國立臺灣大學管理學院資訊管理學研究所

碩士論文

Department of Graduate Institute of Management
College of Information Management
National Taiwan University

Master Thesis

# 考量訊號延遲以及傳輸量下無線通訊網路針對多種流量類型之近似最佳化時槽分配演算法

# A Near-Optimal Time Slot Allocation Algorithm for Wireless Communication Networks under Throughput and Delay Constraints for Multiple Classes of Traffic

林岦毅

Li-Yi Lin

指導教授：林永松 博士

Advisor: Yeong-Song Lin, Ph.D.

中華民國 96 年 7 月

July, 2007

# 國立臺灣大學碩士學位論文
# 口試委員會審定書

## 考量訊號延遲以及傳輸量下無線通訊網路針對多種流量類型之近似最佳化時槽分配演算法

本論文係林岦毅君（學號 R94725041）在國立臺灣大學資訊管理學系、所完成之碩士學位論文，於民國 96 年 7 月 19 日承下列考試委員審查通過及口試及格，特此證明

口試委員：林永松

[illegible]　顏宏旭

孫雅麗　祝國忠

所　　長：孫雅麗

# 謝 詞

很快地兩年的研究生涯就要結束了，很高興當初能夠成為臺大資管所的一份子，在這期間我學習到了許多新的知識與能力，這一趟的學習過程真的讓我感到不虛此行。

首先我要感謝我的恩師 林永松老師，在這兩年的研究生涯中，學生遇到許多困難，老師總是能夠一一引導我往正確的方向前進，而老師嚴謹的研究態度與治學風格更是讓學生一生受用不盡。此外也感謝孫雅麗所長、祝國忠老師、顏宏旭老師、以及呂俊賢老師在口試時對於學生的論文提出許多建議及指正，實在受惠良多。

感謝博士班的演福學長在我踏入無線網路研究領域時，能夠給予我許多協助與指導。感謝政達學長能夠教導我有關馬可夫決策過程的相關知識。也謝謝柏皓學長在簡報技巧上面給予我不少的建議以及對於碩士班的我們都能夠有所照顧。也很高興能夠與翊恆、俊維、承賓、坤道、雅芳、怡孜成為網路最佳化實驗室的同學，讓我的研究生涯充滿了歡笑，也能夠互相分享研究的心得。另外也謝謝研一的學弟們：奐廷、至浩、孜謙、政祐、志元，謝謝你們在口試期間的幫忙，讓我能夠專心準備口試。

此外，也要感謝政大吉他社四十三屆的朋友們以及昇呈、彥廷、浚杰，在我研究生涯的過程中，總是能夠適時地給予我歡樂，讓我能夠再度打起精神投入研究的工作中。

最後，我要感謝我的父母：林傳茂先生與許秀鑾女士，謝謝你們總是給予我最大的支持，尤其感謝我的母親，一個人辛苦把我帶大，讓我在求學的過程中總是能無後顧之憂，專心的在學業上努力。

林豈毅　謹識
于臺大資訊管理研究所
民國 九十六年 七月

# 論文摘要

論文題目：考量訊號延遲以及傳輸量下無線通訊網路針對多種流量類型之近似最佳化時槽分配演算法

作　　者：林岦毅

指導教授：林永松　博士


無線網路能夠帶給使用者更多的方便性，但由於傳輸特性的限制，每位使用者所能分配到的頻寬也有限；在多媒體傳輸的服務需求增加之下，對於資料傳輸的服務品質(Quality of Service)的要求也更為嚴格。對於網際網路提供業者而言，如何在無線網路有限的頻寬資源之下，滿足各種等級的服務品質要求，並且使得網際網路提供業者的收益能夠達到最大化，這是一個相當值得研究的議題。

我們將上述的問題透過馬可夫決策過程並結合拉格蘭日鬆弛法來解決馬可夫決策過程加上額外的服務品質的要求問題。藉由以上所提出的方法，我們預期可以得到一個針對不同系統狀態下的最佳時槽分配策略，能夠在滿足系統服務品質要求之下，達到系統收益最大化的目的。



**關鍵詞：無線網路、時槽分配、最佳化、訊號延遲、傳輸量、馬可夫決策過程、拉格蘭日鬆弛法。**

# THESIS ABSTRACT

GRADUATE INSTITUTE OF INFORMATION MANAGEMENT

NATIONAL TAIWAN UNIVERSITY

NAME: LI-YI LIN

ADVISER: YEONG-SUNG LIN

## A Near-Optimal Time Slot Allocation Algorithm for Wireless Communication Networks under Throughput and Delay Constraints for Multiple Classes of Traffic

Wireless communication networks provide convenience, however, also challenges to multimedia services due to typically limited bandwidth and various QoS (Quality-of-Service) requirements. For a wireless communication network service provider/administrator, it is then essential to develop an effective resource allocation policy so as to fully satisfy possibly different QoS requirements by different classes of traffic, while in the meantime, for example, the overall long-term system revenue rate can be maximized.

In this thesis, the problem of time slot allocation in wireless communication networks under throughput and delay constraints for multiple classes of traffic is considered. The basic approach to the algorithm development is a novel combination of

MDP (Markovian Decision Process) and Lagrangean relaxation. The problem is first formulated as a standard linear-programming form of an MDP problem, however, with additional QoS constraints. Lagrangean relaxation is then applied to relax such QoS constraints. This Lagrangean relaxation problem, after proper regrouping of the terms involved in the objective function, becomes a standard MDP problem (with a new revenue matrix compared with the original problem) and can be solved by standard liner programming techniques or the policy enhancement algorithm. Another primal heuristic based upon the policy enhancement algorithm is also developed for comparison purposed. It is expected that efficient and effective algorithms be developed by the proposed approach.

**Keywords: wireless networks, time slot allocation, optimization, delay, throughput, Markovian decision process, Lagrangean relaxation.**

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1 Introduction

## 1.1 Background

In the last decade, Internet has become more important in our daily life. With the growth of users, many services has been developed to provide more convenient services and entertainments on the Internet, while the demand of date transmission, such as web browsing, multimedia, and etc., has also increased dramatically.

Wireless access is more convenient for users to access the Internet by using their laptops or other handy devices at any place where the wireless access service is provided. New wireless networks technologies, such as the third-generation (3G) cellular system and IEEE 802.16 (Worldwide Interoperability for Microwave Access, WiMax), are designed to provide higher capacity for data services [12], [18], see Table 1 - 1; moreover, with the increasing demand of multimedia or other real-time services transmission, the Quality of Service (QoS) has become one of the important issues to the new wireless networks, and has been considered into the design to support different QoS requirements [11], [15], [19]. However, the standards only define the QoS architecture, the scheduling algorithm for the system with QoS-guaranteed transmission is not specified [11], [13].

The capacity of wireless networks is much less than those of the wired ones, see Table 1 - 2, because of some reasons, such as interference, the channel quality, and etc.; and the number of users who use the wireless access to get the Internet service has become very large. By the reasons mentioned above, the resource allocation is much more important to the wireless networks; with better resource allocation policy, the wireless networks will achieve higher capacity utilization under the required QoS of each service class.

Table 1 - 1 Types of Wireless Networks

| | **802.11 (Wi-Fi)** | **2.5G (GPRS)** | **3G (WCDMA)** | **3.5G (HSDPA)** | **802.16e (WiMAX)** |
|---|---|---|---|---|---|
| **Geography class** | Wireless LAN | Wireless WAN | Wireless WAN | Wireless WAN | Wireless MAN |
| **Bit rate** | b-11 Mbps g-54 Mbps n-300Mbps | 20~40 Kbps | 384/128 Kbps | 14.4M/384 K bps | 30 Mbps |
| **Transmit range** | 100 m | 1 km | 1 km | 1 km | 2~5 km |

Table 1 - 2 The Capacity of Different Networks

| | **Fiber (wired)** | **T3 (wired)** | **3.5G (wireless)** | **IEEE 802.16 (wireless)** |
|---|---|---|---|---|
| **Capacity** | Up to Gbps | 44.736 Mbps | 14.4 M/384 K bps | 75 Mbps (single channel) / 30 Mbps (for IEEE802.16e) |

Some mechanisms, such as time-division multiplexing (TDM), frequency division multiplexing (FDM), time division multiple access (TDMA), frequency division multiple access (FDMA), and code division multiple access (CDMA), have been adopted to improve the utilization of the channel capacity; many resource allocation methods had also been proposed to improve the capacity utilization of the new wireless networks [7-10], [14], [16], [17].

In this paper, we discuss how the resource allocation at the wireless base station (BS) based on the time slotted system can optimize the utilization of the capacity of the wireless networks and also take the QoS requirements into consideration.

## 1.2 Motivation

New wireless networks that are designed to provide higher capacity for wireless access will support the QoS-guaranteed data transmission. In the economic point of view, maximizing the total system revenue is one of the most important parts of the Internet service provider's targets. On the other hand, in the user's point of view, they want to use the wireless service and the corresponding QoS requirements are also being satisfied. There is a tradeoff between the system revenue and the QoS requirements. Internet service provider should not only maximize the revenue of the system but also has to achieve the QoS requirements to maintain the users' satisfaction. In such way, the users will be willing to use the wireless service again.

In [7-9], they had taken the system revenue into consideration, but [9] did not consider the QoS issue and [7], [8] only took the call blocking rate as the QoS requirement. In order to provide multimedia and other real-time services in new wireless networks, call blocking rate is not sufficient as the QoS requirement. Hence, we will take delay and throughput requirements as the QoS criteria. But delay is usually a very difficult issue for some system to estimate, we will do some approximation to calculate the delay of each class of services.

Given the buffering rule and the total capacity of a wireless networks which is a time slotted system, how to construct a best resource allocation policy so that the system revenue will be optimized while the QoS requirements also be satisfied is a very important issue for the Internet service providers.

We construct a state based resource allocation policy, where the state is defined by the situation of the system queue. Because the situation of the wireless networks is very dynamic, we describe this problem as a Markovian decision process problem, and then use the Markovian decision process to solve such a very dynamic resource allocation problem by constructing the optimal policy for the system with consideration of the QoS requirements, which are delay and throughput constraints.

## 1.3 Literature Survey

### 1.3.1 Quality of Service (QoS)

Quality of service is always an important issue for the multimedia transmission or other real-time service on the Internet. We can evaluate the quality of service in many points of view, such as throughput, delay, jitter, and reliability. The new wireless networks are designed to support QoS-guaranteed transmission. For example, in 3G and IEEE 802.16, the packets are classified into several classes of service with different QoS requirements; see Table 1 – 3, Table 1 - 4, [11], [19], and [20]. In [14], the author considered two mode of the bandwidth allocation for IEEE 802.16, namely, complete partitioning and complete sharing. With complete partitioning, a fixed amount of bandwidth is statically allocated for UGS (unsolicited grant service) while the remaining bandwidth is allocated for PS (polling service) and BE (best effort) services. In case of complete sharing, when the bandwidth requirement for UGS traffic is less than the given amount of bandwidth, the remaining available bandwidth will be available for PS. In [11], one of the QoS parameters is the traffic priority. Given two service flows identical in all QoS parameters besides priority, the higher priority service flow should be given lower delay and higher buffering preference.

In order to achieve the QoS requirement, there are many techniques that can be used to improve the QoS. For instance, “Traffic shaping” can smooth out the traffic on the server side and also can be used for traffic policing to monitor the traffic flow; “Resource reservation” can reserve the resource, including bandwidth and buffer space, to make sure the needed resource is available for transmitting the packets; “Admission control” the base station has to decide whether to admit or reject the incoming flow based on its capacity and how many commitments it has already made for other flows [21]. In recent years, many resource allocation methods had been proposed and the QoS issue also had been taken into consideration [7], [8], [14], [16], [17].

Table 1 - 3 Scheduling Services and Usage Rules of IEEE 802.16

| Scheduling type | Requirements | Example |
|---|---|---|
| UGS (unsolicited grant service) | Support real-time service flows that generate fixed-size data packets on a periodic basis | Voice over IP |
| rtPS (real-time polling service) | Support real-time service flows that generate variable size data packets on a periodic basis | Video streaming |
| nrtPS (non-real-time polling service ) | Support for non-real-time flows which require better than BE service | FTP |
| BE (Best effort) | No QoS guarantee | HTTP, E-mail |

Table 1 - 4 The QoS Classes of UMTS

| Traffic class | Fundamental characteristics | Example |
|---|---|---|
| Conversational (Real Time) | • Preserve time relation (variation) between information entities of the stream.<br>• Conversational pattern (stringent and low delay ) | Voice |
| Streaming (Real Time) | • Preserve time relation (variation) between information entities of the stream | Streaming video |
| Interactive (Best Effort) | • Request response pattern<br>• Preserve payload content | Web browsing |
| Background (Best Effort) | • Destination is not expecting the data within a certain time<br>• Preserve payload content | Telemetry, E-mail |

Source: http://www.umtsworld.com/technology/qos.htm

### 1.3.2 Resource Allocation for Wireless Networks Capacity

For wireless networks, the capacity is limited because that the used spectrum is restricted and shared between those users in the same wireless service region; the bandwidth for each user will decrease when the number of users in the same region becomes larger. Even though the new wireless networks technology can provide us with higher capacity, the bandwidth for each user still will not be sufficient when the number of users become very huge. Due to the reasons mentioned above, how to improve the utilization of the limited capacity is always an important issue of wireless networks.

In order to provide QoS-guaranteed data transmission, delay and throughput requirements are usually used as the QoS criteria. Many resource allocation methods had been proposed to maximize the utilization of the capacity with consideration of the QoS requirements. In some previous works, they used the deadline, which is the acceptable delay, of each packet to do the time slot allocation [16], [17]; some do the resource allocation according to the current situation of the queue in the system [14]; admission control was also used to control the incoming flows to make sure that the system can fully satisfy the QoS requirements of the new flows and the admitted ones [14], [17]. For cellular system, call blocking rate is an important criterion for evaluating the QoS satisfaction, [7], [8]. Besides, the common purpose of the previous works mentioned above is to maximize the utilization of the capacity under the given QoS requirements.

## 1.4 Proposed Approaches

In this paper, we describe a wireless system based on the states of the system queue, and then use the Markovian decision process (MDP) to fine the optimal policy for the system. However, with the additional QoS constraints, we can not apply the Markovian decision process to solve this problem directly. Fortunately, by adopting Lagrangean relaxation method, we can remove the QoS constraints and view the Lagrangean relaxation problem as a new Markovian decision process problem with different revenue matrix.

