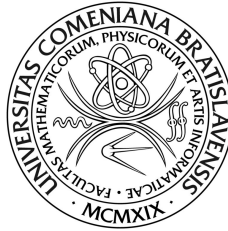


COMENIUS UNIVERSITY IN BRATISLAVA
Faculty of Mathematics, Physics and Informatics

INDEX POLICIES FOR DYNAMIC AND
STOCHASTIC PROBLEMS

COMENIUS UNIVERSITY IN BRATISLAVA
Faculty of Mathematics, Physics and Informatics
Department of Applied Mathematics and Statistics



INDEX POLICIES FOR DYNAMIC AND STOCHASTIC PROBLEMS

Bachelor's thesis

Vladimír Novák

Supervisor:
Mgr. Peter Jacko, PhD.

Co-Supervisor:
Mgr. Jana Szolgayová, PhD.

Branch of study: 9.1.9 Applied Mathematics
Study programme: Economic and Financial Mathematics

Registration number: ed46f30a-0712-4ecb-817b-dd18791671f5

BRATISLAVA 2011

UNIVERZITA KOMENSKÉHO V BRATISLAVE
Fakulta matematiky, fyziky a informatiky
Katedra aplikovanej matematiky a štatistiky



INDEXOVÉ STRATÉGIE PRE DYNAMICKÉ A STOCHASTICKÉ ÚLOHY

Bakalárska práca

Vladimír Novák

Školiteľ :
Mgr. Peter Jacko, PhD.

Pomocný školiteľ :
Mgr. Jana Szolgayová, PhD.

Študijný odbor: 9.1.9 Aplikovaná matematika
Študijný program: Ekonomická a finančná matematika

Evidenčné číslo: ed46f30a-0712-4ecb-817b-dd18791671f5

BRATISLAVA 2011



ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Vladimír Novák
Študijný program: ekonomická a finančná matematika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.1.9. aplikovaná matematika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský


Názov : Index Policies for Dynamic and Stochastic Problems

Cieľ : The goal of the thesis is to learn the method of derivation of an index policy, to apply this method to a specific problem of job scheduling with abandonment and to present a computational study comparing the derived index policy to the optimal policy and to alternative strategies.

Vedúci : Mgr. Peter Jacko, PhD.

Dátum zadania: 02.11.2010

Dátum schválenia: 09.11.2010


.....
doc. RNDr. Margaréta Halická, CSc.
garant študijného programu



.....

študent


.....

vedúci práce

Dátum potvrdenia finálnej verzie práce, súhlas s jej odovzdaním (vrátane spôsobu sprístupnenia)


.....

vedúci práce

Abstrakt

Novák, Vladimír: Indexové stratégie pre dynamické a stochastické úlohy [Bakalárska práca].

Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky,
Katedra aplikovanej matematiky a štatistiky.

Školiteľ: Mgr. Peter Jacko, PhD.

Pomocný školiteľ: Mgr. Jana Szolgayová, PhD.

Bratislava 2011

V našej práci sa zaoberáme Whittlovou metódou odvodenia indexových stratégií pre problémy formulované v prostredí Markovovských rozhodovacích procesov. Analyzujeme model pre rozvrhovanie úloh užívateľov viacerých tried, v ktorom užívatelia môžu aj odchádzať, ak ich úloha nie je ukončená včas. Naším cieľom je minimalizácia celkových nákladov a pokút za odchody užívateľov. Práca poskytuje analytické riešenie optimálnych stratégií pre prípady s 1 a 2 užívateľmi v systéme. Pre prípad s viacerými užívateľmi sme použitím posledných poznatkov v oblasti "Multi-armed restless bandit" odvodili novú jednoduchú stratégiu, označovanú ako AJN, pre systémy s povinným, aj bez povinného obsluhovania. Túto stratégiu navrhujeme používať aj v prípadoch s príchodmi užívateľov. Okrem toho poukazujeme aj na dôkladnú štúdiu numerických experimentov pre oba systémy, v ktorých porovnávame AJN indexovú stratégiu s $c\mu$ -stratégiou, o ktorej je dobre známe, že je optimálna pre systém s príchodmi aj bez príchodov užívateľov, avšak v ktorom nie sú zahrnuté odchody užívateľov. Porovnávame ju aj s $c\mu/\theta$ -stratégiou, o ktorej bolo prednedávnom ukázané, že je asymptoticky optimálna stratégia v preťaženom systéme s viacerými servermi. Táto výpočtová štúdia naznačuje, že naša stratégia je takmer vždy lepšia, alebo porovnateľná s ostatnými stratégiami a často býva optimálna.

Kľúčové slová: Markovovské rozhodovacie procesy • Multi-armed restless bandit • Whittlov index • Indexové stratégie • Bellmanova rovnica

Abstract

Novák, Vladimír: Index Policies for Dynamic and Stochastic Problems [Bachelor's thesis].

Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics.

Supervisor: Mgr. Peter Jacko, PhD.

Co-Supervisor: Mgr. Jana Szolgayová, PhD.

Bratislava, 2011

In our work we investigate the Whittle's index policy derivation framework in the Markov decision process environment. We analyze model for the multi-class job scheduling for user with abandonment, with the objective of minimizing the total holding costs and abandonment penalties. The work provides analytical solution of an optimal index rule for the case in which there are 1 or 2 users in the system. For the case with more users we use recent results from the multi-armed restless bandits approach and derive a new simple index rule, denoted by AJN, for the idling and the non-idling system. This index rule is proposed to use also in the system with arrivals. We also report on an exhaustive study of numerical experiments for both systems, in which we compare AJN index rule with the $c\mu$ -rule, which is well-known to be optimal both with and without arrivals but without abandonments, and with $c\mu/\theta$ - rule that was recently shown to be asymptotically optimal in a multi-server system with overload conditions. This computational study suggests that our rule is almost always superior or equivalent to the other rules, and it is often optimal.

Keywords: Markov decision process • Multi-armed restless bandit • Whittle index • Index Policies • Bellman equation

Acknowledgements

I would like to express special thanks to my supervisor Mgr. Peter Jacko, PhD. and to my co-supervisor Mgr. Jana Szolgayová, PhD. , for all the support and guidance they offered me throughout the elaboration of this thesis.

My special thanks goes to Urtzi Ayesta, PhD. and Ina Maria Verloop, PhD. for my supervision, help and motivation for research work during my internships at Basque Center for Applied Mathematics, where this work was mainly carried out thanks to the BCAM internship program.

Last but not least I warmly thank my family and especially my brother Marek for support and love.

Declaration on Word of Honour

I declare on my honour that this thesis was written on my own, with the only help provided by my supervisors and the referred-to literature and sources.

