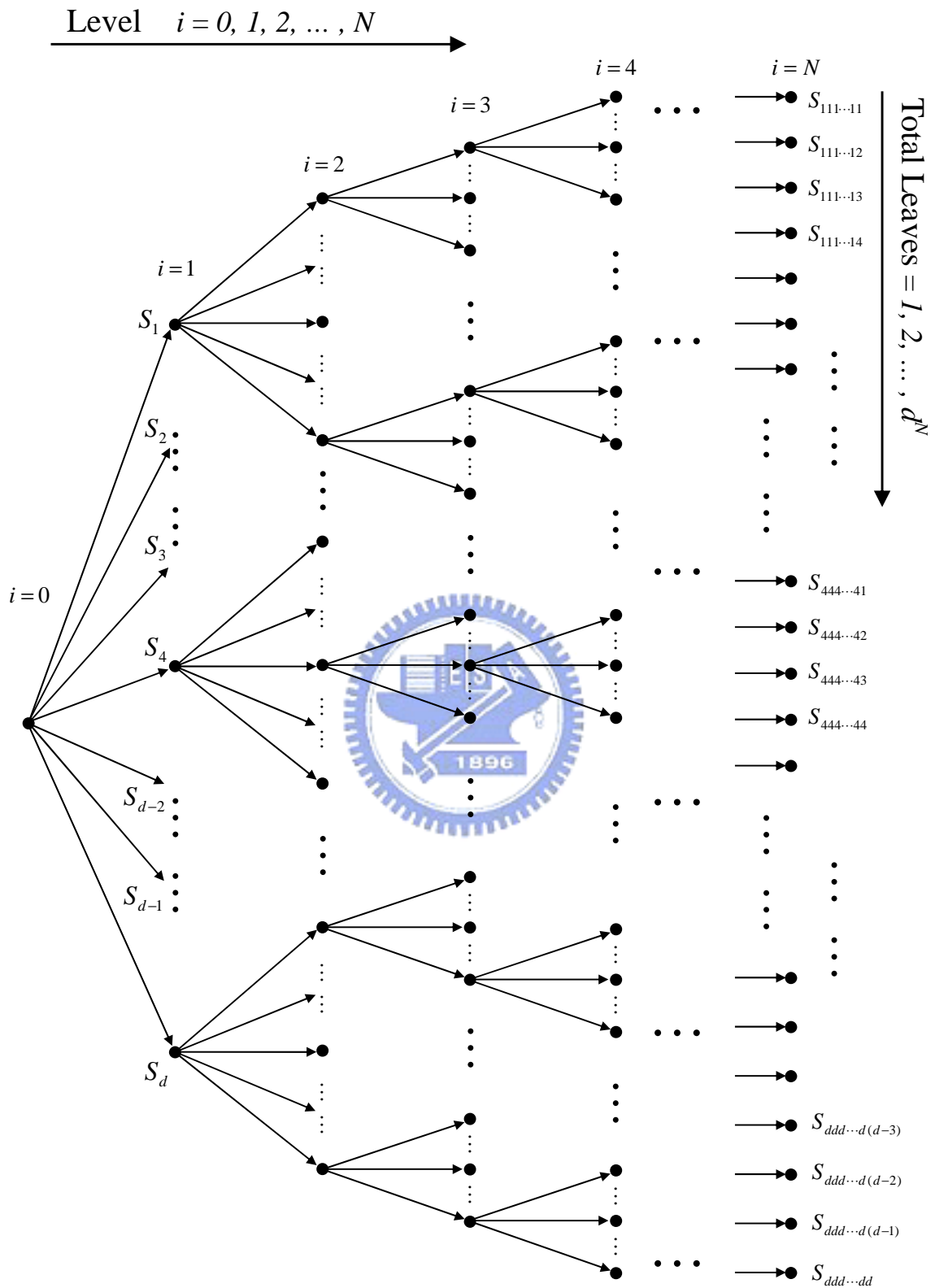


Figure 5.7: The B&B format compiled searching tree for multiuser channel assignment.

corresponding required total power will increase more rapidly, making it more likely to terminate an incorrect path. Thus the proposed decision sequence has the desired property of blocking the wrong path in the earliest possible stages.

Initially, we define some subsets of VOCs,

$$C(j) = \{i \mid 1 \leq i \leq N\}, 1 \leq j \leq d \quad (5.4)$$

$$C_{r \neq 0}(i) = \{j \mid 1 \leq j \leq d, r_{ij} \neq 0\}, 1 \leq i \leq N \quad (5.5)$$

$$C_{allocated} = \{\emptyset\} \quad (5.6)$$

$$C_{\geq 2} = \{i \mid 1 \leq i \leq N, |C_{r \neq 0}(i)| \geq 2, i \notin C_{allocated}\} \quad (5.7)$$

$$C_{< 2} = \{i \mid 1 \leq i \leq N, |C_{r \neq 0}(i)| < 2, i \notin C_{allocated}\} \quad (5.8)$$

For the VOCs in the subset ( $C_{allocated}$ ), it means that the order of the VOCs have been already determined in the tree's level. The VOCs in the subset ( $C_{\geq 2}$ ) are important VOCs in at least two data type subsets. For the reason that the more total data rate of the VOC supporting to all data types, the more power-raising after the VOC is decided to one data type, so we place the VOC with the largest summation serving data rate to the first level. After that, this VOC is virtually assigned to the data type which it supports the highest data rate, since this assignment increases the least power and has the highest probability to obtain the optimal path. Next, this VOC will be moved into the subset ( $C_{allocated}$ ). The data types which lose this specific VOC need to run the iterative algorithm to calculate a new power allocation and modify the subset ( $C_{\geq 2}$ ). Finally, continue the same action to arrange VOCs on the tree's level until the subset ( $C_{\geq 2}$ ) is empty.

Second, as for the VOCs in the subset ( $C_{< 2}$ ), we cannot make a clear difference from them, so a simple method is proposed based on exploring the importance of them to each data type. Concerning the VOCs in the subset ( $C_{allocated}$ ), the VOCs with the greatest channel gain in each subset ( $C(j)$ ) for  $1 \leq j \leq d$  are deleted and each data type runs the mono-rate iterative algorithm under its rate constraint again. The transmitted

rate will be transferred from the greatest VOCs to other VOCs, so there may be some new VOCs in the subset ( $C_{<2}$ ). It means these new VOCs are more important in all VOCs whose order have not been determined. Lastly, run the first step again until all VOCs are placed into the tree's level. However, if the subset ( $C_{\geq 2}$ ) is still empty, the rest VOCs will be sorted by their maximum channel response to all data types.

To describe the detailed procedure, the following parameters are needed to be defined.

$C(j)$	the subset of the $j$ th data type.
$C_{allocated}$	the subset of VOCs which has been allocated in the tree's level.
$C_{unallocated}$	the subset of VOCs which has not been allocated in the tree's level and the order of the subset is sorted by the maximum channel response in the VOC to all data types.
$MaxSumRateIndex$	the index of VOC that has the maximum summation rate of all data type.
$MaxRateIndex$	the index of data type which the $MaxSumRateIndex$ supports the highest transmitted rate.
$MaxVOCIndex(j)$	the index of VOC that serves the most transmitted rate to $j$ th data type.
$It\{ C(j) \}$	the consuming power of the iterative algorithm in Fig. 4.1 with the $j$ th data type subset.

With the above definitions, the complete procedure is described in the table given below,

Table 5.2: The procedure of allocating the order of tree's level.