## 1.5 Thesis Organization

The remainder of this thesis is organized as follows. In chapter 2, we describe the research problem and the corresponding linear programming formulation for Markovian decision process. In chapter 3, the solution approaches, Markovian decision process and Lagrangean relaxation method, are proposed. In chapter 4, the computational results are presented. Finally, in chapter 5, we present the conclusions and the possible directions of future research of this thesis.

# Chapter 2 Problem Formulation

## 2.1 Problem Description

As shown in Figure 2 - 1, we consider a queuing system at the wireless base station. The packets in this system are classified into four service classes. This problem is to determine the best policy for the time slot allocation at the wireless BS to maximize the total system revenue under the consideration of delay and throughput requirements of each service class. When BS has data to be transmitted to the subscriber stations, the data has to enter the queue first and wait to be transmitted. The four service classes have different occupancy priorities that if there is no enough queuing space for the arrived packets, the packet with higher occupancy priority will enter the queue and one with lowest priority will be dropped even it was already in the queue.

Our system can be described by using the system state that is defined by the number of packet of each service class in the queue. For example, state (4, 3, 2, 1) means the number of service classes 1, 2, 3, and 4 in queue are four, three, two, and one respectively. In each state, the system can transmit at most $N$ packets in one frame. The different combination of the four service class packets that can be transmitted in one frame is called the "alternative." For instance, suppose the current state of the system with four service classes is (1, 2, 0, 3) and the maximum number of packets that can be transmitted in a frame is 3, then the alternatives of this state are (1, 2, 0, 0), (1, 1, 0, 1), (1, 0, 0, 2), (0, 2, 0, 1), (0, 1, 0, 2), and (0, 0, 0, 3).

In this problem, we want to find the optimal time slot allocation policy according to each state of the wireless base station, therefore, the transmission error will not be considered into this problem. The system state is discussed in chapter 2.1.1, and system alternative is discussed in chapter 2.1.2. The summary of problem description is listed in Table 2 - 6.

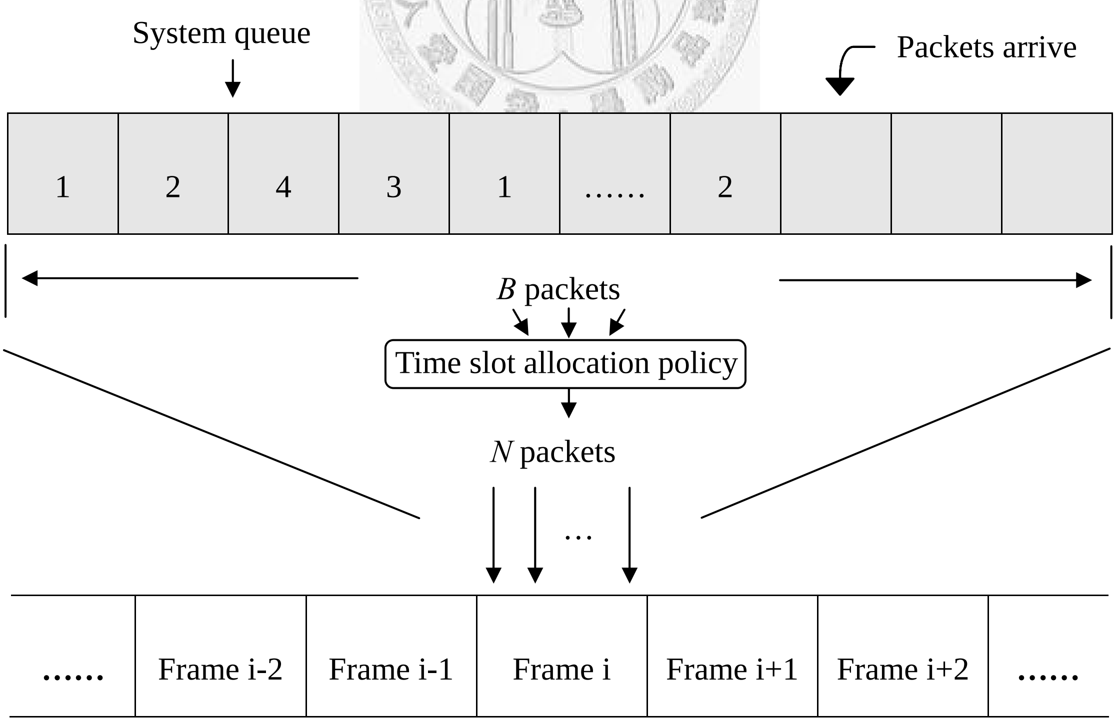


Figure 2 - 1 The Time Slot Allocation at The Wireless Station

### 2.1.1 System States

Suppose the buffer size of the system is $B$ packets and there are $m$ service classes. Then, the number of states $M$ of the system can be calculated as follows:

$$M = H_B^{m+1} = C_B^{B+(m+1)-1} = C_B^{B+m} = \frac{(B+m)!}{(B+m-B)! \cdot B!} = \frac{(B+m)!}{m! \cdot B!}$$

Given the buffer size is 12 packets and the number of service classes is 4, then, the number of states is $M = \frac{(12+4)!}{4! \cdot 12!} = 1820$. The rules and sequence of the states are presented in Table 2 - 1 and Table 2 - 2. We can also note that as the buffer size and service class number increased, the total number of states will increase dramatically, see Table 2 - 3.

Table 2 - 1 The Rules of The States

Given:

$B$: the buffer size of the system; $S$: the set of all states of the system;

$q_c^i$: the number of packets of service class $c$ in queue when system is in state $i$;

$M$: the set of all service classes of the system,

Then:

$$\sum_{c \in M} q_c^i = B, \quad q_c^i \geq 0, \forall i \in S$$

Expression:

$(q_1^i, q_2^i, ..., q_m^i,)$ : the expression of state $i$; $m$ is the number of service classes.

Table 2 - 2 The Sequence of The States ($B$=12, $N$=6, 4 service classes)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | ( 0, 0, 0, 0) | 2 | ( 1, 0, 0, 0) | 3 | ( 0, 1, 0, 0) | 4 | ( 0, 0, 1, 0) |
| 5 | ( 0, 0, 0, 1) | 6 | ( 2, 0, 0, 0) | 7 | ( 1, 1, 0, 0) | 8 | ( 1, 0, 1, 0) |
| 9 | ( 1, 0, 0, 1) | 10 | ( 0, 2, 0, 0) | 11 | ( 0, 1, 1, 0) | 12 | ( 0, 1, 0, 1) |
| 13 | ( 0, 0, 2, 0) | 14 | ( 0, 0, 1, 1) | 15 | ( 0, 0, 0, 2) | 16 | ( 3, 0, 0, 0) |
| 17 | ( 2, 1, 0, 0) | 18 | ( 2, 0, 1, 0) | 19 | ( 2, 0, 0, 1) | 20 | ( 1, 2, 0, 0) |
| 21 | ( 1, 1, 1, 0) | 22 | ( 1, 1, 0, 1) | 23 | ( 1, 0, 2, 0) | 24 | ( 1, 0, 1, 1) |
| 25 | ( 1, 0, 0, 2) | 26 | ( 0, 3, 0, 0) | 27 | ( 0, 2, 1, 0) | 28 | ( 0, 2, 0, 1) |
| 29 | ( 0, 1, 2, 0) | 30 | ( 0, 1, 1, 1) | 31 | ( 0, 1, 0, 2) | 32 | ( 0, 0, 3, 0) |
| 33 | ( 0, 0, 2, 1) | 34 | ( 0, 0, 1, 2) | 35 | ( 0, 0, 0, 3) | 36 | ( 4, 0, 0, 0) |
| ⋮ | | | | | | | |
| 1793 | ( 0, 2, 2, 8) | 1794 | ( 0,2, 1, 9) | 1795 | ( 0, 2, 0,10) | 1796 | ( 0, 1,11, 0) |
| 1797 | ( 0, 1,10, 1) | 1798 | ( 0, 1, 9, 2) | 1799 | (0, 1, 8, 3) | 1800 | ( 0, 1, 7, 4) |
| 1801 | ( 0, 1, 6, 5) | 1802 | ( 0, 1, 5, 6) | 1803 | ( 0, 1, 4, 7) | 1804 | ( 0, 1, 3, 8) |
| 1805 | ( 0, 1, 2, 9) | 1806 | ( 0, 1, 1,10) | 1807 | ( 0, 1, 0,11) | 1808 | ( 0, 0,12, 0) |
| 1809 | (0,0,11, 1) | 1810 | ( 0, 0,10, 2) | 1811 | ( 0, 0, 9, 3) | 1812 | ( 0, 0, 8, 4) |
| 1813 | (0, 0, 7, 5) | 1814 | ( 0, 0, 6, 6) | 1815 | ( 0, 0, 5, 7) | 1816 | ( 0, 0, 4, 8) |
| 1817 | ( 0, 0, 3, 9) | 1818 | ( 0, 0, 2,10) | 1819 | ( 0,0,1,11) | 1820 | ( 0, 0, 0,12) |

Table 2 - 3 The State Numbers with Different Buffer Size and Class Number

| Buffer size | Class number | Total number of states |
|---|---|---|
| 10 | 4 | 1001 |
| 10 | 5 | 3003 |
| 12 | 4 | 1820 |
| 12 | 5 | 6188 |
| 14 | 4 | 3060 |
| 14 | 5 | 11628 |
| 16 | 4 | 4845 |

### 2.1.2 Alternatives of The System

Suppose the maximum number of packets that can be transmitted in one frame is $N$, and the number of service classes is $c$, then the total number of alternative of this system can be calculated as follows:

$$K = H_N^{c+1} = C_N^{N+(c+1)-1} = C_N^{N+c} = \frac{(N+c)!}{(N+c-N)!\cdot N!} = \frac{(N+c)!}{c!\cdot N!}.$$

The rules of alternatives are presented in Table 2 - 4. Given the number of packets that can be transmitted in a frame is 6, then we can get the possible alternatives for the system, which are listed in Table 2 - 5. Some alternatives may not be available for some states, for example, alternative (3, 3, 0, 0) is not available for state (0, 0, 3, 3). In such case, we set the revenue of the alternative for this state to "$-\infty$" and the system will not choose it as the decision for this state.

Table 2 - 4 The Rules of Alternatives

Given:

$N$ : the maximum number of packets that can be transmitted in a frame;

$n_c^i$ : the number of packets of service class $c$ that will be transmitted in state $i$;

$M$ : the set of all service classes of the system; $S$: the set of all states of the system;

$q_c^i$ : the number of packets of service class $c$ in queue when system is in state $i$;

Then:

$\sum_{c\in M} n_c^i \le N$ , and $0 \le n_c^i \le q_c^i$ , $\forall\, i \in S, c \in M$

Expression:

$(n_1^i, n_2^i, ..., n_m^i)$ : the possible alternatives of state $i$; $m$ is the number of the service classes.

Table 2 - 5 Alternatives of The System ( $N$=6, 4 service classes)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | (0, 0, 0, 0) | 2 | (0, 0, 0, 1) | 3 | (0, 0, 0, 2) | 4 | (0, 0, 0, 3) |
| 5 | (0, 0, 0, 4) | 6 | (0, 0, 0, 5) | 7 | (0, 0, 0, 6) | 8 | (0, 0, 1, 0) |
| 9 | (0, 0, 1, 1) | 10 | (0, 0, 1, 2) | 11 | (0, 0, 1, 3) | 12 | (0, 0, 1, 4) |
| 13 | (0, 0, 1, 5) | 14 | (0, 0, 2, 0) | 15 | (0, 0, 2, 1) | 16 | (0, 0, 2, 2) |
| 17 | (0, 0, 2, 3) | 18 | (0, 0, 2, 4) | 19 | (0, 0, 3, 0) | 20 | (0, 0, 3, 1) |
| 21 | (0, 0, 3, 2) | 22 | (0, 0, 3, 3) | 23 | (0, 0, 4, 0) | 24 | (0, 0, 4, 1) |
| ⋮ | | | | | | | |
| 195 | (3, 3, 0, 0) | 196 | (4, 0, 0, 0) | 197 | (4, 0, 0, 1) | 198 | (4, 0, 0, 2) |
| 199 | (4, 0, 1, 0) | 200 | (4, 0, 1, 1), | 201 | (4, 0, 2, 0) | 202 | (4, 1, 0, 0) |
| 203 | (4, 1, 0, 1) | 204 | (4, 1, 1, 0) | 205 | (4, 2, 0, 0) | 206 | (5, 0, 0, 0), |
| 207 | (5, 0, 0, 1) | 208 | (5, 0, 1, 0) | 209 | (5, 1, 0, 0) | 210 | (6, 0, 0, 0) |

Table 2 - 6 Problem Descriptions

***Assumptions:***

1. There are ***m*** service classes in the system and the occupancy priority of them are: class 1 > class 2 >…> class *m.*
2. The data is divided into fixed size packets.
3. Each packet can be completely transmitted in a slot time, and each slot can transmit only one packet.
4. The channel quality will remain the same during the transmission.
5. Transmission error will not be considered in our system.
6. The arrival process of each service class is Poisson and independent to each other.

***Given parameters:***

1. The size of system queue is ***B*** packets, which will be shared among all service classes.
2. Packet arrival rate of each service class.
3. We defined revenue matrix for the four service classes.
4. State transition matrix.

***Objective:***

To Maximize the total system revenue.

***Subject to:***

1. The throughput and queuing delay requirement of each service classes.
2. The maximum number of packets that can be transmitted in a frame is *N*.

***To determine:***

The best time slot allocation policy of the system.