In Bratislava June 1, 2011

.....
Vladimír Novák

Contents

List of Figures	xi
List of Tables	xii
Introduction	1
1 Solution Methods for Stochastic Dynamic Programming Problems	3
1.1 Markov Decision Process Framework	3
1.2 Multi-Armed Restless Bandit Problem	4
1.3 Index Policies and their Designing Methods	5
1.4 Example: Single Restless Bandit Model	5
1.4.1 Example: MDP Formulation of Job Scheduling	6
2 Job Scheduling with Abandonment	8
2.1 Problem Description	8
2.1.1 Variant without Abandonment	9
2.2 MDP Formulation	9
2.2.1 Jobs and Users	9
2.2.2 Alternative Task	10
2.2.3 A Unified Optimization Criterion	10
2.2.4 Optimization Problem	11
2.3 Special Cases	12
2.4 Relaxations and Decomposition	14
2.4.1 Relaxations	15
2.4.2 Decomposition into Single-User Subproblems	16
2.5 Solution	16
2.5.1 Optimal Solution to Single-User Subproblem via Prices	16
2.5.2 Optimal Solution to Relaxations	19
2.5.3 AJN Rule for Original Problem	20
3 Computational Study	21
3.1 Computational Methods	21
3.1.1 Value Iteration and State Space Truncation	21

3.1.2	The Uniformization Approach	22
3.2	Numerical Experiments	23
3.2.1	Idling System	23
3.2.2	Non-Idling System	29
	Conclusion	34
	Resumé	35
	Bibliography	37

List of Figures

3.1	Relative suboptimality gap - Scenario 1	25
3.2	Adjusted relative suboptimality gap - Scenario 1	25
3.3	Relative suboptimality gap - Scenario 2	25
3.4	Adjusted relative suboptimality gap - Scenario 2	25
3.5	Relative suboptimality gap - Scenario 3	26
3.6	Adjusted relative suboptimality gap - Scenario 3	26
3.7	Relative suboptimality gap - Scenario 4	26
3.8	Adjusted relative suboptimality gap - Scenario 4	26
3.9	Relative suboptimality gap - Scenario 5	27
3.10	Adjusted relative suboptimality gap - Scenario 5	27
3.11	Relative suboptimality gap - Scenario 6	27
3.12	Adjusted relative suboptimality gap - Scenario 6	27
3.13	Relative suboptimality gap - Scenario 7	28
3.14	Adjusted relative suboptimality gap - Scenario 7	28
3.15	Relative suboptimality gap - Scenario 8	28
3.16	Adjusted relative suboptimality gap - Scenario 8	28
3.17	Relative suboptimality gap - Non-idling Scenario 1	30
3.18	Adjusted relative suboptimality gap - Non-idling Scenario 1	30
3.19	Relative suboptimality gap - Non-idling Scenario 2	30
3.20	Adjusted relative suboptimality gap - Non-idling Scenario 2	30
3.21	Relative suboptimality gap - Non-idling Scenario 3	31
3.22	Adjusted relative suboptimality gap - Non-idling Scenario 3	31
3.23	Relative suboptimality gap - Non-idling Scenario 4	31
3.24	Adjusted relative suboptimality gap - Non-idling Scenario 4	31
3.25	Relative suboptimality gap - Non-idling Scenario 5	32
3.26	Adjusted relative suboptimality gap - Non-idling Scenario 5	32
3.27	Relative suboptimality gap - Non-idling Scenario 6	32
3.28	Adjusted relative suboptimality gap - Non-idling Scenario 6	32
3.29	Relative suboptimality gap - Non-idling Scenario 7	33
3.30	Adjusted relative suboptimality gap - Non-idling Scenario 7	33
3.31	Relative suboptimality gap - Non-idling Scenario 8	33
3.32	Adjusted relative suboptimality gap - Non-idling Scenario 8	33

List of Tables

3.1	Parameters for all the scenarios in computational experiments for the idling system.	24
3.2	Parameters for all the scenarios in computational experiments for the non-idling system.	29

Introduction

People are confronted with difficult and often too complex problems everyday. Based on our knowledge, intuition and beliefs we have developed several ways how to deal with such problems. One way, that is used very often, is to match alternatives with some priorities and based on them, we have to choose the best alternative. The problem is how to find a most suitable priority to each alternative. Therefore, it is of a great practical interest, to have general methodology for determination of easy-solvable and simple priority rules for complex and intractable problems. Especially, we face these problems when we are sharing resources among multiple users. This arises in diverse areas like marketing, financial economics, engineering systems, medicine, telecommunications, etc.. In each of these cases, there exists some scheduling mechanism that regulates how the resources are shared among competing users. In these cases, it is not always clear what is the best to do. We need to take into account not only efficient use of the available resources and system operating expenses, but also what is the impression obtained by users about the system.

In general, we consider several competitors that are competing for the available resource capacity at the same time. Suppose that independently of other competitors, we can match a value to each competitor. This value is determining the efficiency of attaining a joint goal if we allocate resource capacity to her at a given moment. In addition, reasons mentioned above lead us to deal with an ubiquitous phenomena as abandonment (aka reneging). This happens in multitude of systems, for instance, users in the Internet or users using smartphone's data service may give up a transfer if the connection is slow. User abandonment has a very negative impact from the performance point of view. A user abandonment could imply for system that resources have been wasted by allocating resources to a user that decided to abandon anyway. On the other hand, user who abandons will get a bad impression about the system and for instance, she may want to change a provider of Internet or mobile services.

From mathematical point of view, abandonments motivate the study of queueing-theoretic models, but as a consequence of the complexity, the problem of how to schedule impatient users is not completely understood. In this thesis we aim at solving the problem of scheduling users in a system with abandonments. We formulate

the problem as a discrete time Markov Decision Process (MDP) and we use the recent developments on the theory of Multi-Armed Restless Bandits for deriving a simple implementable scheduling rule (as it was proposed in Whittle (1988)) for multi-class system, based on assigning a priority to every user and serving the user with highest priority.

The thesis is organized as follows. The first Chapter provides theoretical background for the thesis. It consists of an introduction to the topic of Stochastic and Dynamic Resource Allocation, Index Rules derivation, description of a Markov Decision Process (MDP) framework and a view on Multi-Armed Restless Bandits models. In Chapter 2 we make problem description and its formulation as MDP. Moreover, we analytically solve the problem when there are one or two users and derive a new simple scheduling rule for idling and non-idling system. Chapter 3 presents an exhaustive computational study for both systems, suggesting that the proposed rule is nearly-optimal and outperforming several alternatives.

Chapter 1

Solution Methods for Stochastic Dynamic Programming Problems

This Chapter provides an introductory survey of mathematical fields that we will use throughout the whole thesis.