**Step 1: Define**

$$C(j) = \{ i \mid 1 \leq i \leq N \}, 1 \leq j \leq d,$$

$$C_{r \neq 0}(i) = \{ j \mid 1 \leq j \leq d, r_{ij} \neq 0 \}, 1 \leq i \leq N,$$

$$C_{allocated} = \{\emptyset\}, C_{\geq 2} = \{ i \mid 1 \leq i \leq N, |C_{r \neq 0}(i)| \geq 2, i \notin C_{allocated} \}$$

**Step 2: while**  $|C_{\geq 2}| \neq 0$

$$MaxSumRateIndex = \{ i \mid \max_i(\sum_{j=1}^d r_{ij}), i \in C_{\geq 2} \}$$

$$MaxRateIndex = \{ j \mid \max_j(r_{ij}), i = MaxSumRateIndex, 1 \leq j \leq d \}$$

**for**  $j = 1 : d$

**if**  $j \neq MaxRateIndex$

$$C(j) = C(j) \setminus MaxSumRateIndex$$

*It*{  $C(j)$  }

**end**

**end**

$$C_{allocated} = C_{allocated} \cup MaxSumRateIndex$$

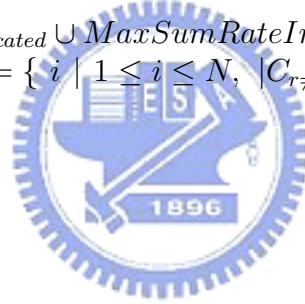
$$\text{Redefine } C_{\geq 2} = \{ i \mid 1 \leq i \leq N, |C_{r \neq 0}(i)| \geq 2, i \notin C_{allocated} \}$$

**end**

**if**  $|C_{allocated}| == N$

Exit Program

**end**



**Step 3: for**  $j = 1 : d$

$$MaxVOCIndex(j) = \{ i \mid \max_i(r_{ij}), i \in C_{allocated}, 1 \leq i \leq N \}$$

$$C(j) = C(j) \setminus MaxVOCIndex(j)$$

*It*{  $C(j)$  }

**end**

$$\text{Redefine } C_{\geq 2} = \{ i \mid 1 \leq i \leq N, |C_{r \neq 0}(i)| \geq 2, i \notin C_{allocated} \}$$

**if**  $|C_{\geq 2}| \neq 0$

go to **Step 2**

**end**

**Step 4: Define**  $C_{unallocated} = \{ i \mid \text{Sort}_i(\max_j(a_{ij})), i \notin C_{allocated}, 1 \leq j \leq d, 1 \leq i \leq N \}$

$$C_{allocated} = C_{allocated} \cup C_{unallocated}$$

## (2) Select the node with the least increasing power first

As we know B&B method before, the bounding value will affect the searching performance a lot, because the tight bounding value can stop the path which does not have the optimal solution earlier. Hence, the earlier a searching path can achieve the optimal solution, the less computational cost is needed. For this reason, there is a technique to increase the speed for searching the optimal solution.

Every parent node has  $d$  child nodes and can choose any one of them as the next node. However, if the worse node is chosen to be the next node, the consuming power at this level is increasing and the probability of this path having the optimal solution is decreasing. Thus, we sort the  $d$  child nodes according to their increasing consuming power and search forward from the node with the least raising power first. After applying this action to each level, the probability of finding the optimal solution in first several paths raises a lot.

## (3) Early termination in tree-searching

Due to our having already placed important VOCs in the front of the tree's level, there may be a lot of weak VOCs which are not used if the number of VOCs is large. Thus, we can apply a simple technique to reduce this useless calculation.

At the  $i$ th level, the  $i$ th VOC will be decided to transmit the  $j$ th data type. However, if the  $i$ th VOC does not support any rates for all data types, it means the  $i$ th VOC is weaker than other VOCs in each data type subset. In addition, most of the VOCs which will be determined after the  $i$ th level are weaker than the  $i$ th VOC, so we can start the checking procedure to verify if all VOCs decided after the  $i$ th level do not transmit any rates for all data types. If so, it implies those VOCs are weaker in each data type subset and the virtual consuming power will not increase even if those VOCs are all deleted from each data type subset. The expansion of the following tree from this parent node is useless and this parent node can be a feasible solution, so we can terminate the search

of this path at this parent node and check if its virtual minimum consuming power is smaller than the bounding value.

**(4) Reuse the results of repetitious calculations in the same level**

The resource allocation of the first level is taken as an example in Fig. 5.8 to illustrate this technique. There are  $d$  child nodes from the initial parent node and the first VOC needs to be decided which data type it should serve. For every child node, there is one subset which is the same as the initial node and the other subsets which exclude from the first VOC. For this reason, the transmitted power of each data type at the initial node is stored into the memory and each data type subset which loses the first VOC is also calculated and stored. Therefore, the total consuming power of the  $j$ th child node is summed up the transmitted power of the  $j$ th data type at the initial node and the other data types which exclude from the first VOC. In addition, if the first VOC does not transmit any rates for the  $j$ th data type, we can delete the first VOC from the  $j$ th data type subset directly and do not need to run the iterative algorithm.

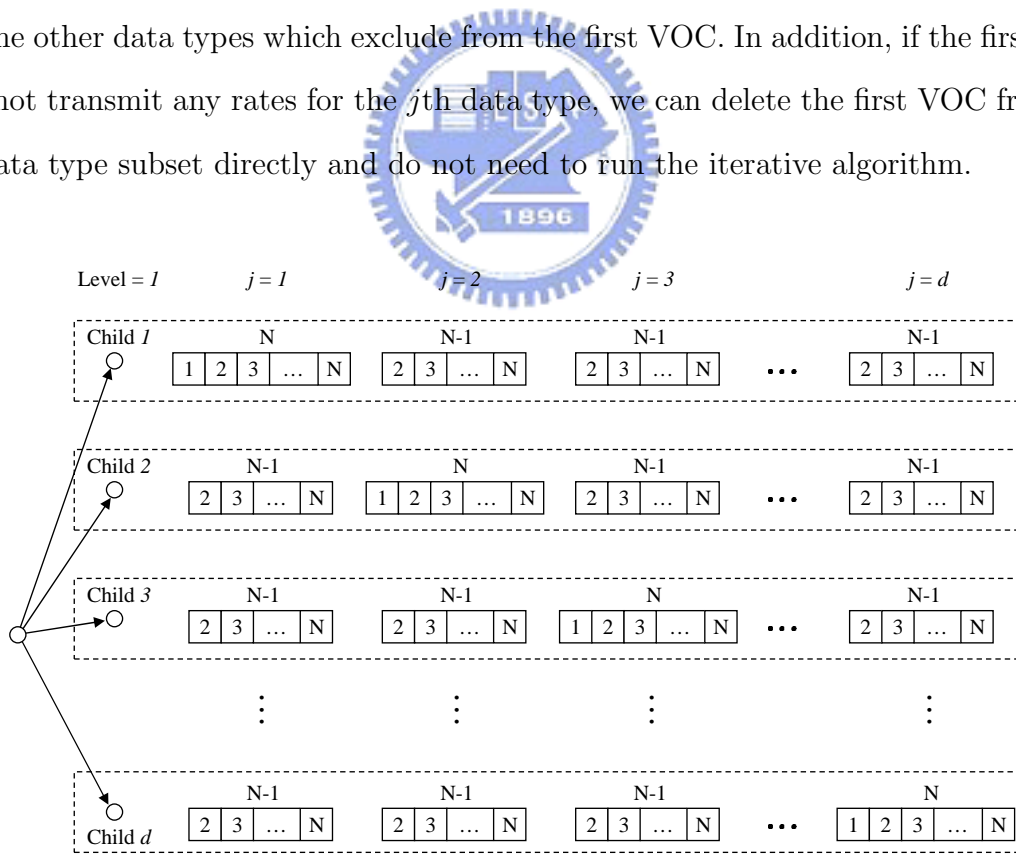
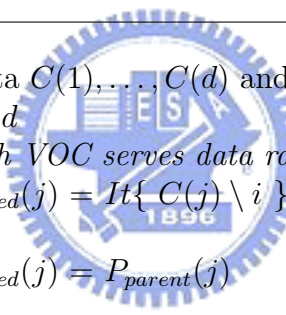


Figure 5.8: Illustration for subsets of all child nodes from the initial node

The following part is to define some parameters to express the detail procedure in Table 5.3.

$C(j)$	the subset of the $j$ th data type.
$P_{parent}(j)$	the virtual consuming power of the parent node's $j$ th data type subset.
$P_{deleted}(j)$	the virtual consuming power of the parent node's $j$ th data type subset excluding $i$ th VOC.
$P_{child}^k(j)$	the virtual consuming power of the $k$ th child node's $j$ th data type subset.
$It\{ C(j) \setminus i \}$	the consuming power of the iterative algorithm in Fig. 4.1 with the $j$ th data type subset excluding the $i$ th VOC.

Table 5.3: The procedure of reusing the results of repetitious calculations.



```

Step 1: Read all data  $C(1), \dots, C(d)$  and  $P_{parent}(1), \dots, P_{parent}(d)$ 
Step 2: for  $j = 1 : d$ 
    if the  $i$ th VOC serves data rate in  $C(j)$ 
         $P_{deleted}(j) = It\{ C(j) \setminus i \}$ 
    else
         $P_{deleted}(j) = P_{parent}(j)$ 
    end
end
Step 3: for  $k = 1 : d$ 
    for  $j = 1 : d$ 
        if  $k == j$ 
             $P_{child}^k(j) = P_{parent}(j)$ 
        else
             $P_{child}^k(j) = P_{deleted}(j)$ 
        end
    end
end
end

```

Lastly, the cost for this method is that it needs  $2 \times d \times N$  memory to store the value of the  $P_{parent}$  and  $P_{deleted}$ . However, calculational times of the iterative algorithm can decrease from  $d^2$  times to at most  $d$  times in each level.