## 2.2 Problem Notations:

Table 2 - 7 Notation Descriptions for Given Parameters

| **Given Parameters** | |
|---|---|
| **Notation** | **Descriptions** |
| $M$ | The set of the service classes |
| $m$ | The number of service classes |
| $\lambda_c$ | The arrival rate of service class $c$, $c \in M$ |
| $R_c$ | System revenue of servicing one class $c$ packet, $c \in M$ |
| $N$ | The maximum number of packets that can be transmitted in a frame |
| $B$ | The queue size of the system (in packet), and $B \geq N$ |
| $S$ | The set of all states |
| $K$ | The set of all alternatives |
| $q_i^c$ | The number of packets of service class $c$ in state $i$, $c \in M$ , $i \in S$ |
| $D_c$ | The delay requirement of the service class $c$, $c \in M$ |
| $T_c$ | The throughput requirement of the service class $c$, $c \in M$ |
| $r_{ij}^k$ | The revenue from state $i$ to state $j$ given decision $k$, $r_{ij}^k = \sum_{c \in M} n_i^c(k) R_c$ |
| $r_i^k$ | The expected system revenue of state $i$ with decision $k$ |
| $n_i^c(k)$ | The number of packet of service class $c$ transmitted in state $i$ if the decision of state $i$ is $k$, $\sum_{c \in M} n_i^c(k) \leq N$ , $\forall\ c \in M$ , $i \in S$ |
| $P_{ij}^k$ | The probability from state $i$ to state $j$ given alternative $k$ |

Table 2 - 8 Notation Descriptions for Decision Variables

| **Decision Variable** | |
|---|---|
| **Notation** | **Descriptions** |
| $d_i^k$ | Conditional probability of choosing alternative $k$ given that the system is in state $i$ |
| $\pi_i$ | The limiting state probability of state $i$ that is independent of starting state |

## 2.2.1 State Transition Probability

We assume that the arrival processes of the four service classes are Poisson arrival with different arrival rate and are mutually independent. Hence, the probability that $x$ packets of service class $c$ arrived in a frame can be calculated by Poisson distribution: $P_c(x) = \frac{e^{-\lambda_c}\lambda_c^x}{x!}$, where $c \in M$ and $x = 0,1,2,3,\ldots$.

As the arrival processes of the four service classes are known, the state transition probability can be calculated by the following function $p_{ij}^k$ that means a system now occupied state $i$ will occupy state $j$ after its next transition given the decision is $k$. The function $p_{ij}^k$ is shown as follows:

$$P\{\text{ state } i\ (q_1,q_2,\ldots,q_m) \rightarrow \text{ state } j\ (q_1^{'},q_2^{'},\ldots,q_m^{'})\ |$$

$$\text{when choosing the alternative } k,\ (n_1,n_2,\ldots,n_m)\}$$

$$= P_{ij}^{k}$$

$$= \begin{cases} 0, \text{ if } \{ \sum_{c\in M} q_c^{'} < B \text{ and } (q_c - q_c^{'} > n_c,\ \exists\ c \in M) \} \\ \quad \text{or } \{ \sum_{c=1}^{t} q_c^{'} = B \text{ and } q_t^{'} \geq 1 \text{ and} \\ \quad\quad ( q_r - q_r^{'} > n_r,\ \exists\ r = \{1,\ldots,t-1\}),\ t \in M - \{1\} \} & \ldots\ldots(1) \\ \prod_{c\in M} P_c(x = q_c^{'} - q_c + n_c), \text{ if } \sum_{c\in M} q_c^{'} < B & \ldots\ldots(2) \\ P_1(x \geq B - q_1 + n_1), \text{ if } q_1^{'} = B & \ldots\ldots(3) \\ \{\prod_{c=1}^{t-1} P_c(x = q_c^{'} - q_c + n_c)\} \cdot P_t(x \geq \max(0,\ q_t^{'} - q_t + n_t)), \\ \quad \text{if } q_t^{'} \geq 1 \text{ and } \sum_{c=1}^{t} q_c^{'} = B, \text{ for } t \in M - \{1\} & \ldots\ldots(4) \end{cases}$$

As we can see above, even though the number of buffer size or the number of service classes is changed, the function $p_{ij}^{k}$ still can be used to calculate the transition probability correctly. Hence, the flexibility of function $p_{ij}^{k}$ $p_{ij}^{k}$ is very high for solving different size of problems.

**Explanation of function $p_{ij}^{k}$:**

Because the system in our problem is a time slotted system, we only discuss the discrete-time model. Equation (1) indicates the transitions that will not happen in our system. One case is that suppose the queue is not full after the transition and the difference between the number of two states of any class is large than $n_c$, which is the transmitted packet number of the service class *c*, then the probability of such transition is zero. For example, for a system with four service classes, given the current system state is in (10, 0, 2, 0) and the decision is (4, 0, 2, 0), it is impossible for the system to make a transition to state (3, 3, 1, 0). The other case is that the queue is full after the transition and class *t* has the lowest occupancy priority in the queue, if the difference between the number of the two states, before and after the transition, of any class is larger than $n_c$, where *c* can be 2, 3, ..., *m*, then, the probability of such transition is also zero. For instance, suppose the current system state is (3, 3, 3, 3) and the decision is (3, 0, 3, 0) given that *B*=12, it is impossible for the system to make a transition to state (9, 0, 2, 1).

Equation (2) indicates that if the queue is not full after the state transition, the probability can be calculated by multiplying the four arrival probability and the arrival number of each class are $q_c^{'} - q_c + n_c$, $\forall c \in M$. We defined $A_c$ as the number of arrivals of service class *c*. $A_c$ can be calculated by the equation: $q_c - n_c + A_c = q_c^{'}$, then $A_c = q_c^{'} - q_c + n_c$.

Equations (3) and (4) indicate that if the queue is full after the transition and service class *t* has the lowest occupancy priority between these packets in the queue that implies the packets with higher occupancy priority than class $t$ will not be dropped in this transition. Therefore, we can calculate the number of arrived packets in this transition by the equation: $A_c = q_c^{'} - q_c + n_c$, $c$ equals from 1 to $t-1$. Because the packet of class $t$ may be dropped in this transition, we set the arrival number of class $t$ packets is "equal to or large than" $A_t = q_t^{'} - q_t + n_t$. In some cases, $A_t$ may be less than zero, which is not reasonable, hence we set $A_t \geq \max(0,\ q_t^{'} - q_t + n_t)$ to avoid the unreasonable calculation of $A_t$. For example, given the current system state $i$ is (3, 0, 9, 0), the decision $k$ is (3, 0, 3, 0), the next state $j$ is (9, 0, 3, 0), and $B$=12, then the probability $p_{ij}^k$ can be calculated as follows:

$$p_{ij}^k = P_1(x = 9-3+3) \cdot P_1(x = 0-0+0) \cdot P_3(x \geq \max(0,\ 3-9+3)$$
$$= P_1(x = 9) \cdot P_1(x = 0) \cdot P_3(x \geq 0)$$

## 2.2.2 The Approximation of The Queuing Delay

Because the queuing system of this problem is very complicated, it is very difficult for us to estimate the queuing delay of each service class. In [22], Little's formulas had related the steady state mean system sizes to the steady state average customer waiting times as follows. The queuing delay, denoted as $W_q$, can be calculated by the equation $W_q = \frac{L_q}{\lambda}$, where $L_q$ is the expected number of packets in queue and $\lambda$ is the arrival rate of packets. One of the conditions of using Little's formulas is that the system must be a conservative system.

Therefore, we will use an approximation of the queuing delay function. The approximation of the queuing delay function of service class *c* will be the average number of packets of service class *c* in queue divided by the average number of transmitted packets of service class *c* in every time frame. When the drop rates of each service class are very low, the approximation is very accurate. The error rate between the real and the approximation queuing delay will be increased if the drop rate of each service class has increased.

## 2.3 Problem Formulation:

We can formulate the Markovian decision process into a linear programming problem formulation [2], which is shown as follows:

**Optimization Problem:**

**Objective function:**

$$\mathrm{E}_{\mathrm{LP1}} = Max\{\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k r_i^k\} \qquad \text{(LP 1)}$$

**Subject to:**

$$\pi_j = \sum_{i\in S}\sum_{k\in K}\pi_i d_i^k p_{ij}^k \qquad \forall\ j\in S \qquad \text{(LP 1.1)}$$

$$\pi_i \geq 0 \qquad \forall\ i\in S \qquad \text{(LP 1.2)}$$

$$\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k = 1 \qquad \text{(LP 1.3)}$$

$$\sum_{j\in S} p_{ij}^k = 1 \qquad \forall\ i\in S,\ k\in K \qquad \text{(LP 1.4)}$$

$$p_{ij}^k \geq 0 \qquad \forall\ i, j\in S,\ k\in K \qquad \text{(LP 1.5)}$$

$$d_i^k \geq 0 \qquad \forall\ i\in S,\ k\in K \qquad \text{(LP 1.6)}$$

$$\sum_{k\in K} d_i^k = 1 \qquad \forall\ i\in S \qquad \text{(LP 1.7)}$$

$$r_i^k = \sum_{j\in S} p_{ij}^k r_{ij}^k \qquad \forall\ i\in S,\ k\in K \qquad \text{(LP 1.8)}$$

$$r_i^k \geq 0 \qquad \forall\ i\in S,\ k\in K \qquad \text{(LP 1.9)}$$

$$\frac{\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k q_i^c}{\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k)} \leq D_c \qquad \forall\ c\in M \qquad \text{(LP 1.10)}$$

$$\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k) \geq T_c \qquad \forall\ c\in M \qquad \text{(LP 1.11)}$$

**Explanation of the objective function:**

The objective function of (LP 1) is to maximize the long term total system revenue under the situation that the system is stationary.

**Explanation of constraints:**

**[1] Steady State Constraints:**

Constraints (LP 1.1), (LP 1.2), and (LP 1.3) are the steady state constraints of the system. (LP 1.1) is the constraint $\pi = \pi P$, where $\pi = (\pi_0,\ \pi_1,...)$ represents the limiting probability vector of the system state and $P$ is the state transition probability matrix. Constraint (LP 1.3) describes that the summation of all the limiting probabilities must equal to 1. Constraints (LP 1.2) and (LP 1.3) jointly restrict the value of each $\pi_i$ must between 0 and 1.

**[2] State Transition Probability Constraints:**

Constraints (LP 1.4) and (LP 1.5) are related to the state transition probability. (LP 1.4) represents that the summation of all the transition probabilities, which the state transits from *i* to all states with decision *k*, must equal to 1. Constraint (LP 1.5) is the property of probability.

**[3] Decision Making Constraints:**

Constraints (LP 1.6) and (LP 1.7) are related to the decision variable $d_i^k$. In state $i$, the system will choose different alternative with different probabilities, and the summation of these probabilities is equal to 1. (LP 1.6) is the property of probability.

**[4] Revenue Constraints:**

Constraints (LP 1.8) and (LP 1.9) are about the revenue calculation.

**[5] QoS Constraints:**

Constraints (LP 1.10) and (LP 1.11) are the delay and throughput requirements of the four service classes. We also note that the numerator of the queuing delay is $\sum_{i \in S} \sum_{k \in K} \pi_i d_i^k q_i^c$ instead of $\sum_{i \in S} \pi_i q_i^c$ because of that we will do some reformulations in chapter 3 and this form, $\sum_{i \in S} \sum_{k \in K} \pi_i d_i^k q_i^c$, can make the reformulation more easily.

# Chapter 3 Solution Approaches

## 3.1 Introduction to Markovian Decision Processes

Markovian decision process (MDP) is an application of dynamic programming to solve a stochastic decision process that can be described by a finite number of states. The transition probabilities between the states are described by a Markov chain. The reward structure of the process is also described by a matrix whose individual elements represent the revenue (or cost) resulting from moving from one state to another. Both the transition and revenue matrices depend on the decision alternatives available to the decision maker. The objective of the problem is to determine the optimal policy that maximizes the expected revenue of the process over a finite or infinite number of stages [1].

Suppose the system has $X$ states and there are $k_i$ alternatives for each state $i$, where $i$=1,2,…,X, then there are $\prod_{i=1}^{X} k_i$ different policies. We can find the gain for each of these policies and choose the one with the largest gain as our optimal policy. However, it becomes unfeasible for very large problems. For example, a problem with 50 states and 50 alternatives in each state has $50^{50}$ ($\approx 10^{85}$) policies. Hence we introduce Policy Iteration Method to find the optimal policy in a small number of iterations.

### 3.1.1 Policy Iteration Method

Policy Iteration Method is composed of two parts, the Value-Determination Operation and the Policy-Improvement Routine. We first define the notations, which are listed in Table 3 - 1, which are used to describe the Policy Iteration Method.

**The Value-Determination Operation:**

Suppose that we are operating the system under a given policy so that we have specified a given Markov process with rewards. As we had defined above, $v_i(n)$ must obey the recurrence relation $v_i(n) = q_i + \sum_{j=1}^{X} p_{ij} v_j(n-1)$, $i = 1,2,...,X$, $n = 1,2,3,...$ and if $n$ is very large, then $v_i(n) = ng + v_i$ [1]. We now let $ng + v_i = q_i + \sum_{j=1}^{X} p_{ij} v_j(n-1)$, and substitute the term $(n-1)g + v_j$ for $v_j(n-1)$, then we can get the equation $ng + v_i = q_i + \sum_{j=1}^{X} p_{ij}[(n-1)g + v_i]$ for $i$=1,2,…,$X$. By doing some operations, the equation becomes $g + v_i = q_i + \sum_{j=1}^{X} p_{ij} v_j$ for each $i$. We have obtained a set of $X$ linear equations with $X$+1 unknowns, which are $X$ $v_i$ and one $g$. To solve this problem, we first set $v_X = 0$ and fine out the values of the other $X$ unknowns. After finding out these unknowns, we next use the relative values $v_i$ to find a better policy than the original one in the Policy-Improvement Routine.

Table 3 - 1 The Notations Used to Describe The Policy Iteration Method

| **Notation** | **Descriptions** |
|---|---|
| $v_i(n)$ | The total expected reward that the system will earn in $n$ moves if it starts from state $i$ under the given policy. |
| $p_{ij}$ | The probability that a system which now occupies state $i$ will occupy state $j$ after its next transition. |
| $p_{ij}^k$ | The probability that a system which now occupies state $i$ will occupy state $j$ after its next transition given the decision is $k$. |
| $r_{ij}$ | The reward associated with the transition from $i$ to $j$. |
| $q_i$ | The expected immediate return in state $i$, and $q_i = \sum_{j \in S} p_{ij} r_{ij}$ . |
| $q_i^k$ | The expected immediate return in state given the decision is $k$. |
| $g$ | The gain of the system, $g = \sum_{i \in S} \pi_i q_i$ . |

**The Policy-Improvement Routine:**

For each state $I$, find the alternative $k$ that maximizes the test quantity $q_i^k + \sum_{j=1}^{X} p_{ij}^k v_j$ using the relative values determined under the old policy. This alternative $k$ now becomes $d_i$, the decision in the $i$th state. A new policy has been determined when this procedure has been performed for every state.