1.1 Markov Decision Process Framework

Imagine a system, where decision maker observes the state of a system at specified points in time. Based on this state, he chooses an action from a set of actions that could be chosen. As a consequence of the chosen action, the decision maker receives a reward (or pays a cost) and the system moves into new state at a subsequent point in time according to a probability distribution determined by the chosen action. (We follow the description given in Puterman (2005)). In this new time point decision maker faces similar problem, but system could be in a different state. In such a *Sequential Decision Model* we assume that decision maker knows: a set of decision epochs \mathcal{T} , a set of system states \mathcal{S} , a set of available actions \mathcal{A} , a set of state and action dependent immediate rewards or costs \mathcal{R} and a set of state and action dependent transition probabilities \mathcal{P} . Decision maker's aim is to maximize the expected reward or to minimize the expected cost over a certain time horizon.

When the set of available actions, rewards, and the transition probabilities depend only on the current state and action and not on previous states and actions we refer to *Markov decision process*. Further we will use terms policy and decision rule. *Policy* tells what to do in any time $t \in \mathcal{T}$. Only if policy is stationary, then it tells what to do in any state. A *decision rule* specifies the action to be chosen at a particular time. A policy is a sequence of decision rules. Implementing a policy generates a sequence of rewards.

From the historical point of view, MDP theory was developed separately for discrete-time and continuous-time models. In this thesis we refer to discrete-time MDPs. Into the bargain, using the uniformization technique we can reformulate important cases of the continuous-time models into the discrete-time models.

Development of discrete-time stream of MDP is mostly connected with Richard Bellman. His method, known as dynamic programming, is based on using the *Principle of Optimality*. The exact formulation of this principle depends on considered optimization type, i.e., we can think about optimization over the finite or infinite horizon etc. For optimization under β -discounted criterion the *Principle of Optimality* is: "At any point in time, an optimal policy must prescribe an action that optimizes the sum of immediate reward and (expected) total reward obtained if an optimal policy is applied from the subsequent point in time on. The mathematical concept associated to the Principle of optimality is the optimality equations of dynamic programming, called the Bellman equations." (Jacko (2010a) page 5, see also Bellman (1957), Bertsekas (2005)). Principle of optimality leads us to the recursive solution method for dynamic programming problems.

As it was mentioned before, the Bellman equation depends on the optimization horizon. We can use the Bellman equation for optimization over *finite horizon*. However, for the optimization over infinite horizon we need to use another type, because terms in the Bellman equation for finite horizon could easily become infinite. In *infinite horizon* case we usually use the β -discounted Bellman equation, where holds $0 < \beta < 1$. This approach is similar to discounting in finance and it is also mainly used in the thesis. Besides this, in infinite horizon case we can also use *time-average criterion*, where we optimize the average expected reward.

Dynamic programming also provides multitude of theoretical results as sufficient conditions for optimality in some cases. Practical contribution of dynamic programming is in the significant decreasing of the problem complexity. Nevertheless, we can still be facing *the curse of dimensionality*, what refers to the problem that the size of dynamic programming formulation is exponentially growing with the size of the model (Jacko (2010a)).

For this purpose, it is often used LP reformulation or relaxation techniques such as Lagrangian relaxation (mentioned later).

1.2 Multi-Armed Restless Bandit Problem

Multi-Armed Restless Bandit Problem is one of the fundamental stochastic resource allocation models. It takes name from the classical slot machine: One-Armed Bandit. In Multi-Armed Bandit Problem we consider multitude of levers and each lever provides a specific reward. Gambler can pull exactly one lever at a time. The gambler's goal is to maximize the sum of rewards earned through a sequence of lever pulls (Gittins (1989)). In practice, Multi-Armed Bandit Problem is used for allocating resources among the competing projects, about which we have not got full information. Such cases occur, for instance, in managing research projects in large organizations.

Whittle (see Whittle (1988)) proposed an extension of the Multi-Armed Bandit Problem, denoted as Multi-Armed Restless Bandit Problem. In this version, bandit admits evolution and reward even if it is not played and we can allocate the scarce resource parallelly to a fixed number of bandits. Due to that, we are able to describe a bigger set of real problems by multi-armed restless bandit problem. However, in this augmented version occur problems with tractability. The multi-armed restless bandit

problem is known as extremely difficult optimization problem and proven PSPACE-hard (Papadimitriou and Tsitsiklis (1999)), that is, its complexity grows exponentially in time and in memory requirements. Therefore, recent research focuses on designing of tractable heuristic rules.

1.3 Index Policies and their Designing Methods

In practice, due to the curse of dimensionality there are often used solutions obtained by ad hoc techniques. Many people propose ad hoc solution also in the case when it is needed to find only a nearly-optimal solution. We can observe several applications of such ad hoc techniques in routing of mobile connections.

Gittins, in the early 70s, presented a series of papers in which he developed feature of multi-armed (classic) bandit problem which is optimality of the *index priority rule*. (Whittle (1980)). We match each of the bandit with a dynamic allocation index, and then apply the index priority policy, defined: "Assign the scarce resource to a bandit of highest current index value." (Jacko (2010a), page 12). This index proposed by Gittins is now well-known as *Gittins index* and a solution of the classic multi-armed bandit problem is known as *Gittins index policy* (see also Gittins (1979)). One of the most remarkable contributions of this index is that it can be computed for every bandit separately, independently on the others, and therefore reducing the multi-dimensional problem to unidimensional.

Further we will use approach provided by Whittle (Whittle (1988)). For solving multi-armed restless bandit problem, he used Lagrangian relaxation method. Before employing the Lagrangian multiplier he proposed to relax the problem by replacing the family of sample-path constraints by unique one. In other words, we replace the constraint of playing fixed number of bandits in every period, by playing the required number of bandits on average. Then, the Lagrangian relaxation results in an unconstrained problem of optimizing the sum of independent reward processes. Thus we can decompose multi-armed problem and simplify the solution procedure. Whittle also proposed an index recovering Gittins index and modified the index priority policy to: "Assign the scarce resource to bandits of highest current index values" (Whittle (1988)). These indices often have an economic meaning. Due to that we often use terms indexes and prices interchangeably.

1.4 Example: Single Restless Bandit Model

In this section following Jacko (2010a) we want to demonstrate the modeling power of above-mentioned frameworks and methods. We will describe them in a more precise way.