## (5) Efficiencies of various complexity reduction techniques

To assess the improvements brought about by various complexity reduction techniques for the *BBRA* method, we resort to computer simulation and conduct 100000 runs for each technique to examine its average effect. The numbers of data types and VOCs are 5 and 128, and the normalized rate of each data type is uniformly distributed in  $[0, 3]$ . The results are summarized in Table 5.4 and the number in this table means the calling times of the mono-rate iterative algorithm. From our simulation, we know that the first technique affects the searching times of *BBRA* approach very much. Without this technique, the searching times are usually more than 1000000 times, so the first technique is always applied in Table 5.4 to compare the effects of other techniques.

The purpose of the second technique is to find the optimal solution as early as possible, so during the searching process, this method can provide a better bounding value to block useless paths at a earlier level. Hence, *BBRA* algorithm can avoid the tree expanding too much and reduce the probability of unusually high searching times. From the simulation results in Table 5.4, this method obviously works well.

For the third technique, the final part of useless level can be early terminated, so within most of large  $N$  cases, the early-terminating method can save a lot of useless computations. In the simulation results, we can know this method can reduce the average calculational times, but it is not able to avoid the deep searching when the bounding value is not close enough in some special cases.

The final technique uses the additional memory to save the repetitious calculations at each level, and in this way, the computational times can decrease much in all kinds of cases. From the results, this technique reduces much calculational complexity. However, it still cannot prevent the occurrence of the deep searching as the third technique, either.

Compared with the complexity associated with the VOC sorting, if all the above techniques are used in the *BBRA* scheme, the computational complexity is approximately reduced by 30 times. With these complexity-reduction techniques, the *BBRA* algorithm

becomes a practical algorithm when the number of required data types is not too large.

Table 5.4: Effects of different techniques on the performance of the *BBRA* approach.

d=5, N=128	DPRA	(1)	(1)+(2)	(1)+(3)	(1)+(4)	(1)+(2)+(3)+(4)
Mean	44.6147	2587.20	773.981	578.0975	116.1923	88.3233
Mean ( < 200000 )	44.6147	1717.80	773.981	549.9142	93.7858	88.3233
Prob. ( > 200000 )	0	0.0012	0	0.00005	0.00004	0
Max Times	81	21622894	180132	1800602	981053	587
Additional Cost	<del> </del>	*1	*1+*2	*1+*3	*1+*4	*1+*2+*3+*4

Cost for each technique :

- \*1. Additional calculation before starting *BBRA*.
- \*2. Additional  $N \times d$  memory to store the order of the next child node.
- \*3. Additional checking process to determine if the early termination is needed.
- \*4. Additional  $2 \times N \times d$  memory to store the  $P_{\text{parent}}$  and  $P_{\text{deleted}}$ .



**Example 5.2.** Consider the same data types and VOCs as Example 5.1 and take the result of it as an initial bounding value. First, Fig. 5.5 illustrates how the first technique works. For this example, on account of the number of VOCs being much fewer, the step 3 and step 4 of the first technique are not executed. For the same reason, the third technique is not, either. Finally, the searching tree and the detail computation of each node are shown in Figs. 5.9 and 5.6.

Table 5.5: A greedy search procedure for solving Example 5.2

Step1	Define $C(j) = \{1,2,3,4,5\}$ , $1 \leq j \leq 3$ , $C_{\geq 2} = \{1,3,4,5\}$ and $C_{allocated} = \{\emptyset\}$			
Step2	$C(1) = \{1,2,3,4,5\}$ $C(2) = \{1,2,3,4,5\}$ $C(3) = \{1,2,3,4,5\}$	$\begin{bmatrix} r_{1j} \\ r_{2j} \\ r_{3j} \\ r_{4j} \\ r_{5j} \end{bmatrix} = \begin{bmatrix} 0 & 1.5323 & 0.9196 \\ 0.1189 & 0 & 0 \\ 0 & 0.1460 & 0.4088 \\ 0.8811 & 0.8392 & 0.1575 \\ 0 & 0.4825 & 0.5141 \end{bmatrix}$	$\sum_{j=1}^d \begin{bmatrix} r_{1j} \\ r_{2j} \\ r_{3j} \\ r_{4j} \\ r_{5j} \end{bmatrix} = \begin{bmatrix} 2.4519 \\ 0.1189 \\ 0.5548 \\ 1.8777 \\ 0.9966 \end{bmatrix}$	MaxSumRateIndex = 1 MaxRateIndex = 2 $C_{allocated} = \{1\}$ $C_{\geq 2} = \{3,4,5\}$
Step2	$C(1) = \{2,3,4,5\}$ $C(2) = \{1,2,3,4,5\}$ $C(3) = \{2,3,4,5\}$	$\begin{bmatrix} r_{2j} \\ r_{3j} \\ r_{4j} \\ r_{5j} \end{bmatrix} = \begin{bmatrix} 0.1189 & 0 & 0 \\ 0 & 0.1460 & 0.7153 \\ 0.8811 & 0.8392 & 0.4640 \\ 0 & 0.4825 & 0.8207 \end{bmatrix}$	$\sum_{j=1}^d \begin{bmatrix} r_{2j} \\ r_{3j} \\ r_{4j} \\ r_{5j} \end{bmatrix} = \begin{bmatrix} 0.1189 \\ 0.8613 \\ 2.1842 \\ 1.3032 \end{bmatrix}$	MaxSumRateIndex = 4 MaxRateIndex = 1 $C_{allocated} = \{1,4\}$ $C_{\geq 2} = \{2,3,5\}$
Step2	$C(1) = \{2,3,4,5\}$ $C(2) = \{1,2,3,5\}$ $C(3) = \{2,3,5\}$	$\begin{bmatrix} r_{2j} \\ r_{3j} \\ r_{5j} \end{bmatrix} = \begin{bmatrix} 0.1189 & 0 & 0.0909 \\ 0 & 0.4257 & 0.9019 \\ 0 & 0.7622 & 1.0072 \end{bmatrix}$	$\sum_{j=1}^d \begin{bmatrix} r_{2j} \\ r_{3j} \\ r_{5j} \end{bmatrix} = \begin{bmatrix} 0.2099 \\ 1.3276 \\ 1.7694 \end{bmatrix}$	MaxSumRateIndex = 5 MaxRateIndex = 3 $C_{allocated} = \{1,4,5\}$ $C_{\geq 2} = \{2,3\}$
Step2	$C(1) = \{2,3,4\}$ $C(2) = \{1,2,3\}$ $C(3) = \{2,3,5\}$	$\begin{bmatrix} r_{2j} \\ r_{3j} \end{bmatrix} = \begin{bmatrix} 0.1189 & 0 & 0.0909 \\ 0 & 0.8069 & 0.9019 \end{bmatrix}$	$\sum_{j=1}^d \begin{bmatrix} r_{2j} \\ r_{3j} \end{bmatrix} = \begin{bmatrix} 0.2099 \\ 1.7087 \end{bmatrix}$	MaxSumRateIndex = 3 MaxRateIndex = 3 $C_{allocated} = \{1,4,5,3\}$ $C_{\geq 2} = \{2\}$
Step2	$C(1) = \{2,4\}$ $C(2) = \{1,2\}$ $C(3) = \{2,3,5\}$	$[r_{2j}] = [0.1189 \quad 0.3487 \quad 0.0909]$	$\sum_{j=1}^d [r_{2j}] = [0.5586]$	MaxSumRateIndex = 2 MaxRateIndex = 2 $C_{allocated} = \{1,4,5,3,2\}$ $C_{\geq 2} = \{\emptyset\}$