If the policies on two successive iterations are identical, then we have got the optimal policy, which will maximize the system gain. Otherwise, use the new policy obtained on this Policy-Improvement Routine to do the Value-Determination Operation again until we have reached the optimal policy. The procedure of Policy Iteration Method is shown in Figure 3 - 1. Because we have additional constraints, which are delay and throughput constrains, in our problem, we can not use Policy Iteration Method directly to solve it. Hence we will introduce Lagrangean Relaxation Method in chapter 3.2 to cooperate with Policy Iteration Method to solve this problem that had been applied successfully in [8].

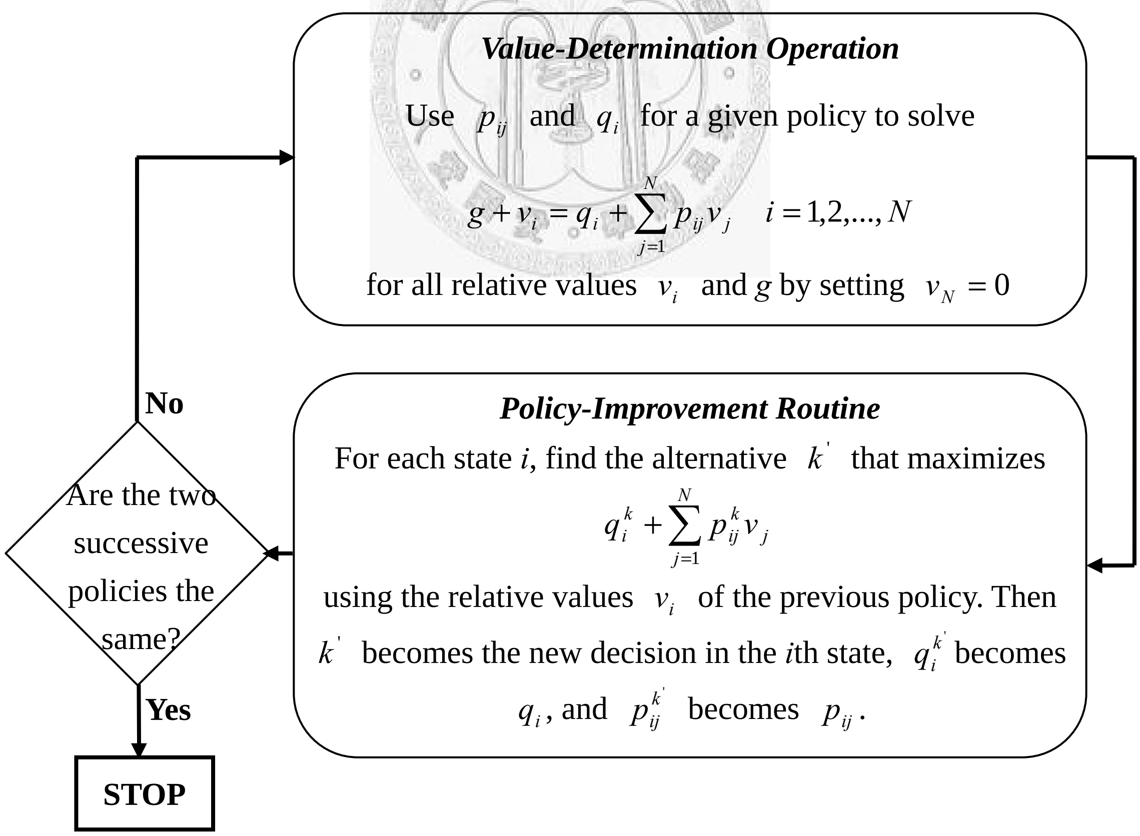


Figure 3 - 1 The Iteration Cycle

## 3.2 Introduction to Lagrangean Relaxation Method

Lagrangean relaxation method was first used to solve large-scale integer programming problems in the 1970s [4]. It can be used to solve the complicated mathematical problem more efficiently, and provide the excellent solutions for these problems. Hence, Lagrangean relaxation method has become one of the best tools for solving optimization problems, such as integer programming, linear programming with combinatorial objective function, and non-linear programming [5], [6]. The procedure of Lagrangean relaxation method will be described in the following.

By relaxing the complicated constraints of the primal mathematical formulation and add them to the objective function with corresponding Lagrangean multipliers ($\mu$), we produce a Lagrangean relaxation problem ($LR_{\mu}$) that will reduce the complexity of the primal problem. After relaxing some complicated constraints, we can decompose the primal problem into several independent subproblems that can be easily and optimally solved by proper algorithms.

With every subproblem being solved, we can get a boundary of the objective function of the primal problem, and is always a lower bound for a minimization problem. The Lagrangean relaxation method can provide us with some hints for designing a heuristic approach to get a primal feasible solution that will satisfy all the constraints of the primal problem and is an upper bound of the optimal solution to the primal problem.

The optimal solution of the primal problem is between the upper and lower bounds. We continuously adjust the multipliers by subgradient method to make the lower bound as large as possible, which is also called the Lagrangean dual problem. Lagrangean multipliers ( $\mu$ ) are also helpful for adjusting the heuristic. We can evaluate the goodness of the solution through the distance of the gap that a better solution will have a smaller gap. If the upper and lower bounds are identical, then we can declare that the optimal solution has been found. The procedure of the Lagrangean relaxation method is shown in Figure 3 - 2 and Figure 3 - 3.

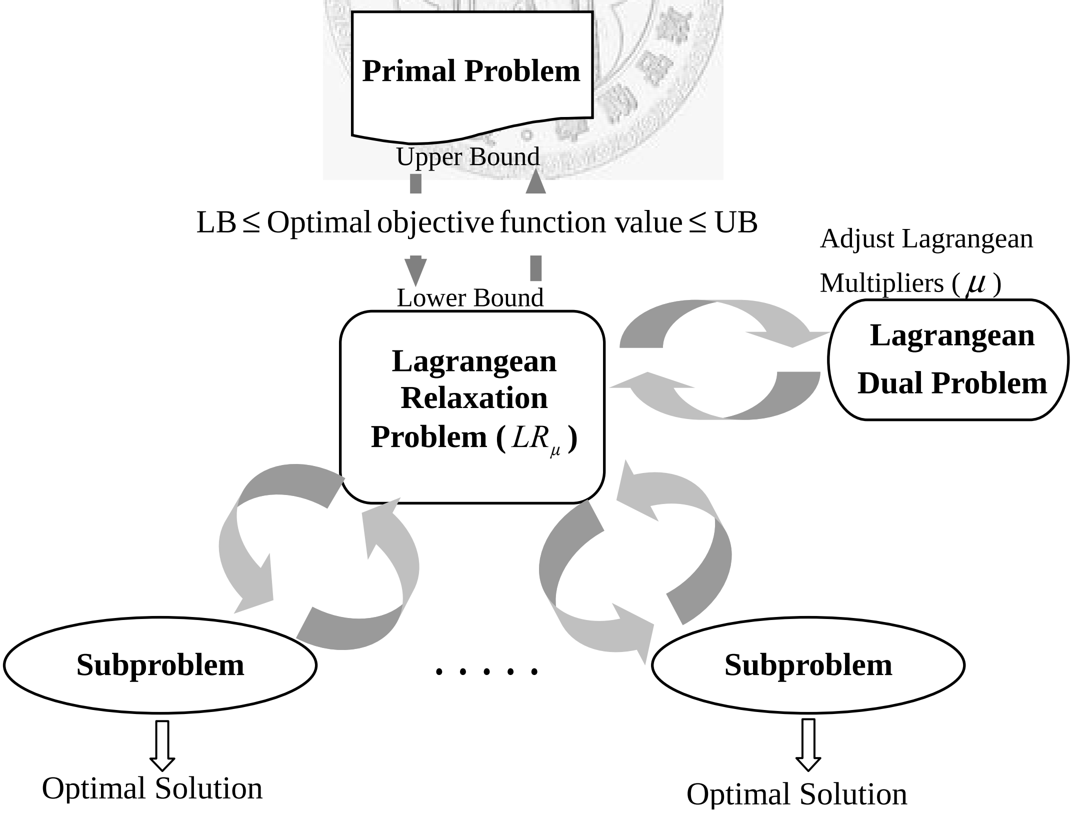


Figure 3 - 2 Illustration of The Lagrangean Relaxation Method

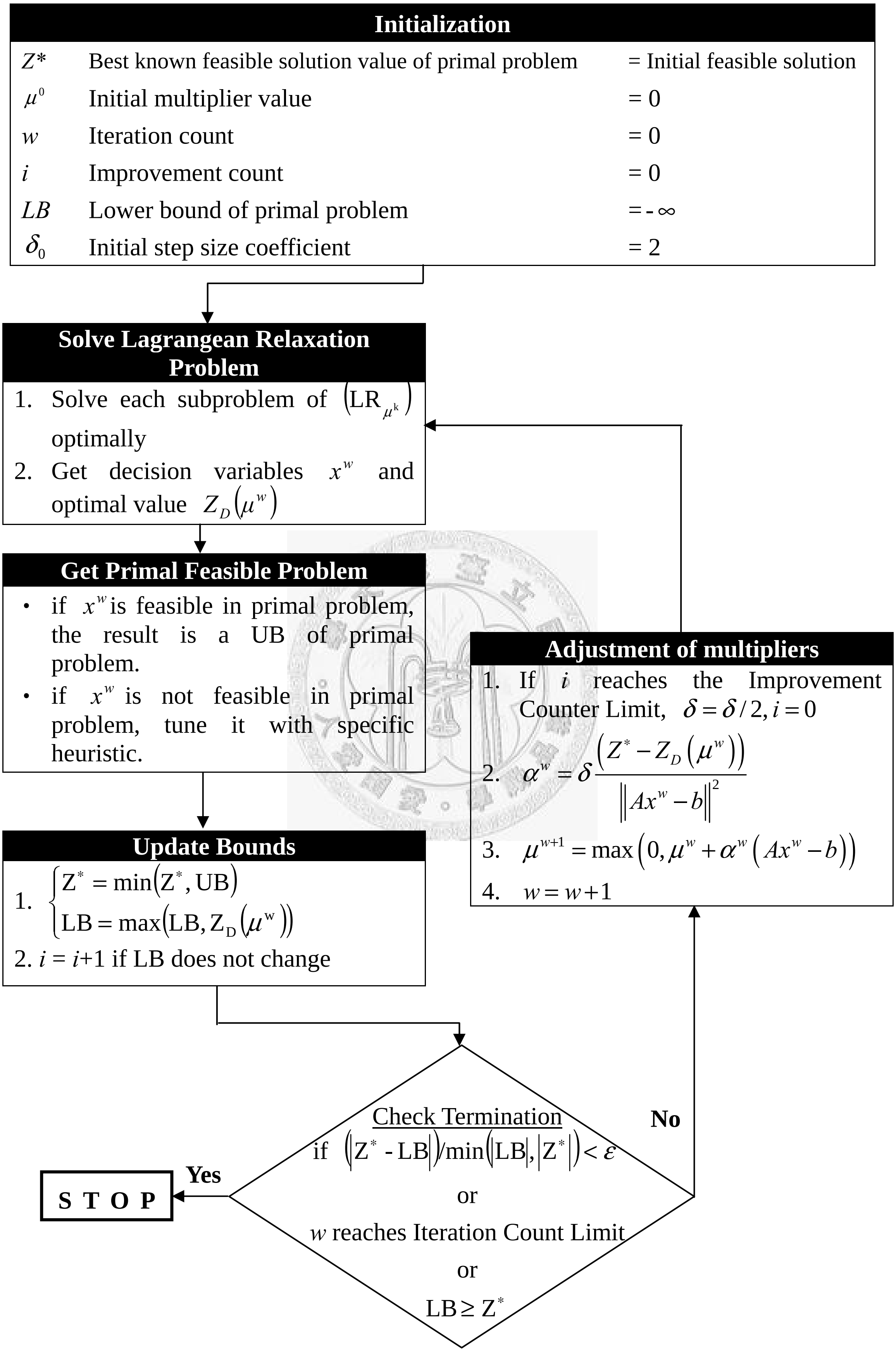


Figure 3 - 3 Lagrangean Relaxation Method Procedure

## 3.3 Lagrangean Relaxation

### 3.3.1 Problem Reformulation:

We first reformulate the objective function of (LP 1) into a minimum form, which will not affect the original result, and the changed formulation is shown as follows:

**Objective function:**

$$E_{LP2} = \min\{-\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k r_i^k\} \qquad \text{(LP 2)}$$

**Subject to:**

$$\pi_j = \sum_{i\in S}\sum_{k\in K}\pi_i d_i^k p_{ij}^k \qquad \forall\ j\in S \qquad \text{(LP 2.1)}$$

$$\pi_i \geq 0 \qquad \forall\ i\in S \qquad \text{(LP 2.2)}$$

$$\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k = 1 \qquad \text{(LP 2.3)}$$

$$\sum_{j\in S} p_{ij}^k = 1 \qquad \forall\ i\in S,\ k\in K \qquad \text{(LP 2.4)}$$

$$p_{ij}^k \geq 0 \qquad \forall\ i,j\in S,\ k\in K \qquad \text{(LP 2.5)}$$

$$d_i^k \geq 0 \qquad \forall\ i\in S,\ k\in K \qquad \text{(LP 2.6)}$$

$$\sum_{k\in K} d_i^k = 1 \qquad \forall\ i\in S \qquad \text{(LP 2.7)}$$

$$r_i^k = \sum_{j\in S} p_{ij}^k r_{ij}^k \qquad \forall\ i\in S,\ k\in K \qquad \text{(LP 2.8)}$$

$$r_i^k \geq 0 \qquad \forall\ i\in S,\ k\in K \qquad \text{(LP 2.9)}$$

$$\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k q_i^c \leq D_c \sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k) \qquad \forall\ c\in M \qquad \text{(LP 2.10)}$$

$$\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k) \geq T_c \qquad \forall\ c\in M \qquad \text{(LP 2.11)}$$

### 3.3.2 Lagrangean Relaxation:

By applying the Lagrangean relaxation method, we changed the primal problem (LP 2) into the following Lagrangean relaxation problem (LR 1), where Constraints (LP 2.10) and (LP 2.11) are relaxed. With a vector of Lagrangean multipliers, the Lagrangean relaxation problem of (LP 2) is shown as follows:

**Objective functions:**

$$
\begin{aligned}
&Z_D(\mu^D,\ \mu^T) \\
&= \min\{-[\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k r_i^k] + \sum_{c\in M}\mu_c^D(\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k q_i^c - D_c\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k)) \\
&\qquad + \sum_{c\in M}\mu_c^T(T_c - \sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k))\} \\
&= \min\{-\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k[r_i^k + \sum_{c\in M}(\mu_c^T n_i^c(k) + \mu_c^D(n_i^c(k)D_c - q_i^c))] + \sum_{c\in M}\mu_c^T T_c\}
\end{aligned}
\qquad \text{(LR 1)}
$$

**Subject to:**

| | | | | |
|---|---|---|---|---|
| $\pi_j$ | $=$ | $\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k p_{ij}^k$ | $\forall\ j\in S$ | (LR 1.1) |
| $\pi_i$ | $\geq$ | $0$ | $\forall\ i\in S$ | (LR 1.2) |
| $\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k$ | $=$ | $1$ | | (LR 1.3) |
| $\sum_{j\in S}p_{ij}^k$ | $=$ | $1$ | $\forall\ i\in S,\ k\in K$ | (LR 1.4) |
| $p_{ij}^k$ | $\geq$ | $0$ | $\forall\ i,j\in S,\ k\in K$ | (LR 1.5) |
| $d_i^k$ | $\geq$ | $0$ | $\forall\ i\in S,\ k\in K$ | (LR 1.6) |
| $\sum_{k\in K}d_i^k$ | $=$ | $1$ | $\forall\ i\in S$ | (LR 1.7) |
| $r_i^k$ | $=$ | $\sum_{j\in S}p_{ij}^k r_{ij}^k$ | $\forall\ i\in S,\ k\in K$ | (LR 1.8) |
| $r_i^k$ | $\geq$ | $0$ | $\forall\ i\in S,\ k\in K.$ | (LR 1.9) |

By doing some modification, the Lagrangean relaxation problem can be modified into Subproblem 1. And we can use Markovian decision process, which we have mentioned above, to easily find the optimal solution for subproblem 1.