Consider the time slotted into time epoch $t \in \mathcal{T} := \{0, 1, 2, \dots\}$. The time epoch t corresponds to the beginning of the time period t . We can choose to work or not to work on a given project. Project is a general work-reward restless bandit. We denote by $\mathcal{A} := \{0, 1\}$ the action space, where 0 corresponds to not working (idling) and 1 corresponds to working. The action space is same for all projects. Each project from

set \mathcal{K} can be modeled independently of the others as tuple:

$$(\mathcal{N}_k, (\mathbf{W}_k^a)_{a \in \mathcal{A}}, (\mathbf{C}_k^a)_{a \in \mathcal{A}}, (\mathbf{P}_k^a)_{a \in \mathcal{A}}),$$

where

- \mathcal{N}_k is the *state space*, i.e., a finite set of possible states project k can occupy
- $\mathbf{W}_k^a := (W_{k,n}^a)_{n \in \mathcal{N}_k}$, where $W_{k,n}^a$ is the (expected) one-period *work* required by project k at state n under action a ;
- $\mathbf{R}_k^a := (R_{k,n}^a)_{n \in \mathcal{N}_k}$, where $R_{k,n}^a$ is the expected one-period *reward* earned by project k at state n under action a
- $\mathbf{P}_k^a := (p_{k,n,m}^a)_{n,m \in \mathcal{N}_k}$ is the project- k stationary one-period *state-transition probability matrix* if action a is decided at the beginning of a period, i.e., $p_{k,n,m}^a$ is the probability of moving to state m from state n under action a ;

The dynamics of the project k is captured by the *state process* $X_k(\cdot)$ and the *action process* $a_k(\cdot)$, in each time epoch t . Naturally, we require that the expected one-period work is nonnegative, i.e., $0 \leq \mathbf{W}_k^a$.

Sometimes it may be more appropriate to consider $\mathbf{C}_k^a := -\mathbf{R}_k^a$, the expected one-period cost paid by project k at state n , under action a .

1.4.1 Example: MDP Formulation of Job Scheduling

Consider K jobs waiting at the beginning (i.e., at time epoch $t = 0$) for service at a server that can serve one job at every time period. Let $0 < \mu_k < 1$ be the probability that the service of job k is completed within one period and let $0 < c_k$ be the holding cost per period incurred for job k waiting. These jobs can be viewed as competing projects and while there are at least two jobs waiting, one must decide to which job the server should be allocated. Suppose that the server is preemptive, i.e., the service of a job can be interrupted at any time epoch even if not completed.

We will define job k by tuple (Jacko (2009)), as it was proposed in previous subsection.

- $\mathcal{N}_k := \{\text{completed}, \text{waiting}\}$;
- expected one-period work

$$\begin{aligned} \mathbf{W}_{k,\text{completed}}^1 &:= 1, & \mathbf{W}_{k,\text{waiting}}^1 &:= 1, \\ \mathbf{W}_{k,\text{completed}}^0 &:= 0, & \mathbf{W}_{k,\text{waiting}}^0 &:= 0; \end{aligned}$$

- expected one-period reward

$$\begin{aligned} \mathbf{R}_{k,completed}^1 &:= 0, & \mathbf{R}_{k,waiting}^1 &:= -c_k(1 - \mu_k) - 0\mu_k, \\ \mathbf{R}_{k,completed}^0 &:= 0, & \mathbf{R}_{k,waiting}^0 &:= -c_k; \end{aligned}$$

- one-period state-transition probability matrix if serving a job,

$$\mathbf{P}_k^1 := \begin{array}{c} \text{completed} \\ \text{waiting} \end{array} \begin{array}{cc} \text{completed} & \text{waiting} \\ \left(\begin{array}{cc} 1 & 0 \\ \mu_k & 1 - \mu_k \end{array} \right), \end{array}$$

and if not being served

$$\mathbf{P}_k^0 := \begin{array}{c} \text{completed} \\ \text{waiting} \end{array} \begin{array}{cc} \text{completed} & \text{waiting} \\ \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right). \end{array}$$

In the following section we apply this approach and notation to a more complicated problem. In mentioned example projects cannot abandon, and we are talking about *work-conserving* systems, what means that the server cannot idle if there are jobs waiting, so systems are *non-idling*. In *idling* systems, the server can idle even if there are jobs waiting.

In systems with abandonments, however, this requirement should not be imposed, because it may be optimal to idle the server even if there are jobs waiting.

The Whittle index and the Gittins index for this job is $\frac{c_k \mu_k}{1 - \beta}$ (see Jacko (2010b)), so the index policy is called the $c\mu$ -rule: Work on job with the highest $c_k \mu_k$. This rule was known to be optimal since Smith (1956) without arrivals and Cox and Smith (1961) with arrivals but proofs do not easily extend to the problem with abandonments.

Job Scheduling with Abandonment

In this chapter we present the multi-class job scheduling problem, in which we allow for abandonment due to users mobility or impatience. We will describe the problem and formulate it as a Markov Decision Process. Further, we introduce the relaxed formulation of the problem. The main emphasis of this chapter (based on my joint work with Urtzi Ayesta and Peter Jacko: (Ayesta et al. (2011))) is put on analytical solving of the original problem in some special cases, but the biggest effort is given on the analytical solving of the relaxed problem and thus derivation of the heuristic rule for original stochastic optimization formulation.

2.1 Problem Description

Consider $K - 1$ jobs waiting for service of a server that can serve one job at a time. The service of job k is completed (if being served) with probability $\mu_k > 0$ and the probability of her abandonment (if not being served) is $\theta_k \geq 0$. We assume that the user in service cannot abandon. Thus, the jobs (i.e., users) are assumed independent of each other.

Let $c_k > 0$ be the holding cost incurred for user k waiting in the queue. Further, let $d_k \geq 0$ be the abandonment penalty incurred for user k if she abandons the system without having her job completed. If the server is allocated to a user whose job has already been completed, then no service occurs.

We incorporate the following parameter that makes the problem extensively flexible to incorporate additional conditions or options, and turns out to be crucial for creating not work-conserving system. It is allowed to allocate the server to an alternative task (such as idling, battery recharging or service maintenance), for which we obtain an *alternative-task reward* κ . For instance, the role of this alternative task with a positive κ could be to turn off the server allocation when all the users have too high abandonment rates. On the other hand, by setting this parameter to a negative value we may force the server to be non-idling (whenever there are waiting jobs), or it can be simply set to zero narrowing the focus to the classic problem.

The joint goal is to minimize the expected aggregate holding and abandonment

costs minus the alternative-task reward, over an infinite horizon. The server is assumed to be preemptive (i.e., the service of a job or the alternative task can be interrupted at any moment even if not completed). Thus, the server continuously decides to which user (if any) it should be allocated.

2.1.1 Variant without Abandonment

If $\theta_k = 0$ for all k (i.e., there is no abandonment) and $\kappa = 0$, then this problem recovers the classic job scheduling problem considered in Cox and Smith (1961); Smith (1956); Fife (1965); Buyukkoc et al. (1985), for which the following greedy rule attains such a goal:

Rule 1 ($c\mu$ -rule). *Allocate the server to any waiting job of the non-empty class with the highest value $c_k\mu_k$.*

This quantity measures the expected savings on holding costs, or the efficiency of attaining the goal, if user of class k is served. Thus, the $c\mu$ -rule allocates the server to the user who contributes most efficiently to minimization of the expected aggregate holding cost.