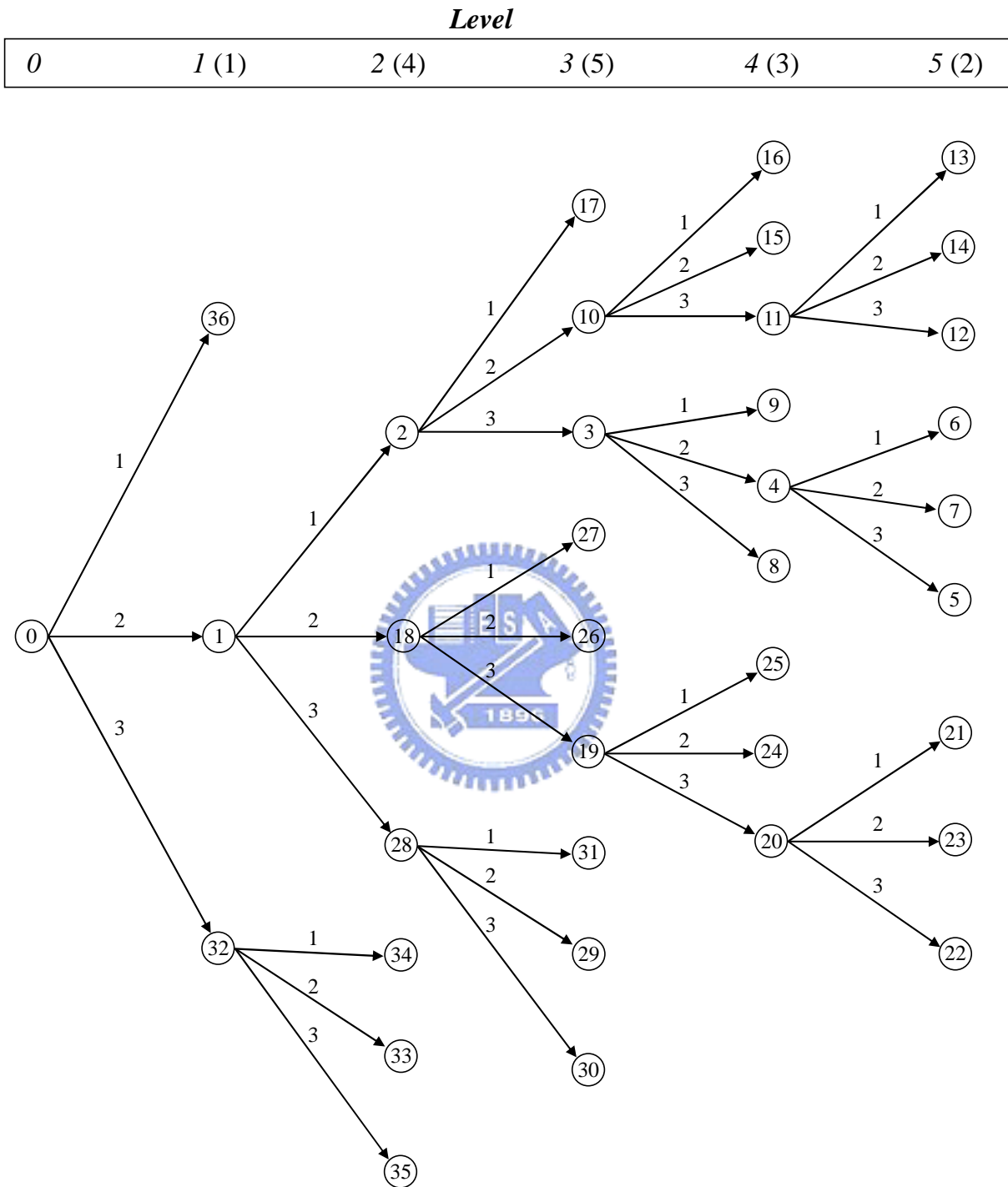


Figure 5.9: The searching tree associated with Example 5.2.

Table 5.6: The step-by-step searching procedure for solving Example 5.2.

Index	$C(I)$	Power of $C(I)$	$C(2)$	Power of $C(2)$	$C(3)$	Power of $C(3)$	Total Power	Upper Bound
0	{1,2,3,4,5}	1.1227	{1,2,3,4,5}	4.3292	{1,2,3,4,5}	2.4825	7.9344	12.7049
1	{2,3,4,5}	1.1227	{1,2,3,4,5}	4.3292	{2,3,4,5}	3.2764	8.7284	12.7049
2	{2,3,4,5}	1.1227	{1,2,3,5}	5.2558	{2,3,5}	3.6028	9.9813	12.7049
3	{2,3,4}	1.1227	{1,2,3}	6.4634	{2,3,5}	3.6028	11.1889	12.7049
4	{2,4}	1.1227	{1,2,3}	6.4634	{2,5}	5.0960	12.6821	12.7049
5	{4}	1.1455	{1,3}	6.4634	{2,5}	5.0960	12.7049	12.7049
6	{2,4}	1.1227	{1,3}	6.4634	{5}	6.3891	13.9752	12.7049
7	{4}	1.1455	{1,2,3}	6.4634	{5}	6.3891	13.9980	12.7049
8	{2,4}	1.1227	{1,2}	8.6724	{2,3,5}	3.6028	13.3979	12.7049
9	{2,3,4}	1.1227	{1,2}	8.6724	{2,5}	5.0960	14.8910	12.7049
10	{2,3,4}	1.1227	{1,2,3,5}	5.2558	{2,3}	5.4498	11.8284	12.7049
11	{2,4}	1.1227	{1,2,5}	5.6469	{2,3}	5.4498	12.2194	12.7049
12	{4}	1.1455	{1,5}	5.6469	{2,3}	5.4498	12.2422	12.7049
13	{2,4}	1.1227	{1,5}	5.6469	{3}	7.0990	13.8686	12.2422
14	{4}	1.1455	{1,2,5}	5.6469	{3}	7.0990	13.8913	12.2422
15	{2,4}	1.1227	{1,2,3,5}	5.2558	{2}	15.9726	22.3512	12.2422
16	{2,3,4}	1.1227	{1,2,5}	5.6469	{2}	15.9726	22.7422	12.2422
17	{2,3,4,5}	1.1227	{1,2,3}	6.4634	{2,3}	5.4498	13.0359	12.2422
18	{2,3,5}	2.3030	{1,2,3,4,5}	4.3292	{2,3,5}	3.6028	10.2350	12.2422
19	{2,3}	2.3030	{1,2,3,4}	4.6549	{2,3,5}	3.6028	10.5607	12.2422
20	{2}	2.4547	{1,2,4}	4.8381	{2,3,5}	3.6028	10.8956	12.2422
21	{2}	2.4547	{1,4}	4.8381	{3,5}	3.6195	10.9123	12.2422
22	{}	Inf	{1,4}	4.8381	{2,3,5}	3.6028	Inf	10.9123
23	{}	Inf	{1,2,4}	4.8381	{3,5}	3.6195	Inf	10.9123
24	{2}	2.4547	{1,2,3,4}	4.6549	{2,5}	5.0960	12.2055	10.9123
25	{2,3}	2.3030	{1,2,4}	4.8381	{2,5}	5.0960	12.2370	10.9123
26	{2,3}	2.3030	{1,2,3,4,5}	4.3292	{2,3}	5.4498	12.0820	10.9123
27	{2,3,5}	2.3030	{1,2,3,4}	4.6549	{2,3}	5.4498	12.4077	10.9123
28	{2,3,5}	2.3030	{1,2,3,5}	5.2558	{2,3,4,5}	3.2764	10.8353	10.9123
29	{2,3}	2.3030	{1,2,3,5}	5.2558	{2,3,4}	4.2112	11.7700	10.9123
30	{2,3}	2.3030	{1,2,3}	6.4634	{2,3,4,5}	3.2764	12.0428	10.9123
31	{2,3,5}	2.3030	{1,2,3}	6.4634	{2,3,4}	4.2112	12.9776	10.9123
32	{2,3,4,5}	1.1227	{2,3,4,5}	7.1430	{1,2,3,4,5}	2.4825	10.7482	10.9123
33	{2,3,5}	2.3030	{2,3,4,5}	7.1430	{1,2,3,5}	2.5092	11.9552	10.9123
34	{2,3,4,5}	1.1227	{2,3,5}	11.3586	{1,2,3,5}	2.5092	14.9905	10.9123
35	{2,3,5}	2.3030	{2,3,5}	11.3586	{1,2,3,4,5}	2.4825	16.1440	10.9123
36	{1,2,3,4,5}	1.1227	{2,3,4,5}	7.1430	{2,3,4,5}	3.2764	11.5422	10.9123

## 5.4 Complexity Consideration

For the conventional exhaustive-searching optimization algorithm, it has to try all possible channel allocations and runs the iterative optimization algorithm in Fig. 4.1 in each channel allocation. If there are  $N$  VOCs and  $d$  requested data types, the exhaustive-searching optimization algorithm will have  $d^N$  possible channel allocations. In addition, in each channel allocation, it has to run the iterative optimization algorithm once for each data type. Thus, the conventional exhaustive-searching optimization algorithm is an  $O(d \times d^N)$  computational procedure.

The computational complexity of the *DPRA* approach is a function of the number of stages and states. For the *DPRA* algorithm, there are  $N$  stages and  $d$  states in each stage. Since the  $i$ th VOC will be excluded from each data type subset at the  $i$ th stage, we can use the idea discussed in the fourth complexity-reduction technique of the *BBRA* scheme to reduce the repetitious calculations in each state at the  $i$ th stage. For this reason, *DPRA* method requires to run the iterative optimization algorithm for at most  $d$  times at each stage and  $d \times N$  times towards the whole algorithm, so  $O(d \times N)$  is the upper bound of its computational procedure.

The complexity of the *BBRA* method is hard to estimate directly. It is highly dependent on the bounding value and techniques. We use computer simulations to estimate the approximate complexity.