### 3.3.3 Subproblem 1 (related to decision variables: $d_i^k$, $P_{ij}^k$, $\pi_i$, $n_i^c(k)$)

**Objective functions:**

$$\min\{-\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k [r_i^k + \sum_{c\in M}(\mu_c^T n_i^c(k) + \mu_c^D(n_i^c(k)D_c - q_i^c))]\} \qquad \text{(SUB 1)}$$

**Subject to:**

$$\begin{array}{lcll}
\pi_j & = & \sum_{i\in S}\sum_{k\in K}\pi_i d_i^k p_{ij}^k & \forall\ j\in S \qquad \text{(SUB 1.1)} \\
\pi_i & \geq & 0 & \forall\ i\in S \qquad \text{(SUB 1.2)} \\
\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k & = & 1 & \qquad \text{(SUB 1.3)} \\
\sum_{j\in S} p_{ij}^k & = & 1 & \forall\ i\in S,\ k\in K \qquad \text{(SUB 1.4)} \\
p_{ij}^k & \geq & 0 & \forall\ i,j\in S,\ k\in K \qquad \text{(SUB 1.5)} \\
d_i^k & \geq & 0 & \forall\ i\in S,\ k\in K \qquad \text{(SUB 1.6)} \\
\sum_{k\in K} d_i^k & = & 1 & \forall\ i\in S \qquad \text{(SUB 1.7)} \\
r_i^k & = & \sum_{j\in S} p_{ij}^k r_{ij}^k & \forall\ i\in S,\ k\in K \qquad \text{(SUB 1.8)} \\
r_i^k & \geq & 0 & \forall\ i\in S,\ k\in K \qquad \text{(SUB 1.9)}
\end{array}$$

## 3.4 The Dual Problem and The Subgradient Method

According to the algorithms proposed above, we can effectively solve the Lagrangean relaxation problem optimally. Based on the weak Lagrangean duality theorem [4], for any given set of nonnegative multipliers, $Z_D(\mu^D, \mu^T)$ yields a lower bound of $E_{LP2}$. We construct the following dual problem to calculate the tightest lower bound and solve the dual problem by using the subgradient method.

**Dual Problem (D)**

$Z_D^* = \max Z_D(\mu^D, \mu^T)$ **(D)**

Subject to:

$\mu^D, \mu^T \geq 0$

Let the vector $U$ be a subgradient of $Z_D(\mu^D, \mu^T)$. In iteration $w$ of the subgradient procedure, the multiplier vector $V^w = (\mu^{D,w}, \mu^{T,w})$ is updated by

$$V^{w+1} = V^w + \alpha^w U^w$$

where

$$U^w(\mu^D, \mu^T) = (\sum_{c \in M} D_c^{'} - D_c, T_c - \sum_{i \in S} \sum_{k \in K} \pi_i d_i^k n_i^c(k)).$$

and the step size, $\alpha^w$, is determined by

$$\alpha^w = \delta \frac{Z^* - Z_D(V^w)}{\left\|V^w\right\|^2}$$

where $Z^*$ is the best upper bound on the primal objective function value found by iteration $w$. Note that $\delta$ is a scalar between 0 and 2 and is usually initiated with the value, 2, and halved if the best objective function value does not improve within a given iteration count.

## 3.5 Getting Primal Feasible Solutions

After the subproblem 1 had been solved, we can get some hints from the associated multipliers and decision variables, and then use the information to find a primal feasible solution for (LP 1). The proposed heuristic has two phases, Feasible_Solution and Objective_Value_Improvement. We will do Feasible_Solution before Objective_Value_Improvement. The procedures of these two phases are described in the following.

First, the decision of some states can easily be determined before we start the procedure of Feasible_Solution. For some states $i$ that the queue length is smaller than $N$, we can set the alternative $k$ that $n_i^c(k) = q_i^c$ as the decisions for these states; moreover, if the packets in the queue belong to the same service class, the decision for such state $i$ must be the alternative $k$ *that* $n_i^c(k) = N$. Hence, $d_i^k = 1$ and $d_i^h = 0, \forall\, h \in K - \{k\}$ for those states $i$ and alternatives $k$ mentioned above.

After that, we adjust the decisions for the remainder states by applying the proposed Feasible_Solution algorithm to get the feasible solution to the original problem. The proposed Feasible_Solution algorithm is divided into two stages.

In the beginning of the Feasible_Solution stage 1, we first sort the steady state probabilities, $\pi_i$, that are solved by MDP of each state from large to small. Then, the violation factor of each service class will be calculated by the equation, $\max(\frac{\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k q_i^c}{\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k)} - D_c,\ T_c - \sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k))$. We choose the state *i* that has the highest steady state probability as the first state for the decision adjustment. For such chosen state *i* and its original decision *k*, we adjust one slot from the service class *Ta* that has the lowest violation factor to service class *Tb* that has the highest violation factor if $n_i^{Ta}(k) \geq 1$ and $0 \leq n_i^{Tb}(k) < q_i^{Tb}$. Next we will choose the state *j* that has the steady state probability only lower than the highest one to do the decision adjustment, and so and so forth.

The adjustment of the decision in one iteration will be repeated until the total number of changed decisions has reached a given limit, denoted as *decision_change_limit*. When the *decision_change_limit* has been reached, we will calculate the steady state probabilities again.

If the violation status of some QoS requirements had been changed in the procedure of the decision adjustment, we called such a phenomenon “oscillation”. If there is an oscillation happens in the decision adjustment procedure, it may because that we changed too many decisions in one iteration. To reduce the probability that the oscillation will happen, we can modify *decision_change_limit* by the rule described in Table 3 – 4. If the *decision_change_limit* is small for a certain number of iteration, we will adopt Feasible_Solution stage 2 to adjust the decisions. The only one difference between stage 1 and 2 of Feasible_Solution is that we randomly choose states to do decision adjustment.

Finally, when all the constraints of each service class have been satisfied, we then stop the procedure of Feasible_Solution, and go to the next procedure, Objective_Value_Improvement.

In the procedure of Objective_Value_Improvement, we will adjust decisions to make the UB as better as possible while the feasibility remains the same. We start the adjustment from the state that the corresponding steady state probability is the lowest one. In each state, we will try to adjust one slot to the service class that has the higher reward. The adjustment will be repeated until the policy become infeasible, and the last feasible policy is the primal feasible solution to the original problem.

The detail procedures of Heuristic_LR_stage_1, Heuristic_LR_Stage_2, and Objective_Value_Improvement are described in Table 3 – 2 and Table 3 – 4. The detail rule of adjusting *decision_change_limit* is shown in Table 3 – 3. The flow of the proposed getting primal feasible solution algorithm is shown as Figure 3 – 4.

Table 3 - 2 Phase 1: Feasible_Solution (stage 1 and 2)

**Step 1:**

Sort the steady state probabilities of each state from large to small.

**Step 2:**

Calculate the violation factor.

$$violation\_c = \max\left(\frac{\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k q_i^c}{\sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k)} - D_c,\ T_c - \sum_{i\in S}\sum_{k\in K}\pi_i d_i^k n_i^c(k)\right),$$

for $c \in M$ and *violation_c* $\geqq 0$ means that the relative constraints has been violated. Then, we sort the degree of violation from large to small. "*violation_order_1*" represents the service class which has the largest violation factor.

**Step 3:**

Stage 1: Select states which have large steady state probability to do decision adjustment.

Stage 2: Randomly select states to do decision adjustment.

**while** (*total_changed_decision* < *decision_change_limit*)

{

**IF** ( (*violation_order_4* service class did not violate the relative constraints) &&

(there is at least one slot that can be moved from the *violation_order_4* service class to the *violation_order_1* service class) )

{

Move one slot from the *violation_order_4* service class to the *violation_order_1* service class in this state;

*total_changed_decision* ++；

}

**ELSE IF** ((*violation_order_3* service class did not violate the relative constraints)

&& (there is at least one slot that can be moved from the *violation_order_3* service class to the *violation_order_1* service class) )

{

Move one slot from the *violation_order_3* service class to the *violation_order_1* service class in this state;

*total_changed_decision* ++；

}

**ELSE IF** ((*violation_order_2* service class did not violate the relative constraints)

&& (there is at least one slot that can be moved from the *violation_order_2* service class to the *violation_order_1* service class))

{

Move one slot from the *violation_order_2* service class to the *violation_order_1* service class in this state;

*total_changed_decision* ++；

}

Next state；

}

**Step 4:**

Calculate the delay and throughput performance of each service class.

**IF** (all constraints are being satisfied)

```
{
    Stop the procedure of Phase 1: Feasible_Solution;
    Go to Phase 2: Objective_Value_Improvement;
}

ELSE
{
    Adjust decision_change_limit according to the rule described in Table 3 – 3;
    Calculate the steady state probabilities of each state;
    IF (oscillation_limit has been reached)
        Go to Step 2 and take Feasible_Solution stage 2;
    ELSE
        Go to Step 1 and take Feasible_Solution stage 1;
}
```

Table 3 - 3 The Rule of Adjusting The Parameter, *decision_change_limit*

```
IF (no oscillation)
{    // Y is a small number
     IF ( (violation_Cn < Y for all  n∈ M ) &&
          (decision_change_limit > threshold_A) )
          Set decision_change_limit to threshold_A;
     ELSE
          decision_change_limit remains the same;
}


ELSE// oscillation
{
     IF (threshold _B < decision_change_limit < threshold _A)
          Reduce decision_change_limit by one unit;
     ELSE IF ( (violation_Cn < Y for all  n∈ M ) &&
                    (decision_change_limit > threshold _A) )
          Set decision_change_limit to threshold_ A;
     ELSE
        decision_change_limit = decision_change_limit / 2;
}
```

Table 3 – 4 Phase 2: Objective_Value_Improvement

**Step 1:**

Sort the steady state probabilities of each state.

**Step 2:**

**while (***is_feasible***)**

{

//start from the state which has the lowest steady state probability

Adjust one slot from the service class which has the lower reward to the one which has the higher reward;

IF( new UB is better than the original UB )

Update the value of UB;

Check feasibility; //if not feasible, then *is_feasible* = **false**

//the final feasible solution is the getting primal solution to the problem

Next state;

}

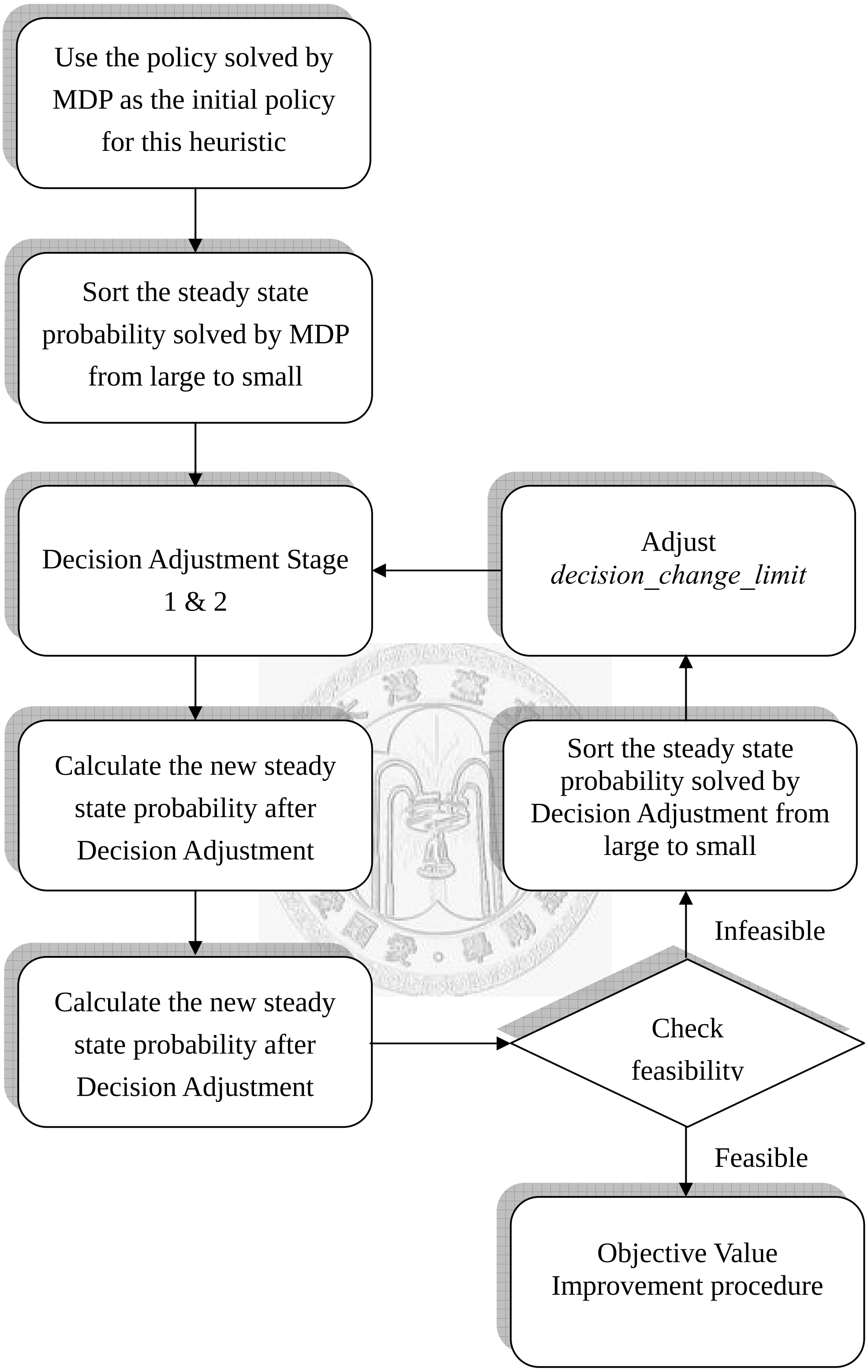


Figure 3 – 4 The Flow of The Proposed Getting Primal Feasible Solution Algorithm

# Chapter 4 Computational Experiments

In this chapter, in order to test the quality of the proposed getting primal feasible solution, we construct several experiments, and also compare the results of our proposed heuristic with one simple algorithm.