2.2 MDP Formulation

Since the $c\mu$ -rule is optimal both under general arrival distribution and under no arrivals, and both in the continuous-time and the discrete-time model, we set out to analyze the discrete-time model without arrivals, in order to obtain a rule accounting for abandonment whose performance in the continuous-time model with arrivals we later evaluate by means of numerical experiments. We set the model in the framework of the dynamic and stochastic resource allocation problem and follow the approach to design prices as described in Jacko (2009).

Consider the time slotted into epochs $t \in \mathcal{T} := \{0, 1, 2, \dots\}$ at which decisions can be made. The time epoch t corresponds to the beginning of the time period t . Suppose that at $t = 0$ there are $K - 1 \geq 1$ users awaiting service from the server that at each epoch chooses (at most) one of the users to serve. If no user is chosen, then the server is allocated to the alternative task, i.e., there are K competing options, labeled by $k \in \mathcal{K}$. Thus, the server is allocated to exactly one option at a time.

2.2.1 Jobs and Users

Every user $k = 1, 2, \dots, K - 1$ can be allocated either zero or full capacity of the server. We denote by $\mathcal{A} := \{0, 1\}$ the *action space*, i.e., the set of allowable levels of capacity allocation. Here, action 0 means allocating zero capacity (i.e., “not serving”), and action 1 means allocating full capacity (i.e., “serving”). This action space is the same for every user k .

Each job/user k is defined independently of other jobs/users as the tuple

$$(\mathcal{N}_k, (\mathbf{W}_k^a)_{a \in \mathcal{A}}, (\mathbf{C}_k^a)_{a \in \mathcal{A}}, (\mathbf{P}_k^a)_{a \in \mathcal{A}}),$$

where

- $\mathcal{N}_k := \{0, 1\}$ is the *state space*, where state 0 represents a job already completed or abandoned, and state 1 means that the job is uncompleted and not abandoned;
- $\mathbf{W}_k^a := (W_{k,n}^a)_{n \in \mathcal{N}_k}$, where $W_{k,n}^a$ is the (expected) one-period capacity consumption, or *work* required by user k at state n if action a is decided at the beginning of a period; in particular, for any $n \in \mathcal{N}_k$,

$$W_{k,n}^1 := 1, \quad W_{k,n}^0 := 0;$$

- $\mathbf{C}_k^a := (C_{k,n}^a)_{n \in \mathcal{N}_k}$, where $C_{k,n}^a$ is the expected one-period *cost* paid by user k at state n if action a is decided at the beginning of a period; in particular,

$$\begin{aligned} C_{k,0}^1 &:= 0, & C_{k,1}^1 &:= c_k \cdot (1 - \mu_k) + 0 \cdot \mu_k, \\ C_{k,0}^0 &:= 0, & C_{k,1}^0 &:= c_k \cdot (1 - \theta_k) + d_k \cdot \theta_k; \end{aligned}$$

- $\mathbf{P}_k^a := (p_{k,n,m}^a)_{n,m \in \mathcal{N}_k}$ is the user- k stationary one-period *state-transition probability matrix* if action a is decided at the beginning of a period, i.e., $p_{k,n,m}^a$ is the probability of moving to state m from state n under action a ; in particular, we have

$$\mathbf{P}_k^1 := \begin{matrix} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{pmatrix} 1 & 0 \\ \mu_k & 1 - \mu_k \end{pmatrix} \end{matrix}, \quad \mathbf{P}_k^0 := \begin{matrix} & \begin{matrix} 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \begin{pmatrix} 1 & 0 \\ \theta_k & 1 - \theta_k \end{pmatrix} \end{matrix}.$$

The dynamics of user k is thus captured by the *state process* $X_k(\cdot)$ and the *action process* $a_k(\cdot)$, which correspond to state $X_k(t) \in \mathcal{N}_k$ and action $a_k(t) \in \mathcal{A}$ at all time epochs $t \in \mathcal{T}$. As a result of deciding action $a_k(t)$ in state $X_k(t)$ at time epoch t , the user k consumes the allocated capacity, earns the reward, and evolves its state for the time epoch $t + 1$.

Note that we have the same action space \mathcal{A} available at every state, which assures a technically useful property that $\mathbf{W}_k^a, \mathbf{C}_k^a, \mathbf{P}_k^a$ are defined in the same dimensions under any $a \in \mathcal{A}$. Note also that state 0 is absorbing.

2.2.2 Alternative Task

We model the alternative task as a *static κ -user* with a single state 0 and with reward κ if served. i.e., such a user $k = K$ is defined by $\mathcal{N}_K := \{0\}, W_{K,0}^a := a, C_{K,0}^a := -\kappa a, p_{K,0,0}^a := 1$ for all $a \in \mathcal{A}$.

2.2.3 A Unified Optimization Criterion

Before describing the problem we first define an averaging operator that will allow us to discuss the infinite-horizon problem under the traditional β -discounted criterion and the time-average criterion in parallel. Let $\Pi_{X,a}$ be the set of all the policies that for each time epoch t decide (possibly *randomized*) action $a(t)$ based only

on the state-process history $X(0), X(1), \dots, X(t)$ and on the action-process history $a(0), a(1), \dots, a(t-1)$ (i.e., *non-anticipative*). Let \mathbb{E}_τ^π denote the expectation over the state process $X(\cdot)$ and over the action process $a(\cdot)$, conditioned on the state-process history $X(0), X(1), \dots, X(\tau)$ and on policy π .

Consider any expected one-period quantity $Q_{X(t)}^{a(t)}$ that depends on state $X(t)$ and on action $a(t)$ at any time epoch t . For any policy $\pi \in \Pi_{X,a}$, any initial time epoch $\tau \in \mathcal{T}$, and any *discount factor* $0 \leq \beta \leq 1$ we define the infinite-horizon β -average quantity as¹

$$\mathbb{B}_\tau^\pi \left[Q_{X(\cdot)}^{a(\cdot)}, \beta, \infty \right] := \lim_{T \rightarrow \infty} \frac{\sum_{t=\tau}^{T-1} \beta^{t-\tau} \mathbb{E}_\tau^\pi \left[Q_{X(t)}^{a(t)} \right]}{\sum_{t=\tau}^{T-1} \beta^{t-\tau}}. \quad (2.1)$$

The β -average quantity recovers the traditionally considered quantities in the following three cases:

- *expected time-average quantity* when $\beta = 1$.
- *expected total β -discounted quantity*, scaled by constant $1 - \beta$, when $0 < \beta < 1$;
- *myopic quantity* when $\beta = 0$.

Thus, when $\beta = 1$, the problem is formulated under the *time-average criterion*, whereas when $0 < \beta < 1$ the problem is considered under the *β -discounted criterion*. The remaining case when $\beta = 0$ reduces to a static problem and hence is considered in order to define a *myopic policy*. In the following we consider the discount factor β to be fixed and the horizon to be infinite, therefore we omit them in the notation and write briefly $\mathbb{B}_\tau^\pi \left[Q_{X(\cdot)}^{a(\cdot)} \right]$.