Fig. 4.1 compares the complexities of *DPRA* and *BBRA* schemes with the computational complexity evaluated by their processing times. The normalized noise power level  $\sigma^2$  is assumed to be  $0.01$  and different numbers of data types with the same normalized sum rate are compared.

For the downlink scenario, Figs. 5.10 and 5.11 indicate that the complexity of the *DPRA* scheme increases with the number of VOCs, as has been expected. The complexity, however, is much lower than the upper bound. Besides, these results reveal an interesting fact that the average complexity of the case with 64 VOCs is higher than

that of the case with 128 VOCs. This is because there are much more VOCs with high GNRs in the 128-VOC case. A better VOC can support larger data rate, so less VOCs are needed.

Figs. 5.10 and 5.11 indicate that the complexities of both *DPRA* and *BBRA* schemes increase with the number of data types. For the *DPRA* algorithm, the degradation with respect to the optimal performance is an increasing function of the number of data types as is shown in Fig. 5.3. Using the *DPRA* solution as its bounding function, the complexity of *BBRA* algorithm thus increases very rapidly when the number of data types increases.

With regard to the uplink scenario, the complexities of *DPRA* and *BBRA* schemes in Figs. 5.12 and 5.13 increase with the number of data types as the downlink scenario. The major difference is that for the uplink scenario, the complexity of *BBRA* approach with more VOCs is higher than it with fewer VOCs. The cause of it is that the GNR of the VOC is the same to each data type. When there are more important VOCs in each data type subset, the more compare is needed to be calculated.

From the results, *DPRA* method has much reduced the computational complexity compared to the full-searching algorithm and guarantees that the complexity is not over the upper bound. Thus, *DPRA* algorithm is very suitable to be a practical algorithm for its low computational and hardware requirements. Moreover, *BBRA* approach also shows an acceptable computational complexity to find the optimal solution within less normalized sum rate.

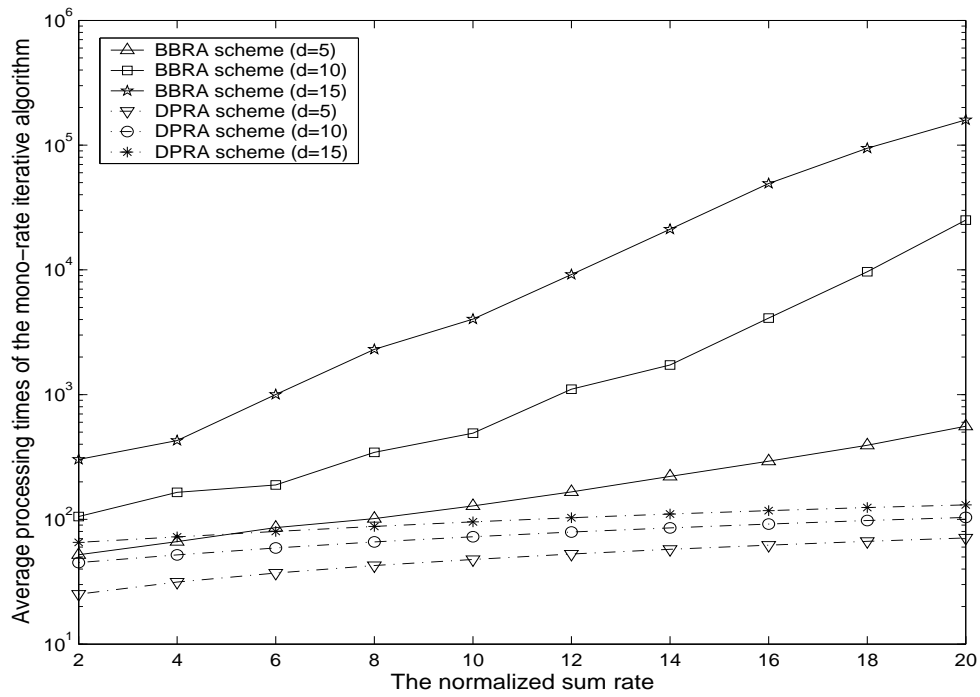


Figure 5.10: Average complexities of *BBRA* and *DPRA* schemes in a 64-VOCs downlink scenario.

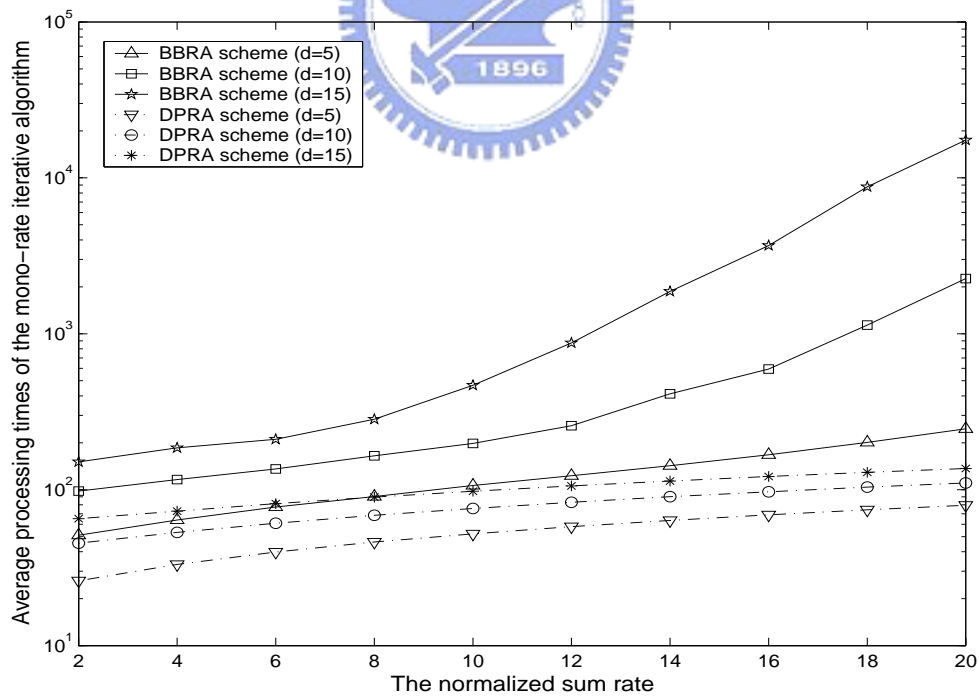


Figure 5.11: Average complexities of *BBRA* and *DPRA* schemes in a 128-VOCs downlink scenario.



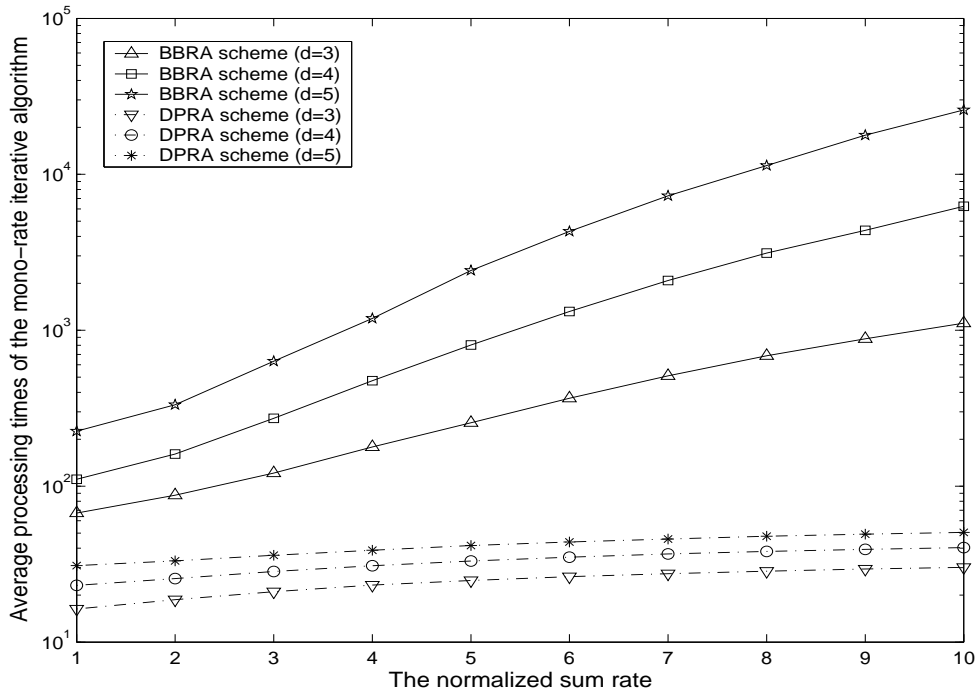


Figure 5.12: Average complexities of *BBRA* and *DPRA* schemes in a 10-VOCs uplink scenario.