Here we denoted our dual solution as *LB*, Lagrangean Relaxation based heuristic as *LR* and simple algorithm based as *SA*. We use two metrics, "Gap" and "Improvement Ratio" to evaluate our solution quality. Gap is calculated by $\frac{LB-LR}{LR}\times 100\%$. And Improvement Ratio is calculated by $\frac{LR-SA}{SA}\times 100\%$. We have to note that the values of *LR*, *LB* and *SA* are lower than zero because we have reformulated the objective function in a minimum form.

## 4.1 Simple Algorithm

We will use a non-iteration based algorithm to compare with our proposed iteration based algorithm. This algorithm will use the "weight" to allocate the slots to each service class. The weight of each service class takes the number of packets in queue of each service class and the throughput and delay requirements into consideration.

At each state, we first assign one slot to a service class which has the highest weight, and then divide the corresponding weight by two, and do the assignment again until all slots have been assigned.

## 4.2 Experimental Environment

Table 4 – 1 Experimental Environment and Parameters

| **Parameter** | **Value** |
| --- | --- |
| Service class | 4 service classes |
| Reward | (2.0, 1.5, 1.0, 0.5),<br>(2.0, 1.0, 0.5, 0.5),<br>(2.0, 1.0, 1.0, 1.0),<br>(2.0, 2.0, 2.0, 2.0). |
| Queue size ($B$) | 12, (state number = 1820)<br>14, (state number = 3060)<br>16, (state number = 4845) |
| $N$ | 6, (total alternatives = 210) |
| Test Platforms | PC Ⅰ: CPU: Intel(R) Core(TM)2 CPU6400@2.13GHz/2.13GHz,<br>RAM: 1.99GB,<br>OS: Microsoft Windows XP Home Edition Version 2002 Service Pack 2. |
| | PC Ⅱ: CPU: Intel(R) Pentium(R) 4 CPU 3.00GHz/2.99GHz,<br>RAM: 992MB,<br>OS: Microsoft Windows XP Professional Version 2002 Service Pack 2. |

## 4.3 Experimental Scenarios

We design several scenarios to test the quality of our proposed algorithm under different setting of parameters. These scenarios are listed as follows:

1. Different queue sizes under different revenue matrixes.
2. The performance under different QoS requirements.
3. The impact under different adjustments of *decision_change_limit*.

In scenario 4, we also want to know the impact of different adjustments of the parameter, *decision_change_limit*, in the procedure of the proposed getting primal feasible solution algorithm. We will show the number of iterations that is needed to achieve the QoS requirements and the objective values of different adjustments.

If the algorithm can not find a feasible solution, the objective value of such experiment will be set to zero.

## 4.4 Different Queue Sizes under Different Revenue Matrixes

Table 4 – 2 Parameters of Different Queue Sizes under Different Revenue Matrixes

| **Parameters** | **Value** |
|---|---|
| Queue size (*B*) | A: 12, (state number = 1820), |
| | B: 14, (state number = 3060), |
| | C: 16, (state number = 4845). |
| *N* | 6, (total alternatives = 210) |
| Reward | R1: (2.0, 1.5, 1.0, 0.5), |
| | R2: (2.0, 1.0, 0.5, 0.5), |
| | R3: (2.0, 1.0, 1.0, 1.0), |
| | R4: (2.0, 2.0, 2.0, 2.0). |
| Arrival rates | (1.2, 1.8, 1.8, 6.0) |
| Delay requirements | (2.0, 2.5, 3.0, 5.0) |
| Throughput requirements | (1.15, 1.70, 1.70, 1.30), |

Table 4 – 3 Experiment Results of Different Queue Sizes under Different Revenue Matrixes

| | | **LR** | **LB** | **Gap** | **SA** | **Improvement Ratio** |
|---|---|---|---|---|---|---|
| R1 | A | -7.447951 | -7.482082 | 0.4583% | -7.446970 | 0.0132% |
| | B | -7.449532 | -7.449532 | 0.0000% | 0 | 100% |
| | C | -7.449970 | -7.497685 | 0.6405% | 0 | 100% |
| R2 | A | -5.698607 | -5.699954 | 0.0236% | -5.698414 | 0.0034% |
| | B | -5.699639 | -5.699996 | 0.0063% | 0 | 100% |
| | C | -5.699982 | -5.700000 | 0.0003% | 0 | 100% |
| R3 | A | -7.199956 | -7.199956 | 0.0000% | -7.199956 | 0% |
| | B | -7.199997 | -7.199997 | 0.0000% | 0 | 100% |
| | C | -7.200000 | -7.200000 | 0.0000% | 0 | 100% |
| R4 | A | -11.999912 | -11.999912 | 0.0000% | -11.999910 | 0% |
| | B | -11.999993 | -11.999993 | 0.0000% | 0 | 100% |
| | C | -12.000000 | -12.000000 | 0.0000% | 0 | 100% |

Table 4 – 4 Throughput and Delay Performances of LR and SA

| | | | **Throughput of LR** | **Throughput of SA** | **Delay of LR** | **Delay of SA** |
|---|---|---|---|---|---|---|
| R1 | A | Class 1 | 1.200000 | 1.200000 | 1.156201 | 1.884772 |
| | | Class 2 | 1.797135 | 1.796874 | 2.220194 | 1.749122 |
| | | Class 3 | 1.701676 | 1.700237 | 1.457381 | 1.557620 |
| | | Class 4 | 1.301145 | 1.302845 | 3.124508 | 2.970310 |
| | B | Class 1 | 1.200000 | 1.200000 | 1.365969 | 1.008447 |
| | | Class 2 | 1.799269 | 1.799826 | 2.497360 | 2.036595 |
| | | Class 3 | 1.700529 | 1.714118 | 1.939676 | 2.688103 |
| | | Class 4 | 1.300199 | 1.286053 | 3.454667 | 3.451985 |
| | C | Class 1 | 1.200000 | 1.200000 | 1.156949 | 1.008450 |
| | | Class 2 | 1.799966 | 1.799921 | 2.483053 | 2.665125 |
| | | Class 3 | 1.700009 | 1.728898 | 2.868190 | 2.929956 |
| | | Class 4 | 1.300025 | 1.271180 | 3.991548 | 3.815397 |
| R2 | A | Class 1 | 1.200000 | 1.200000 | 1.153895 | 1.884772 |
| | | Class 2 | 1.797259 | 1.796874 | 2.215669 | 1.749122 |
| | | Class 3 | 1.702319 | 1.700237 | 1.460557 | 1.557620 |
| | | Class 4 | 1.300378 | 1.302845 | 3.129643 | 2.970310 |
| | B | Class 1 | 1.200000 | 1.200000 | 1.377573 | 1.008447 |
| | | Class 2 | 1.799281 | 1.799826 | 2.489728 | 2.036595 |
| | | Class 3 | 1.700630 | 1.714118 | 1.978979 | 2.688103 |
| | | Class 4 | 1.300086 | 1.286053 | 3.403232 | 3.451985 |
| | C | Class 1 | 1.200000 | 1.200000 | 1.193619 | 1.008450 |
| | | Class 2 | 1.799965 | 1.799921 | 2.474072 | 2.665125 |
| | | Class 3 | 1.700031 | 1.728898 | 2.849865 | 2.929956 |
| | | Class 4 | 1.300003 | 1.271180 | 3.994115 | 3.815397 |

| | | | Throughput of LR | Throughput of SA | Delay of LR | Delay of SA |
|---|---|---|---|---|---|---|
| R3 | A | Class 1 | 1.200000 | 1.200000 | 1.026024 | 1.884772 |
| | | Class 2 | 1.799766 | 1.796874 | 1.562383 | 1.749122 |
| | | Class 3 | 1.700080 | 1.700237 | 2.111245 | 1.557620 |
| | | Class 4 | 1.300110 | 1.302845 | 3.300042 | 2.970310 |
| | B | Class 1 | 1.200000 | 1.200000 | 1.373122 | 1.008447 |
| | | Class 2 | 1.799284 | 1.799826 | 2.495484 | 2.036595 |
| | | Class 3 | 1.700074 | 1.714118 | 1.976482 | 2.688103 |
| | | Class 4 | 1.300639 | 1.286053 | 3.402033 | 3.451985 |
| | C | Class 1 | 1.200000 | 1.200000 | 1.199246 | 1.008450 |
| | | Class 2 | 1.799965 | 1.799921 | 2.477648 | 2.665125 |
| | | Class 3 | 1.700026 | 1.728898 | 2.851628 | 2.929956 |
| | | Class 4 | 1.300009 | 1.271180 | 3.981659 | 3.815397 |
| R4 | A | Class 1 | 1.200000 | 1.200000 | 1.032485 | 1.884772 |
| | | Class 2 | 1.799567 | 1.796874 | 1.651193 | 1.749122 |
| | | Class 3 | 1.700072 | 1.700237 | 2.036819 | 1.557620 |
| | | Class 4 | 1.300317 | 1.302845 | 3.268207 | 2.970310 |
| | B | Class 1 | 1.200000 | 1.200000 | 1.054745 | 1.008447 |
| | | Class 2 | 1.799924 | 1.799826 | 1.779050 | 2.036595 |
| | | Class 3 | 1.700020 | 1.714118 | 2.676395 | 2.688103 |
| | | Class 4 | 1.300053 | 1.286053 | 3.772955 | 3.451985 |
| | C | Class 1 | 1.200000 | 1.200000 | 1.065703 | 1.008450 |
| | | Class 2 | 1.799910 | 1.799921 | 2.139183 | 2.665125 |
| | | Class 3 | 1.700036 | 1.728898 | 2.998666 | 2.929956 |
| | | Class 4 | 1.300054 | 1.271180 | 4.381194 | 3.815397 |

## 4.5 The Performance under Different QoS Requirements

Table 4 – 5 Parameters of The Performance under Different QoS Requiremetns

| Parameters | Value |
|---|---|
| Queue size ($B$) | 12, (state number = 1820) |
| $N$ | 6, (total alternatives = 210) |
| Reward | (2.0, 1.5, 1.0, 0.5) |
| Arrival rates | (1.2, 1.8, 1.8, 6.0) |
| Delay requirements | (2.0, 2.5, 3.0, 5.0) |
| Throughput requirements | A: (1.15, 1.70, 1.70, 1.25),<br>B: (1.15, 1.70, 1.65, 1.25),<br>C: (1.15, 1.70, 1.70, 1.30),<br>D: (1.15, 1.70, 1.65, 1.35),<br>E: (1.15, 1.70, 1.60, 1.40),<br>F: (1.15, 1.70, 1.55, 1.45),<br>G: (1.15, 1.70, 1.50, 1.50). |

Table 4 – 6 Experiment Results of The Performance under Different QoS Requirements

| | LR | LB | Gap | SA | Improvement Ratio |
|---|---|---|---|---|---|
| A | -7.474845 | -7.485633 | 0.1443% | -7.419138 | 0.7453% |
| B | -7.474845 | -7.485633 | 0.1443% | -7.449697 | 0.3364% |
| C | -7.448439 | -7.482082 | 0.4517% | -7.446970 | 0.0197% |
| D | -7.424186 | -7.475957 | 0.6973% | -7.421915 | 0.0306% |
| E | -7.399047 | -7.465256 | 0.8948% | -7.397112 | 0.0262% |
| F | -7.373882 | -7.448913 | 1.0175% | -7.373018 | 0.0117% |
| G | -7.347679 | -7.436975 | 1.2153% | 0 | 100% |