2.2.4 Optimization Problem

We now describe in more detail the problem we consider. Let $\Pi_{\mathbf{X}, \mathbf{a}}$ be the space of randomized and non-anticipative policies depending on the joint state-process $\mathbf{X}(\cdot) := (X_k(\cdot))_{k \in \mathcal{K}}$ and deciding the joint action-process $\mathbf{a}(\cdot) := (a_k(\cdot))_{k \in \mathcal{K}}$, i.e., $\Pi_{\mathbf{X}, \mathbf{a}}$ is the *joint policy space*.

For any discount factor β , the problem is to find a joint policy $\boldsymbol{\pi}$ maximizing the *objective* given by the β -average aggregate reward starting from the initial time epoch 0 subject to the family of *sample path* allocation constraints, i.e.,

$$\begin{aligned} \min_{\boldsymbol{\pi} \in \Pi_{\mathbf{X}, \mathbf{a}}} \mathbb{B}_0^\pi \left[\sum_{k \in \mathcal{K}} C_{k, X_k(\cdot)}^{a_k(\cdot)} \right] & \quad (\text{P}) \\ \text{subject to } \mathbb{E}_t^\pi \left[\sum_{k \in \mathcal{K}} a_k(t) \right] & = 1, \text{ for all } t \in \mathcal{T} \end{aligned}$$

¹For definiteness, we consider $\beta^0 = 1$ for $\beta = 0$.

Note that the constraint could equivalently be expressed in words as that for all $t \in \mathcal{T}$: $\sum_{k \in \mathcal{K}} a_k(t) = 1$ under policy π and for any possible joint state-process history $\mathbf{X}(0), \mathbf{X}(1), \dots, \mathbf{X}(t)$.

2.3 Special Cases

Problem (P) is hard to solve in the whole generality, but we have identified special cases that admit an analytical solution of the Bellman equation, summarized in this section. Surprisingly, to the best of our knowledge no one has optimally solved the cases of one or two users before.

In the case of a single user (1U) competing with the alternative task, we introduce the following index:

$$\nu_k^{1U} := c_k(\mu_k - \theta_k) + d_k\theta_k(1 - \beta + \beta\mu_k). \quad (2.2)$$

Proposition 2.3.1. *The following holds for problem (P) with $K = 2$ and the alternative task reward $\kappa = 0$:*

1. If $\nu_1^{1U} \geq 0$, then it is optimal to serve user;
2. If $\nu_1^{1U} \leq 0$, then it is optimal to allocate the server to the alternative task ($k = 2$).

Proof. Let us denote the optimal value function by $V^*(n_1, n_2)$, where n_1 refers to number of users in a class 1 and n_2 refers to number of users in a class 2. When $\kappa = 0$, empty class 2 can represent the alternative task. Then assume that we have two classes and in each class there is nobody. In such a case, using the Bellman equation it is straightforward to obtain that

$$V^*(0, 0) = 0.$$

If there is a player in the class one and nobody in the class two then the Bellman's equation is:

$$V^*(1, 0) = \min\{C_{1,1}^1 + \beta p_{1,1,1}^1 V^*(1, 0) + \beta p_{1,1,0}^1 V^*(0, 0); \\ C_{1,1}^0 + \beta p_{1,1,1}^0 V^*(1, 0) + \beta p_{1,1,0}^0 V^*(0, 0)\},$$

where V^* is a optimal value function for the consequent number of users in the classes. The first term refers to serving the user and the second term to idling. After plugging the definitions into these two terms, we obtain:

$$V^*(1, 0) = \min\{\mu_1(\beta V^*(0, 0)) + (1 - \mu_1)(c_1 + \beta V^*(1, 0)); \\ \theta_1(d_1 + \beta V^*(0, 0)) + (1 - \theta_1)(c_1 + \beta V^*(1, 0))\}. \quad (2.3)$$

We solve the Bellman's equation and evaluate the value function assuming that serving is optimal:

$$V^*(1, 0) = \frac{c_1(1 - \mu_1)}{1 - \beta + \mu_1\beta}.$$

From (2.3), we obtain that serving the user is better than or equivalent to idling if

$$(-\beta V^*(1, 0) + c_1)(\mu_1 - \theta_1) + d_1\theta_1 \geq 0.$$

Putting the last equality together with this inequality then yields to:

$$c_k(\mu_k - \theta_k) + d_k\theta_k(1 - \beta + \beta\mu_k) \geq 0$$

what is the stated result (1). All steps are equivalent, therefore it holds also from the reverse direction. Claim (2) is obtained analogously. The only difference is that in a case of not serving inequality is reverse and the value function is equal to:

$$V^*(1, 0) = \frac{\theta_1(d_1 - c_1) + c_1}{1 + \theta_1\beta - \beta}$$

□

In the case of two users (2U) competing among themselves, due to the technical complexity of the problem we concentrate only on an undiscounted case ($\beta = 1$). We introduce the following index for users $k = 1, 2$ with respect to the other user:

$$\nu_k^{2U} := \frac{c_k(\mu_k - \theta_k) + d_k\theta_k\mu_k}{\mu_k[1 - (1 - \mu_{3-k})(1 - \theta_k)]}. \quad (2.4)$$

Notice that it depends on the other user's parameters, but only through the service rate μ_{3-k} .

Proposition 2.3.2. *Suppose that $\nu_k^{1U} \geq 0$ for $k = 1, 2$. The following holds for problem (P) with $K = 3$, $\beta = 1$, and the alternative task reward $\kappa = 0$:*

1. *If $\nu_1^{2U} \geq \nu_2^{2U}$, then it is optimal to serve user 1;*
2. *If $\nu_1^{2U} \leq \nu_2^{2U}$, then it is optimal to serve user 2.*
3. *It is optimal to allocate the server to the alternative task if and only if $\nu_1^{1U} = \nu_2^{1U} = 0$*

Proof. The proof goes along the same lines as the one above.