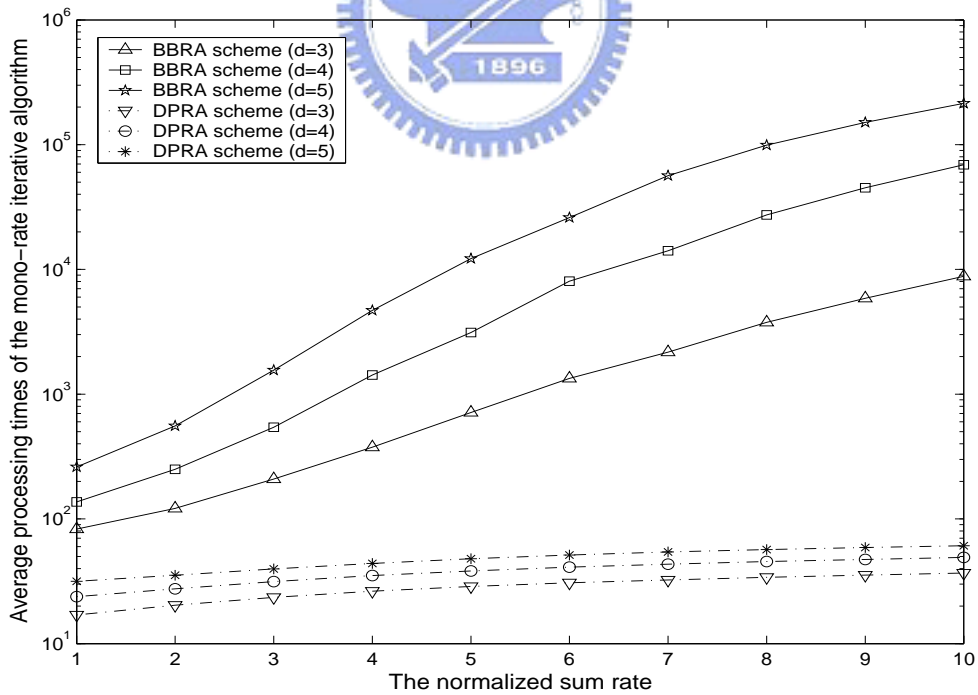



Figure 5.13: Average complexities of *BBRA* and *DPRA* schemes in a 15-VOCs uplink scenario.

# Chapter 6

## Simulation Results

In this chapter we revisit the two application scenarios discussed in Chapter 2 and examine the numerical performance of our algorithms when applied to solve the radio resource allocation problems arisen in these two operation scenarios. To demonstrate the usefulness of the proposed algorithms and see how they perform with realistic QoS constraints, we consider four distinct services whose rate requirements are given by

Table 6.1: Transmission rate requirements for video, audio, voice and data services



Service	Data rate
Video	128 kbps
Audio	56 kbps
Voice	9.6 kbps
Data	No specific

Two independent multimedia sources whose respective probabilities of generating different services are listed in the following table are assumed in our simulation.

Table 6.2: Statistical characterizations of two independent multimedia sources.

	Video	Audio	Voice	Data
Source 1	0.25	0.25	0.25	0.25
Source 2	0.125	0.125	0.5	0.25

## 6.1 Resource Allocation for an OFDMA Downlink System

The first application example we consider is an OFDMA system which has  $N = 64$  or 128 VOCs. A similar system can be found in IEEE 802.16e. The normalized required rates for video, audio and voice are calculated by dividing the required data rates of Table 6.1 by the sub-carrier frequency spacing. As for the data service, since there is no specific QoS constraint, the normalized rate for data is assumed to be uniformly distributed in  $[0, 5]$ . The system parameters used in simulation are given in Table 6.3 below. We first examine the performance of *DPRA* approach which gives suboptimal

Table 6.3: Simulation parameters of the OFDMA system

Sub-carrier frequency spacing ( $W$ )	10.94 kHz
Number of sub-carriers ( $N$ )	64, 128
Number of data types ( $d$ )	3, 5, 8, 10
Noise power level ( $\sigma^2$ )	0.01
Normalized required rate for video	0.872
Normalized required rate for audio	5.09
Normalized required rate for voice	11.64
Normalized required rate for data	0~5

performance. Since the probabilities of generating video and audio services in Source 1 are higher than those of Source 2, the expected normalized sum rate for Source 1 should be higher than that of Source 2. Thus, in Figs. 6.1 and 6.2, the probability of achieving the optimum allocation and the performance loss for Source 1 are inferior to those of Source 2.

Figs. 5.2 and 5.3 indicate that the performance loss increases with the number of data types or the normalized sum rate. In this case, the normalized sum rate is proportional to the number of data types, so the performance loss in Fig. 6.2 degrade much fast than that in Fig. 5.3, when the number of data types increases. Note that the performance

loss of the *DPRA* method is very low: even if the number of data types is as large as 10, the performance loss is still maintained to within 1%.

On the other hand, Fig. 6.3 shows that the *BBRA* approach requires very high computational complexity when the number of data types becomes larger than 5. By contrast, the complexity of the *DPRA* scheme increases very slowly; it is still reasonably affordable even when the number of data types is 10. We conclude that for the downlink OFDMA system, the *DPRA* algorithm provides a simple and efficient solution that offer near optimum solution ( $< 1\%$ ) with very little complexity.

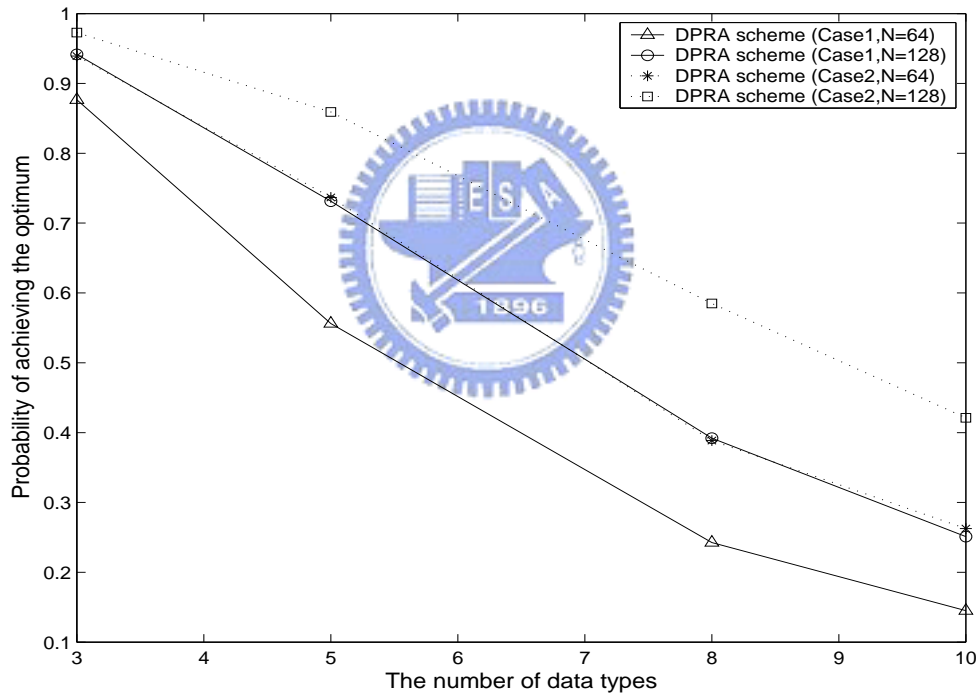


Figure 6.1: The probability that the *DPRA* method yields the optimum performance in an OFDMA downlink system.

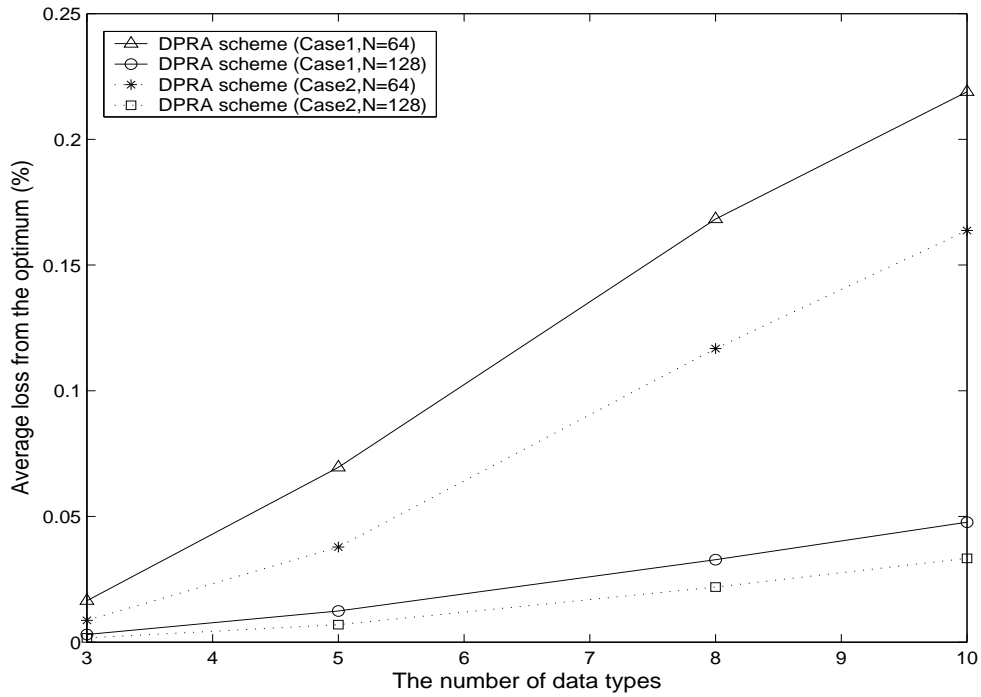


Figure 6.2: The average performance degradation (with respect to the optimum performance) of the *DPR*A algorithm in an OFDMA downlink system.