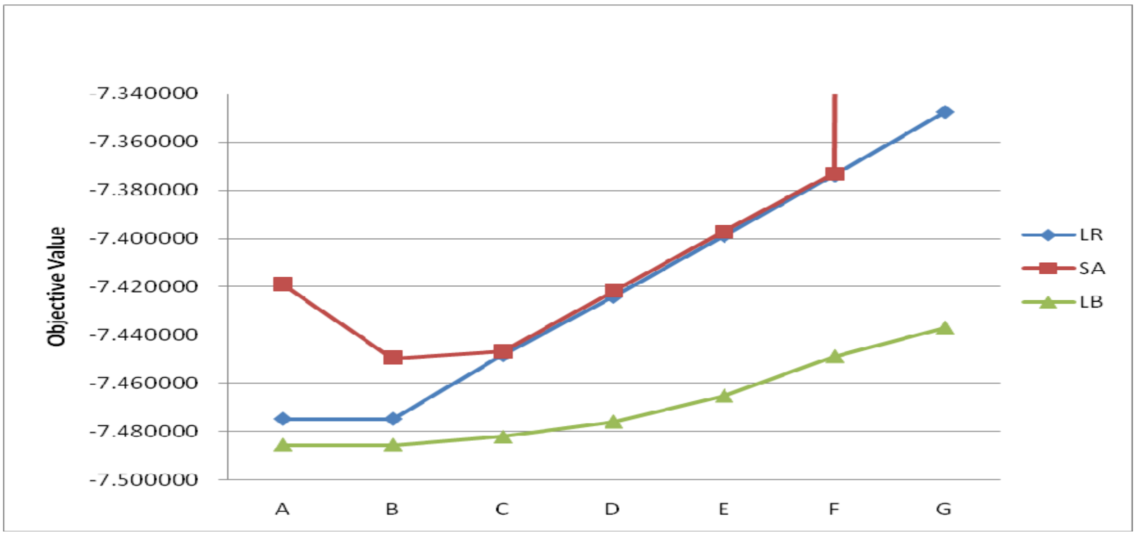

Figure 4 – 1 Objective Values under Different QoS Requirements

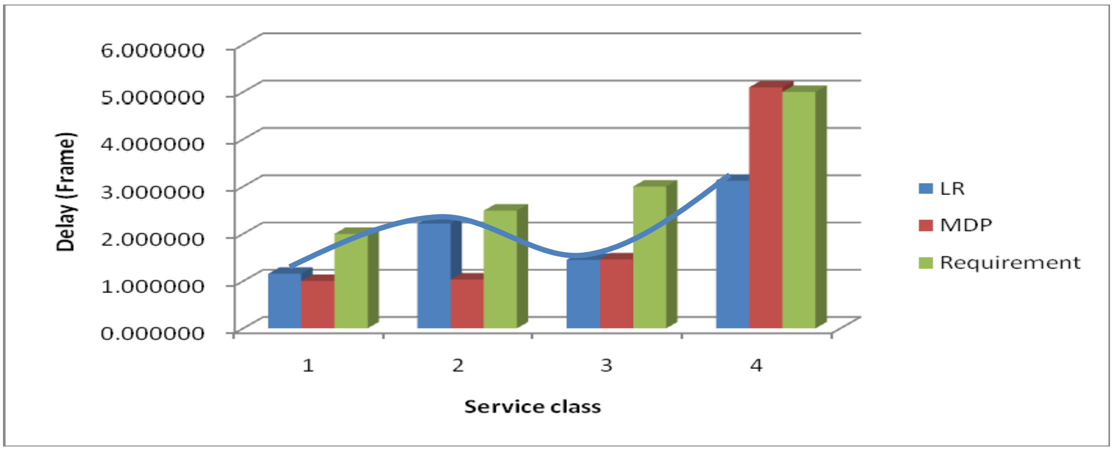

Figure 4 – 2 The Queuing Delay under Throughput Requirements, (1.15, 1.7, 1.7, 1.3)

Figure 4 – 3 The Queuing Delay under Throughput Requirements, (1.15, 1.7, 1.65, 1.35)

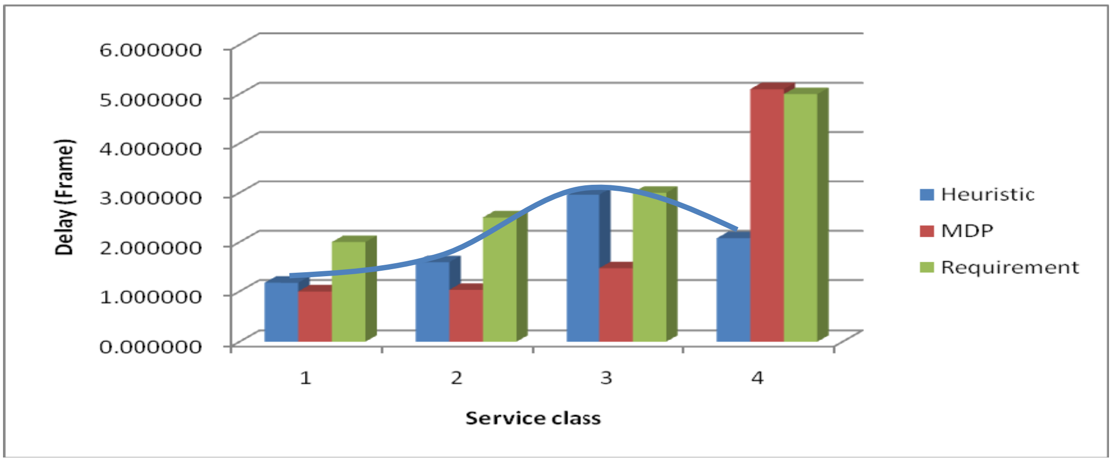


Figure 4 – 4 The Queuing Delay under Throughput Requirements, (1.15, 1.7, 1.6, 1.4)

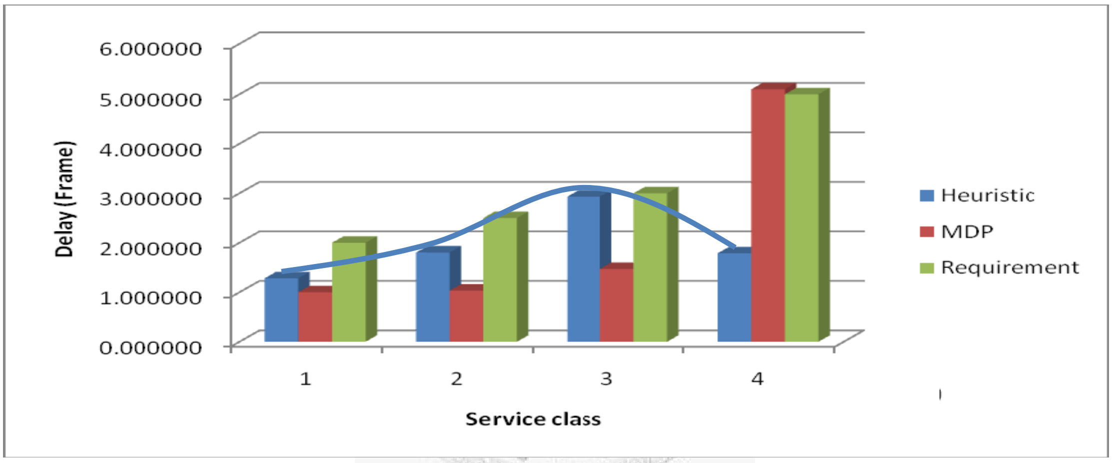


Figure 4 – 5 The Queuing Delay under Throughput Requirements, (1.15, 1.7, 1.55, 1.45)


Figure 4 – 6 The Queuing Delay under Throughput Requirements, (1.15, 1.7, 1.5, 1.5)

## 4.6 The Impact under Different Adjustments of *decision_change_limit*

Table 4 – 7 Parameters of The Experiments of The Impact under Different Adjustments of *decision_change_limit*

| **Parameters** | **Value** |
|---|---|
| Queue size (*B*) | 12, (state number = 1820) |
| *N* | 6, (total alternatives = 210) |
| Reward | (2.0, 1.5, 1.0, 0.5) |
| Arrival rates | (1.2, 1.8, 1.8, 6.0) |
| Throughput requirements | (1.15, 1.7, 1.6, 1.4) |
| Queuing delay requirements | (2.0, 2.5, 3.0, 5.0) |

Table 4 – 8 The Results of Different Adjustments of *decision_change_limit* (The Initial Value of *decision_change_limit* is 80)

| ***threshold_A*** | ***threshold_B*** | **Number of Iterations** | **System Revenue** |
|---|---|---|---|
| 30 | 5 | 13 | 7.398827 |
| | 10 | 14 | 7.399143 |
| | 15 | 18 | 7.398719 |
| 40 | 5 | 21 | 7.399027 |
| | 10 | 25 | 7.399407 |
| | 15 | 37 | 7.399369 |

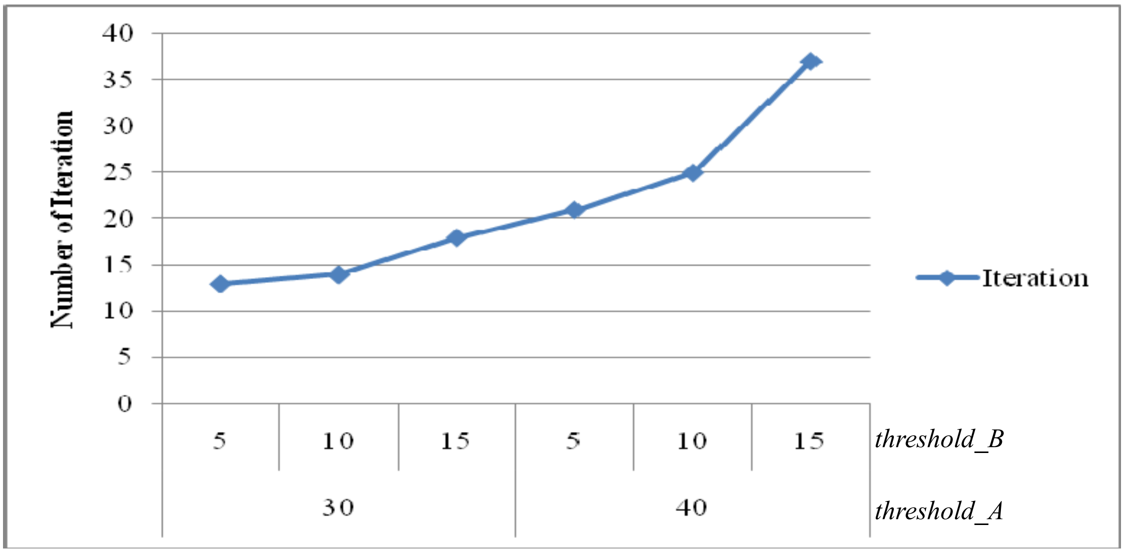


Figure 4 – 7 The Number of Iterations of Different Adjustments of *decision_change_limit* (The Initial Value of *decision_change_limit* is 80)

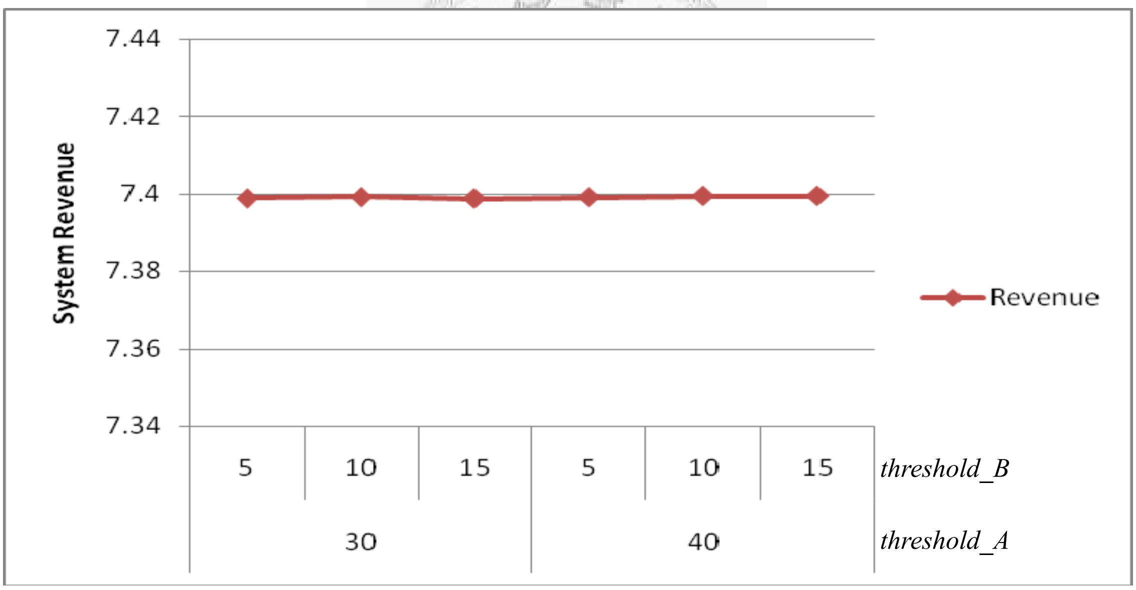


Figure 4 – 8 The System Revenue of Different Adjustments of *decision_change_limit* (The Initial Value of *decision_change_limit* is 80)

Table 4 – 9 The Results of Different Adjustments of *decision_change_limit*
(The Initial Value of *decision_change_limit* is 120)

| ***threshold_A*** | ***threshold_B*** | **Number of Iterations** | **System Revenue** |
|---|---|---|---|
| 30 | 5 | 24 | 7.398918 |
| | 10 | 24 | 7.398918 |
| | 15 | 38 | 7.398833 |
| 40 | 5 | 24 | 7.398918 |
| | 10 | 24 | 7.398918 |
| | 15 | 38 | 7.398833 |
| 50 | 5 | 24 | 7.398918 |
| | 10 | 24 | 7.398918 |
| | 15 | 38 | 7.398833 |

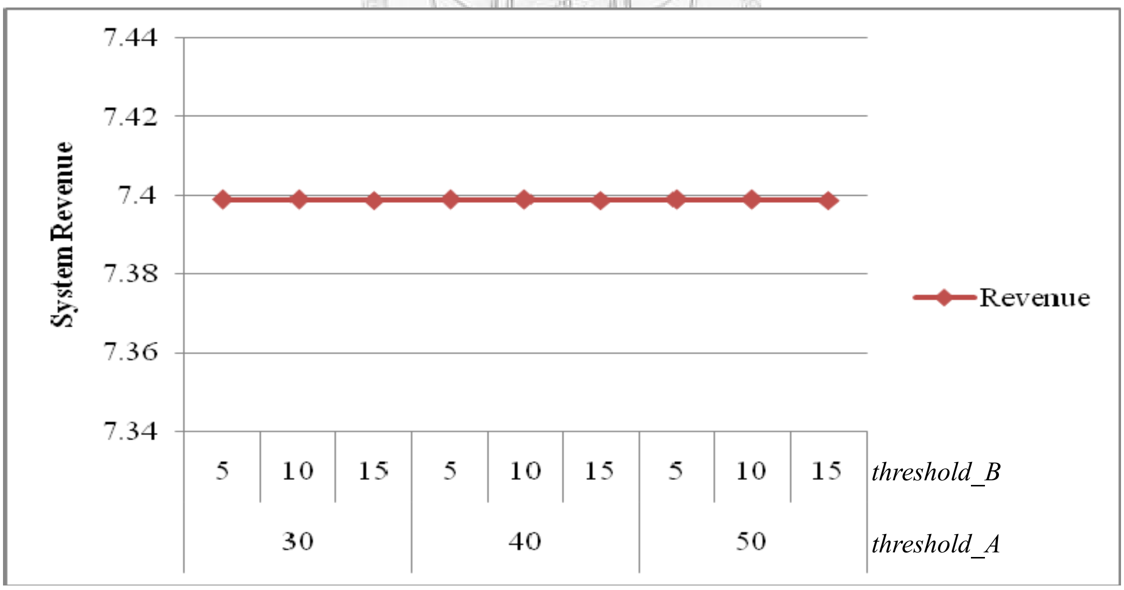


Figure 4 – 9 The System Revenue of Different Adjustments of *decision_change_limit*
(The Initial Value of *decision_change_limit* is 120)