When we have two classes and in each there is a player, the Bellman's equation is:

$$\begin{aligned}
V^*(1, 1) = \min\{ & \mu_1\theta_2(d_2 + \beta V^*(0, 0)) + \mu_1(1 - \theta_2)(c_2 + \beta V^*(0, 1)) + \\
& + (1 - \mu_1)\theta_2(c_1 + d_2 + \beta V^*(1, 0)) + \\
& + (1 - \mu_1)(1 - \theta_2)(c_1 + c_2 + \beta V^*(1, 1)); \\
& \theta_1\mu_2(d_1 + \beta V^*(0, 0)) + \theta_1(1 - \mu_2)(d_1 + c_2 + \beta V^*(0, 1)) + \\
& + (1 - \theta_1)\mu_2(c_1 + \beta V^*(0, 1)) + \\
& + (1 - \theta_1)(1 - \mu_2)(c_1 + c_2 + \beta V^*(1, 1)); 0\}
\end{aligned}$$

Using the Bellman's equation straightforward we derive simplified equation for $V^*(1, 1)$:

$$\begin{aligned}
V^*(1, 1) = \min\{ & (1 - \mu_1)(1 - \theta_2)V^*(1, 1) + \theta_2d_2 + c_2\frac{1 - \theta_2}{\mu_2}(\mu_1 + \mu_2 - \mu_1\mu_2) + \\
& + c_1\frac{1 - \mu_1}{\mu_1}(\theta_2 + \mu_1 - \theta_2\mu_1); (1 - \mu_2)(1 - \theta_1)V^*(1, 1) + \theta_1d_1 + \\
& + c_1\frac{1 - \theta_1}{\mu_1}(\mu_2 + \mu_1 - \mu_1\mu_2) + c_2\frac{1 - \mu_2}{\mu_2}(\theta_1 + \mu_2 - \theta_1\mu_2); 0\}.
\end{aligned}$$

From this equation we can derive condition, when it is optimal to serve class 1. In such a case, first term is bigger than a second term. Easily we obtain a condition for serving class 1.

$$\frac{d_1\theta_1\mu_1 + c_1(\mu_1 - \theta_1)}{\mu_1[1 - (1 - \mu_2)(1 - \theta_1)]} > \frac{d_2\theta_2\mu_2 + c_2(\mu_2 - \theta_2)}{\mu_2[1 - (1 - \mu_1)(1 - \theta_2)]}$$

In the case that optimal is to serve class 2, condition is similar. Only the second term is bigger than the first term. Thus we proved statements (1) and (2) and we obtained index for two users:

$$\nu_k^{2U} := \frac{c_k(\mu_k - \theta_k) + d_k\theta_k\mu_k}{\mu_k[1 - (1 - \mu_{3-k})(1 - \theta_k)]}.$$

Claim (3) is obtained by comparing the first(second) term with the third term. \square

Corollary 2.3.3. *This index satisfies the following inequality:*

$$\frac{d_k\theta_k\mu_k + c_k(\mu_k - \theta_k)}{\mu_k} \leq \frac{d_k\theta_k\mu_k + c_k(\mu_k - \theta_k)}{\mu_k[1 - (1 - \mu_{3-k})(1 - \theta_k)]} \leq \frac{d_k\theta_k\mu_k + c_k(\mu_k - \theta_k)}{\mu_k\theta_k}$$

2.4 Relaxations and Decomposition

For larger values of K the problem is most likely analytically intractable, and therefore we approach it in an alternative way.

2.4.1 Relaxations

For notational reasons we will use the fact that $W_{k, X_k(t)}^{a_k(t)} = a_k(t)$ (cf. definitions in (2.2)) and instead of the constraints in (P) we will consider the sample path *consumption* constraints $\mathbb{E}_t^\pi \left[\sum_{k \in \mathcal{K}} W_{k, X_k(t)}^{a_k(t)} \right] = 1$, for all $t \in \mathcal{T}$. These constraints imply the *epoch- t expected* consumption constraints,

$$\mathbb{E}_0^\pi \left[\sum_{k \in \mathcal{K}} W_{k, X_k(t)}^{a_k(t)} \right] = 1, \text{ for all } t \in \mathcal{T} \quad (2.5)$$

requiring that the capacity be fully allocated at every time epoch if conditioned on $\mathbf{X}(0)$ only. Finally, we may require this constraint to hold only on β -average, as the *β -average capacity consumption constraint*

$$\mathbb{B}_0^\pi \left[\sum_{k \in \mathcal{K}} W_{k, X_k(\cdot)}^{a_k(\cdot)} \right] = \mathbb{B}_0^\pi [1]. \quad (2.6)$$

Using $\mathbb{B}_0^\pi [1] = 1$, we obtain the following *relaxation* of problem (P),

$$\begin{aligned} \min_{\pi \in \Pi_{\mathbf{X}, \mathbf{a}}} \mathbb{B}_0^\pi \left[\sum_{k \in \mathcal{K}} C_{k, X_k(\cdot)}^{a_k(\cdot)} \right] & \quad (\text{P}^W) \\ \text{subject to } \mathbb{B}_0^\pi \left[\sum_{k \in \mathcal{K}} W_{k, X_k(\cdot)}^{a_k(\cdot)} \right] & = 1. \end{aligned}$$

This relaxation was introduced in Whittle (1988). The above arguments thus provide a proof of the following result.

Proposition 2.4.1. *Problem (P^W) is a relaxation of problem (P).*

The *Whittle relaxation* (P^W) can be approached by traditional Lagrangian methods, introducing a Lagrangian parameter, say ν , to dualize the constraint, obtaining thus the following Lagrangian relaxation,

$$\min_{\pi \in \Pi_{\mathbf{X}, \mathbf{a}}} \mathbb{B}_0^\pi \left[\sum_{k \in \mathcal{K}} C_{k, X_k(\cdot)}^{a_k(\cdot)} + \nu \sum_{k \in \mathcal{K}} W_{k, X_k(\cdot)}^{a_k(\cdot)} \right] - \nu. \quad (\text{P}_\nu^L)$$

The classic Lagrangian result says the following:

Proposition 2.4.2. *For any ν , problem (P _{ν} ^L) is a relaxation of problem (P^W), and further a relaxation of problem (P).*

Note finally that by the definition of relaxation, (P _{ν} ^L) for every ν provides an upper bound for the optimal value of both problem (P^W) and problem (P).

2.4.2 Decomposition into Single-User Subproblems

We now set out to decompose the optimization problem (P_ν^L) as it is standard for Lagrangian relaxations, considering ν as a parameter. Notice that any joint policy $\boldsymbol{\pi} \in \Pi_{\mathbf{X},a}$ defines a set of single-user policies $\tilde{\pi}_k$ for all $k \in \mathcal{K}$, where $\tilde{\pi}_k$ is a randomized and non-anticipative policy depending on the *joint* state-process $\mathbf{X}(\cdot)$ and deciding the *user- k* action-process $a_k(\cdot)$. We will write $\tilde{\pi}_k \in \Pi_{\mathbf{X},a_k}$. We will therefore study the user- k subproblem

$$\min_{\tilde{\pi}_k \in \Pi_{\mathbf{X},a_k}} \mathbb{B}_0^{\tilde{\pi}_k} \left[C_{k,X_k(\cdot)}^{a_k(\cdot)} + \nu W_{k,X_k(\cdot)}^{a_k(\cdot)} \right]. \quad (2.7)$$

2.5 Solution

In this section we will identify a set of optimal policies $\tilde{\pi}_k^*$ to (2.7) for all users k , and using them we will construct a joint policy $\boldsymbol{\pi}$ feasible though not necessarily optimal for problem (P).