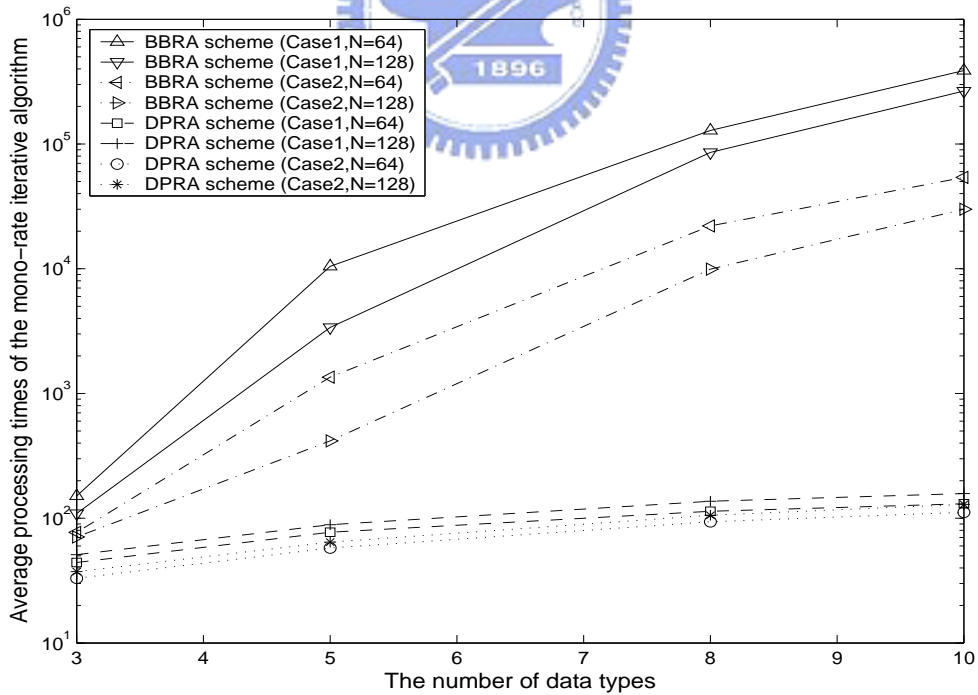
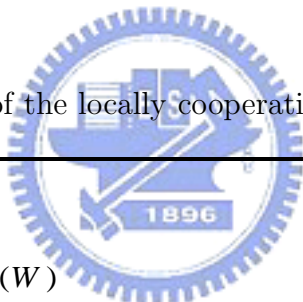


Figure 6.3: The average complexity of the *BBRA* and *DPR*A schemes when used in an OFDMA downlink system.

## 6.2 Resource Allocation for a Locally Cooperative Uplink System

A locally cooperative uplink communication scenario in which the user terminals are located within a small neighborhood is considered. We assume that all uplink users possess a dual-mode capability with the same local radio interface. The local terminals might be WLAN/WCDMA or WLAN/GPRS mobile stations. We also assume that the inter-user distance is far smaller than the uplink (terminal-to-base-station) distance, and the inter-user link has a much higher capacity and QoS than the uplink. Hence, the latency, transmitted errors and power consumed within the local area network are negligibly small. The normalized required rates for video, audio, voice and data are the same as the downlink scenario discussed before. Table 6.4 lists the simulation parameters used for such a system.

Table 6.4: System parameters of the locally cooperative uplink system under consideration.



Chip rate	3.84 Mcps
Spreading factor	128
Channel bit rate ( $W$ )	30 kbps
Number of cooperative VOCs ( $N$ )	10, 15
Number of data types ( $d$ )	3, 4, 5
Noise power level ( $\sigma^2$ )	0.01
Normalized required rate for video	0.32
Normalized required rate for audio	1.87
Normalized required rate for voice	4.27
Normalized required rate for data	0~5

As the mean normalized sum rate for Source 1 is higher than that of Source 2, the performance of Source 1 is worse than that of Source 2 which is evident from Figs. 6.4 and 6.5. As mentioned before, good (high GNR) VOCs are important to all data types, the probability of the wrong decision at each state increases a lot. Thus, the perfor-

mance loss among the downlink scenario is obviously much superior to it among the uplink scenario. Moreover, the complexity of *BBRA* approach for the uplink scenario increases much because of worse initial bounding value.

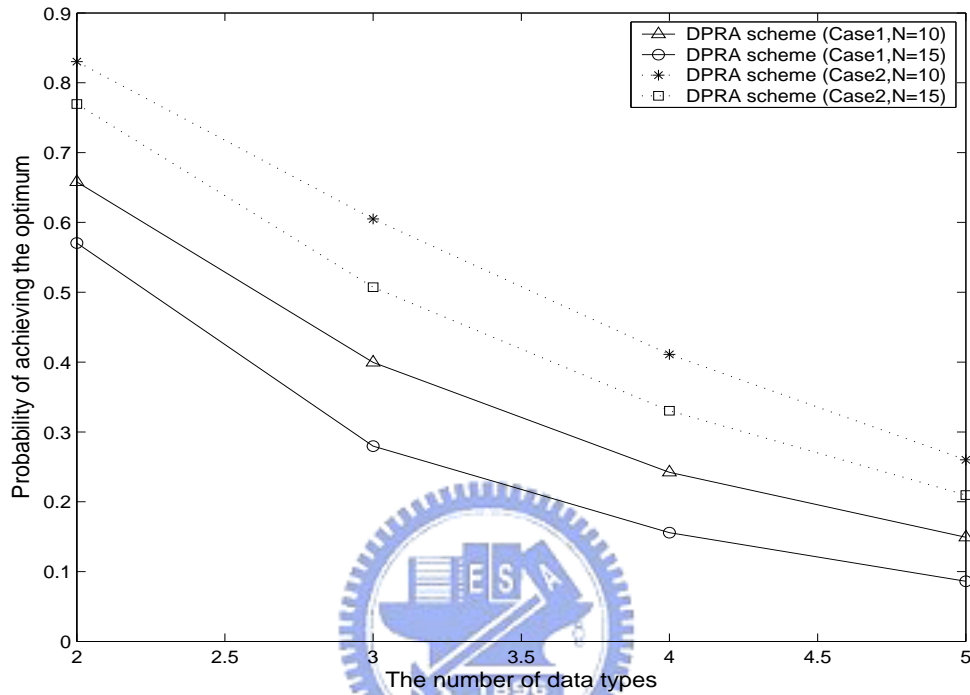


Figure 6.4: The probability of the *DPRA* method achieving the optimum in a locally cooperative uplink scenario

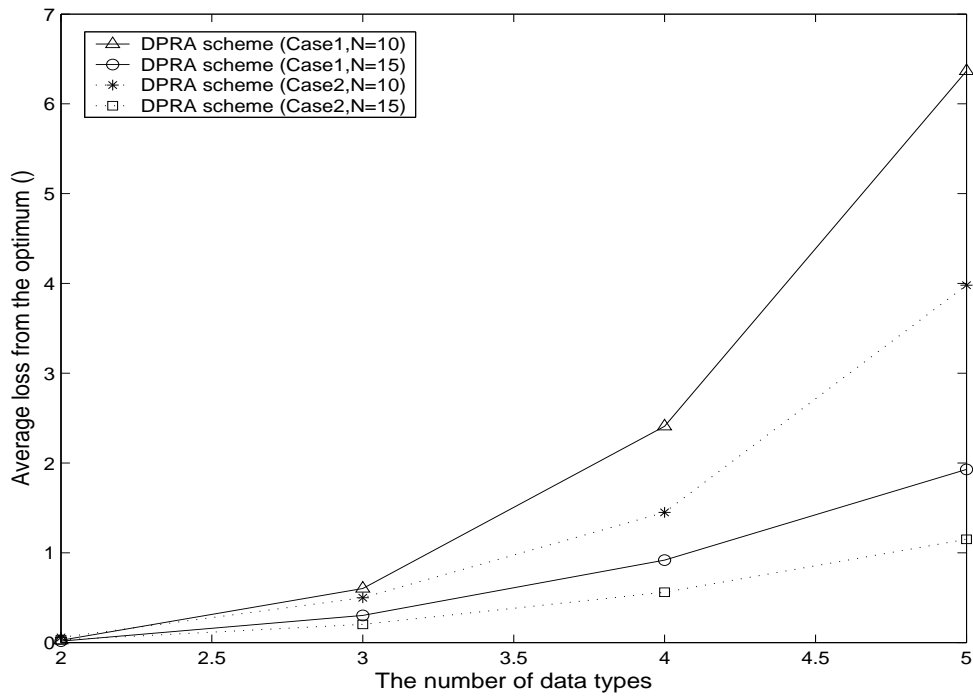


Figure 6.5: The average performance loss from the optimum by the *DPRA* algorithm in a locally cooperative uplink scenario