Table 4 – 10 The Results of Different Initial Values of *decision_change_limit*

| | **Initial Value of *decision_change_limit*** | **Number of Iterations** | **System Revenue** |
|---|---|---|---|
| *threshold_A* = 30, *threshold_B* = 10. | 60 | 29 | 7.399342 |
| | 80 | 14 | 7.399143 |
| | 120 | 24 | 7.398918 |
| *threshold_A* = 40, *threshold_B* = 10. | 80 | 25 | 7.399407 |
| | 120 | 24 | 7.398918 |

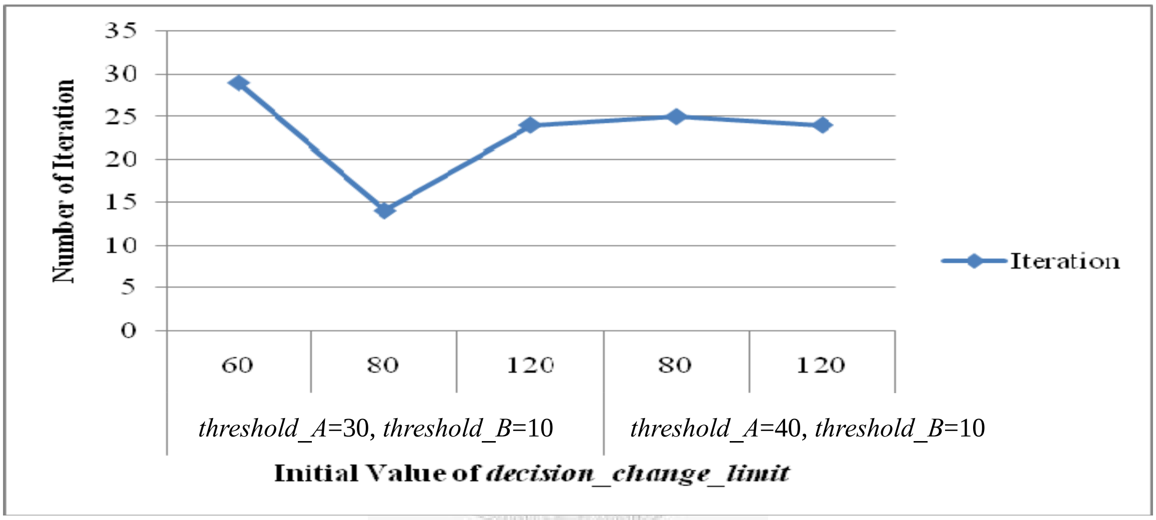


Figure 4 – 10 The Number of Iterations of Different Initial Values of *decision_change_limit*

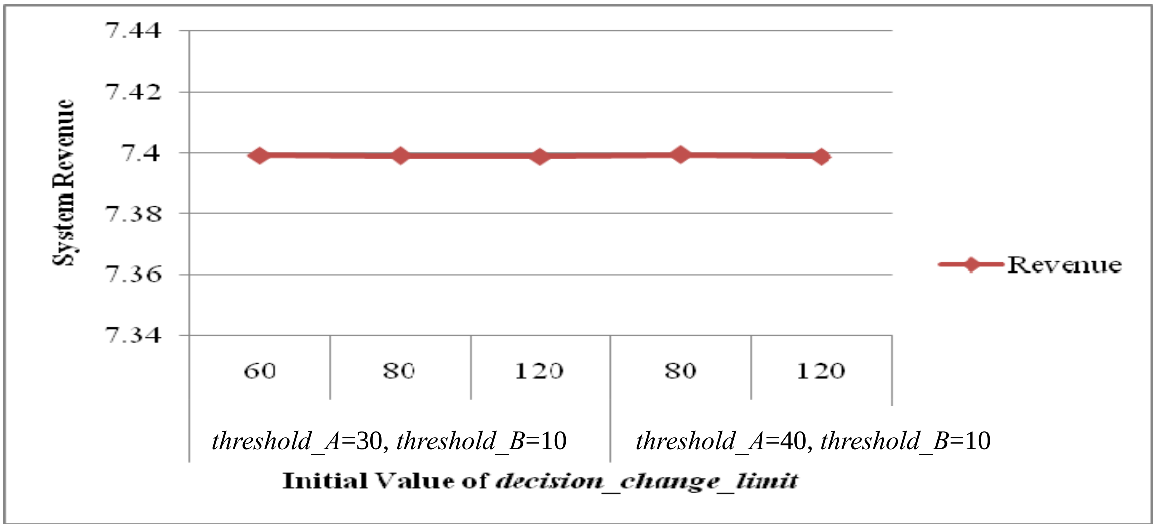


Figure 4 – 11 The System Revenue of Different Initial Values of *decision_change_limit*

## 4.7 Discussions of The Experiment Results

### 4.7.1 The Objective Value and The Improvement Ratio

As we can see in the experiment results shown above, the objective values will depend on the throughput performances. Therefore, if the throughput requirements are set very loosely, both of LR and SA can easily find a feasible solution to the problem. But, when the throughput requirements are set very tightly, it becomes much harder for LR and SA to find a feasible solution.

As we have mentioned above, the objective value will depend on the throughput performance, therefore, when the throughput requirements are set very tightly, the throughput performances will be very close to throughput requirements when a feasible solution have been found. According to this reason we have discussed above, the improvement ratio will become very small if both of LR and SA can find a feasible solution to the problem for which the throughput requirements are set very tightly. As the requirements are getting tighter, SA may be unable to find a feasible solution, but our proposed algorithm can still find a feasible solution to the problem.

### 4.7.2 The Change of the Queuing Delay under Different throughput Requirements

Because of the different occupancy priorities, the service class which has the lower occupancy priority will have higher probability to be dropped. When the throughput requirement of such service class is increased, the corresponding queuing delay will be decreased. The reason is that the system wants to transmit the packets of such service class, which has the lower occupancy priority, before it has been dropped by other service class to enlarge the throughput performance, so it has higher chance than before to be transmitted. Therefore, the queuing delay of this service class will be decreased when the throughput requirement becomes larger under the same arrival rates of each service class.

### 4.7.3 The Impact of Different Adjustments of *decision_change_limit*

In the adjustment rule of *decision_change_limit*, there are three parameters, *threshold_A*, *threshold_B,* and the initial value of *decision_change_limit,* that can be modified to suit different total number of states. The experiment results show that different settings of the parameters affect the objective values very slightly but affect the total number of iterations that are needed to find the feasible solution

When *threshold_A* and *threshold_B* become larger, the total number of iterations will become larger because that it may cause more oscillations in the decision adjustment procedure before the feasible solutions have been found. In the other hand, too small initial value of the *decision_change_limit* may need more iterations to find the feasible solutions because the improvement of each adjustment iteration is relatively small.

# Chapter 5 Conclusion and Future Work

## 5.1 Summary

In this thesis, we emphasize on a problem of finding a time slot allocation policy to a queuing system under the given QoS requirements, throughput and queuing delay, to maximize the long term system revenue. We formulate this problem as a linear programming problem and the objective function is to maximize the long term system revenue. In chapter 3, we develop a Lagrangean Relaxation based heuristic combined with Markovian Decision Process to solve this problem. In chapter 4, the experiment results show that our proposed algorithm can easily find a near optimal feasible solution to the problem and outperform the simple algorithm. The contributions of this thesis are listed as follows:

1. We proposed a mathematical formulation and use an optimization based algorithm to find out the near optimal policy for the queuing system under the throughput and queuing delay requirements.
2. We constructed a general form of state transition probability for such queuing system.

## 5.2 Future Work

In this thesis, we have known that the arrival rate will dominate the throughput performance because of the occupancy priority. Therefore, we can do some modification of the occupancy rule of the queue space. For example, we can divided the queue into two partitions, and packets in one of the partitions will not be dropped even there is a packet with higher occupancy priority want to enter the queue. The illustration of such queuing system is shown as Figure 5 – 1. We also can consider that packets of some service class will not be dropped when they are already in the queue.

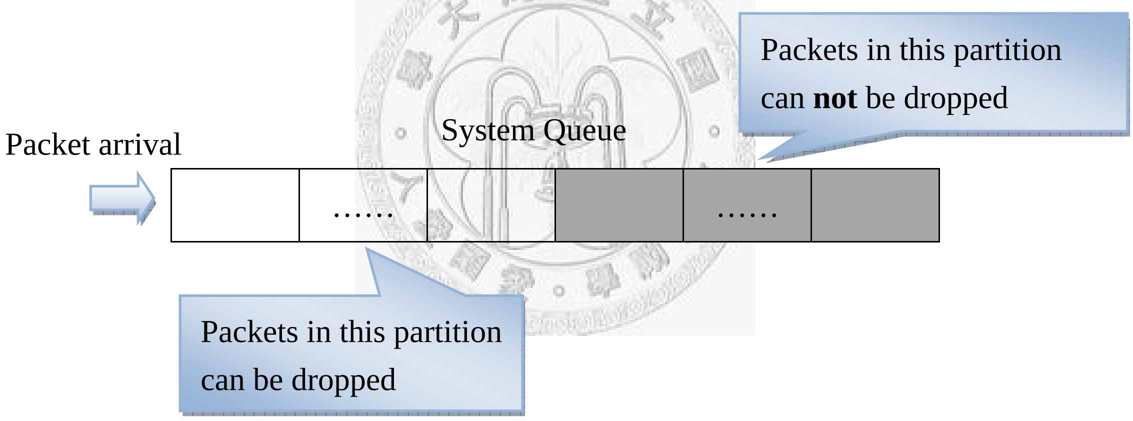


Figure 5 – 1A Modified Queuing System

In the problem formulation, we used an approximation of the queuing delay function to calculate the delay time. Because the error ratio will increase when the total arrival rate of all service classes becomes larger than the limited number of packets that can be transmitted in one frame, another future work is that we can try to find a more accurate queuing delay function for our problem formulation.

The queuing system discussed in this thesis has only one system queue, and we can try to extend it to multiple system queues for multiple communication channels to accommodate different types of the network system.

# References


[1] Ronald A. Howard, "Dynamic Programming and Markov Processes," Wiley, 1970.

[2] Hamdy A. Taha, "Markovian Decision Process," *Operations Research: An Introduction*, 5th edition, pp. 682-712, Macmillan Publishing Company, 1992.

[3] K. Karen Yin, Hu Liu, and Neil E. Johnson, "Markovian inventory policy with application to the paper industry," *Computers and Chemical Engineering*, Vol. 26, NO. 10, pp. 1399-1413(15), October, 2002.

[4] Marshall L. Fisher, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science*, Vol. 27, NO. 1, pp. 1-18, January 1981.

[5] Marshall L. Fisher, "An application Oriented Guide to Lagrangian Relaxation," *Interfaces*, Vol. 15, NO. 2, pp. 10-21, April 1985.

[6] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin, "Lagrangian Relaxation and Network Optimization," *Network Flows: Theory, Algorithms, and Applications*, pp. 598-638. Prentice-Hall, Inc. 1993.

[7] Hsu-Kuan Hung, ADVISER: Yeong-Sung Lin, "Optimization of GPRS Time Slot Allocation," June, 2001.

[8] Hui-Ting Chuang, ADVISER: Yeong-Sung Lin, "Optimization of GPRS Time Slot

Allocation Considering Call Blocking Probability Constraints," June, 2002.

[9] Peter Marbach and Randall Berry, "Downlink Resource Allocation and Pricing for Wireless Networks," *IEEE INFOCOM*, 2002.

[10] Min Cao, Wenchao Ma, Qian Zhang, Xiaodong Wang, and Wenwu Zhu, "Modelling and Performance Analysis of the Distributed Scheduler in IEEE802.16 Mesh Mode," *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing MobiHoc '05*, May, 2005.

[11] "IEEE Standard for Local and metropolitan area networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems," *IEEE Computer Society and the IEEE Microwave Theory and Techniques Society*, 2004.

[12] Steven J. Vaughan-Nichols, "Achieving Wireless Broadband with WiMax," *Computer*, Vol. 37, Issue 6, June, 2004.

[13] Carl Eklund, Roger B. Marks, Kenneth L. Stanwood, and Stanley Wang, "IEEE Standard 802.16: A Technical Overview of the WirelessMAN$^{TM}$ Air Interface for Broadband Wireless Access," *IEEE Communications Magazine*, pp. 98-107, June, 2002.

[14] Dusit Niyato and Ekram Hossain, "Queue-Aware Uplink Bandwidth Allocation and Rate Control for Polling Service in IEEE 802.16 Broad band Wireless Networks," *IEEE TRANSACTIONS ON MOBILE COMPUTING*, VOL. 5, NO. 6,

JUNE, 2006.

[15] Claudio Cicconetti, Luciano Lenzini, Enzo Mingozzi and Carl Eklund, "Quality of Service Support in IEEE 802.16 Networks," *IEEE Network*, March/April, 2006.

[16] Qingwen Liu, Xin Wang, and Georgios B. Giannakis, "A Cross-Layer Scheduling Algorithm With QoS Support in Wireless Networks," *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, VOL. 55, NO. 3, MAY 2006.

[17] Kitti Wongthavarawat and Aura Ganz, "Packet scheduling for QoS support in IEEE 802.16 broadband wireless access systems," *INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS*, 2003.

[18] K. W. Richardson, "UMTS overview," *ELECTRONICS & COMMUNICATION ENGINEERING JOURNAL*, June, 2000.

[19] Vijay K. Garg and Oliver T.W. Yu, "Integrated QoS Support in 3G UMTS Networks," *Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE*, Vol. 3, pp. 1187-1192, 2000.

[20] http://www.umtsworld.com/technology/qos.htm

[21] Andrew S. Tanenbaum, "QUALITY OF SERVICE," *Computer Network*, 4th edition, pp. 397-417, Pearson Education, Inc. Publishing as Prentice Hall PTR, 2003.

[22] Donald Gross Carl M. Harris "Fundamentals of Queueing Theory," 3rd edition, pp.

10-13 and 141-143, Wiley-Interscience Publication, 1998.

# 簡 歷

姓　名：林岦毅

出生地：臺灣省臺中市

出生日：中華民國七十一年十月九日

學　歷：九十年九月至九十四年六月

國立政治大學應用數學系學士

九十四年九月至九十六年七月

國立臺灣大學資訊管理研究所碩士