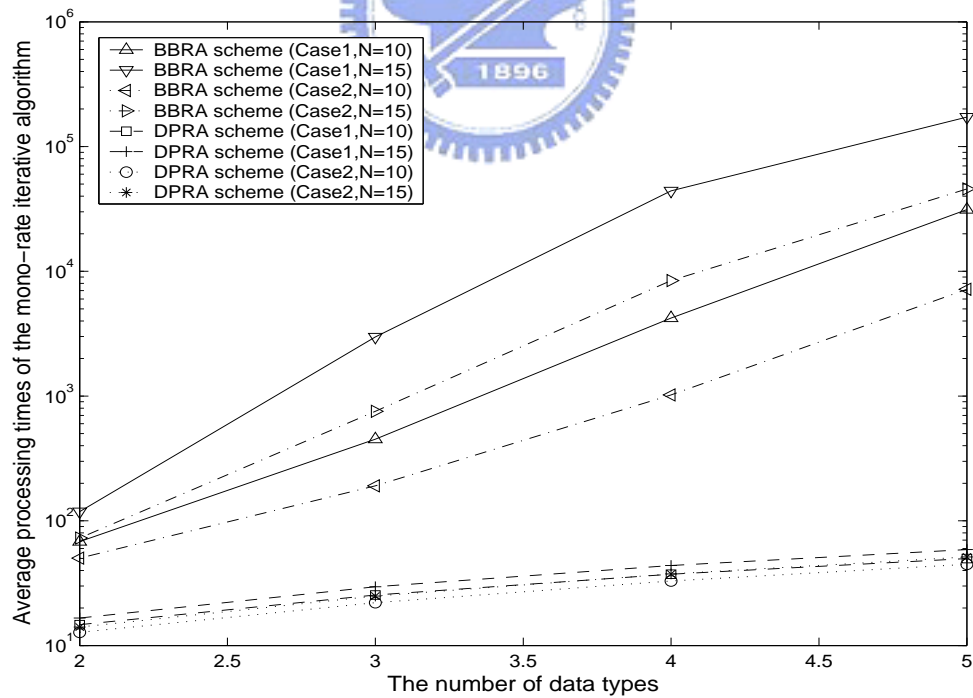


Figure 6.6: The average complexity of the *BBRA* and *DPRA* schemes in a locally cooperative uplink scenario



# Chapter 7

## Conclusion

We have presented efficient solutions for constrained optimization problems arisen in radio resource allocation in wireless multiuser multimedia communication networks. The applications of our algorithms, however, go far beyond the original problems.

For many cases, the best radio resource allocation strategy is a water-filling-like solution and the search of such an optimal strategy is accomplished by a greedy approach, which is not only time-consuming but also takes large memory space. The computational complexities for existing suboptimal algorithms are still relatively high.

We first present an efficient iterative algorithm based on an analytical closed form to solve the optimal resource allocation for the mono-rate case. Using the mono-rate solution, we then extend our investigation to multi-rate cases. We suggest efficient procedures for obtaining optimal and near-optimal solutions. A DP-based algorithm called *DPRA* algorithm is proposed to obtain near-optimal solution. This algorithm has much reduced complexity and suffers only minor performance degradation within the range of interest. Optimal multi-rate solution is obtained by a B&B-based approach called *BBRA* approach which incurs only minor complexity increase from *DPRA* method but guarantees zero performance loss. Finally, we provide two application examples to validate our claims on the performance and effectiveness of both *DPRA* and *BBRA* schemes.

There are many issues remain to explored and require further investigations. For

example, it would be interesting to look into the scenario when the channels are not orthogonal and inter-channel interference has to be taken into account. The fairness issue is not touched upon in this thesis, a possible candidate objective function that incorporates the concept of fairness is to maximize product rate instead of sum rate. The proposed methods can also be applied to problems in scheduling and admission/congestion control. Finally, a static channel condition is assumed in our work, a worthy extension would be the assessment on the impact of channels' time-varying nature.



# Bibliography

- [1] A. J. Goldsmith and P. P. Varaiya, "Capacity of fading channels with channel side information," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1986-1992, Nov. 1997.
- [2] R. Knopp and P. A. Humblet, "Information capacity and power control in single-cell multiuser communications," in *Proc. IEEE Int. Communications Conf. (ICC'95)*, Seattle, WA, June 1995.
- [3] D. Tse, and S. V. Hanly, "Multiaccess Fading Channels-Part I: Polymatroid Structure, Optimal Resource Allocation and Throughput Capacities," *IEEE Trans. Inform. Theory*, vol.44, pp. 2796-2815, Nov. 1998.
- [4] D. Tse and S. V. Hanly, "Multi-Access Fading Channels: Part II: Delay-Limited Capacities" *IEEE Trans. Inform. Theory*, vol.44, pp. 2816-2831, Nov. 1998.
- [5] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity, Part I: system description," *IEEE Trans. Commun.*, vol.51, pp. 1927-1938, Nov. 2003.
- [6] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity, Part II: implementation aspects and performance analysis," *IEEE Trans. Commun.*, vol.51, pp. 1939-1948, Nov. 2003.
- [7] J. Laneman, D. Tse, and G. Wornell, "Cooperative diversity in wireless networks: Efficient protocols and outage behavior," *IEEE Trans. Inform. Theory*, vol.50, pp. 3062-3080, Dec. 2004.

- [8] G. Kramer, M. Gastpar, and P. Gupta, "Cooperative strategies and capacity theorems for relay networks," *IEEE Trans. Inform. Theory*, vol.51, pp. 3037-3063, Sep. 2005.
- [9] I. Maric, and R. D. Yates, "Cooperative broadcast for maximum network lifetime," *IEEE J. Sel. Area Commun.*, vol 23, pp. 127-135, Jan. 2005.
- [10] Z. Han, T. Himsoon, W. P. Siriwongpairat, and K. J. R. Liu, "Energy-efficient cooperative transmission over multiuser OFDM networks: who helps whom and how to cooperate," *IEEE Wireless Commun. Networking Conf. (WCNC)*, vol. 2, pp. 1030-1035, Mar. 2005.
- [11] W. Yu, W. Rhee, S. Boyd, and J. Cioffi, "Iterative water-filling for Gaussian vector multiple-access channels," *IEEE Trans. Inform. Theory*, vol. 50, no. 1, pp. 145V152, Jan. 2004.
- [12] N. Jindal, W. Rhee, S. Vishwanath, S. A. Jafar, and A. Goldsmith, "Sum power iterative waterfilling for multi-antenna Gaussian broadcast channels," *IEEE Trans. Inform. Theory*, vol. 51, pp. 1570V1580, Apr. 2005.
- [13] W. Yu, "Sum-capacity computation for the Gaussian vector broadcast channel via dual decomposition," *IEEE Trans. Inform. Theory*, vol. 52, no. 2, pp. 754V759, Feb. 2006.
- [14] R. Bellman, *Dynamic Programming*. Princeton, N.J: Princeton University Press, 1957.
- [15] C. N. Chuah, D. Tse, J. Kahn, and R. A. Valenzuela, "Capacity scaling in MIMO wireless systems under correlated fading," *IEEE Trans. Inform. Theory*, vol.48, pp. 637-650, Mar. 2002.

# 作 者 簡 歷

邱泰翔，台北市北投區人，1982 年生

台北市立建國高級中學	1998.9 ~ 2001.6
國立交通大學電信工程學系	2001.9 ~ 2005.6
國立交通大學電信工程學系研究所系統組	2005.9 ~ 2007.6

## Graduate Course :

1. Digital Signal Processing
2. Digital Communications
3. Coding Theory
4. Advanced Coding Theory
5. Detection and Estimation Theory
6. Data Compression
7. Wireless Communications
8. Embedded System Design