2.5.1 Optimal Solution to Single-User Subproblem via Prices

Problem (2.7) falls into the framework of *restless bandits* and can be optimally solved by assigning a set of prices $\nu_{k,n}$ to each state $n \in \mathcal{N}_k$ under certain conditions (Niño-Mora (2007)).

Let us denote for user $k \leq K - 1$, $\nu_{k,0}^{AJN} := 0$, and

$$\nu_{k,1}^{AJN} := \begin{cases} \frac{c_k(\mu_k - \theta_k) + d_k\theta_k(1 - \beta + \beta\mu_k)}{1 - \beta + \beta\theta_k}, & \text{if } \nu_k^{1U} \geq 0 \\ \frac{c_k(\mu_k - \theta_k) + d_k\theta_k(1 - \beta + \beta\mu_k)}{1 - \beta + \beta\mu_k}, & \text{if } \nu_k^{1U} < 0 \end{cases} \quad (2.8)$$

where

$$\nu_k^{1U} := c_k(\mu_k - \theta_k) + d_k\theta_k(1 - \beta + \beta\mu_k).$$

and for the alternative task $k = K$, $\nu_{K,0}^{AJN} := \kappa$. Then we can prove the following result.

Proposition 2.5.1. *For problem (2.7) and $k \leq K - 1$, the following holds:*

1. if $\nu \leq \nu_{k,1}^{AJN}$, then it is optimal to serve waiting user k ;
2. if $\nu \geq \nu_{k,1}^{AJN}$, then it is optimal not to serve waiting user k ;
3. if $\nu \leq \nu_{k,0}^{AJN}$, then it is optimal to serve job k when it is already completed or abandoned;
4. if $\nu \geq \nu_{k,0}^{AJN}$, then it is optimal not to serve job k when it is already completed or abandoned;

5. if $\nu \leq \nu_{K,0}^{AJN}$, then it is optimal to serve to the alternative task K ;
6. if $\nu \geq \nu_{K,0}^{AJN}$, then it is optimal not to serve the alternative task K .

Proof. The proof of this proposition is based on establishing indexability of the problem and computing the index values following the survey Niño-Mora (2007). For following the survey Niño-Mora (2007) we will rewrite our minimalization problem as a maximalization problem. Indexability is in fact equivalent to existence of the quantities with stated properties, and is valid because any binary-state MDP is indexable.

Let us denote the optimal value function by $V_{k,n}^*$ for user k and state n and by $\mathbf{R}_k^a := (R_{k,n}^a)_{n \in \mathcal{N}_k} = -\mathbf{C}_k^a := -(C_{k,n}^a)_{n \in \mathcal{N}_k}$, where $R_{k,n}^a$ is the expected one-period reward earned by user k at state n if action a is decided at the beginning of a period. The Bellman equation for state 1 and user $k \leq K - 1$, after plugging the definitions of the action-dependent parameters for a state is:

$$V_{k,1}^* = -\max\{R_{k,1}^1 - \nu W_{k,1}^1 - \beta[\mu_k V_{k,0} + (1 - \mu_k)V_{k,1}]; \\ R_{k,1}^0 - \nu W_{k,1}^0 - \beta[\theta_k V_{k,0}^* + (1 - \theta_k)V_{k,1}^*]\}, \quad (2.9)$$

After plugging the formulas for expected rewards and expected one-period capacity consumption, we obtain:

$$V_{k,1}^* = -(\max\{-c_k(1 - \mu_k) - \nu - \beta[\mu_k V_{k,0} + (1 - \mu_k)V_{k,1}]; \\ +c_k(1 - \theta_k) + d_k\theta_k + \beta[\theta_k V_{k,0}^* + (1 - \theta_k)V_{k,1}^*]; 0\} \\ -c_k(1 - \theta_k) - d_k\theta_k - \beta[\theta_k V_{k,0}^* + (1 - \theta_k)V_{k,1}^*]), \quad (2.10)$$

where the first term in the curly brackets corresponds to allocating a full capacity and the second term corresponds to allocating a zero capacity.

The Bellman equation for $V_{k,0}^*$ is:

$$V_{k,0}^* = -\max\{R_{k,0}^1 - \nu W_{k,0}^1 - \beta V_{k,0}; R_{k,0}^0 - \nu W_{k,0}^0 - \beta V_{k,0}\}. \quad (2.11)$$

Further for $\nu \geq 0$, using the Bellman equation above it is straightforward to obtain that $V_{k,0}^* = 0$, under assumption that $\beta \neq 0$. Analogous for $\nu < 0$, we obtain:

$$V_{k,0}^* = \frac{\nu}{1 - \beta}.$$

In the following two statements it is supposed to hold $\nu \geq 0$.

In the statement (1) we want to show that under stated assumptions it is optimal to serve waiting user k . Therefore, we derive value of $V_{k,1}^*$.

When we choose to allocate full capacity consumption, formula for $V_{k,1}^*$, is:

$$V_{k,1}^* = c_k(1 - \mu_k) + \nu + \beta[\mu_k V_{k,0}^* + (1 - \mu_k)V_{k,1}^*].$$

Straightforward we obtain:

$$V_{k,1}^* = \frac{c_k(1 - \mu_k) + \nu}{(1 - \beta + \beta\mu_k)}.$$

We want to show that under assumption:

$$\frac{c_k(\mu_k - \theta_k) + d_k\theta_k(1 - \beta + \beta\mu_k)}{1 - \beta + \beta\theta_k} \geq \nu,$$

what is:

$$\nu_{k,1}^{AJN} \geq \nu,$$

it is optimal to user waiting user k . Therefore we substitute terms $V_{k,0}^*$ and $V_{k,1}^*$ for serving class k in equation (2.10), and thus we obtained condition for serving class k :

$$c_k(\mu_k - \theta_k) + d_k\theta_k + \beta \left(\frac{c_k(1 - \mu_k) + \nu}{1 - \beta + \beta\mu_k} \right) (\mu_k - \theta_k) \geq \nu$$

This condition is satisfied under our assumptions, because this inequality and our assumptions are same.

Analogous to above we can show that if

$$\nu_{k,1}^{AJN} \leq \nu,$$

then the action corresponding to allocating zero capacity consumption is bigger than the action corresponding to allocating full capacity consumption. In other words, it is optimal not to serve waiting user k .

We can prove statements (1) and (2) also for the cases when $\nu < 0$, i.e. $\nu_k^{AJN} < 0$. All steps will be similar. If $\nu < 0$, then we have $V_{k,0}^* = \frac{\nu}{(1-\beta)}$.

If serving is optimal, from the equation (2.10), we have

$$V_{k,1}^* = \frac{c_k(1 - \beta_k) + \nu + \beta\mu_k \frac{\nu}{(1-\beta)}}{1 - \beta + \beta\mu_k}. \quad (2.12)$$

Next, we substitute terms $V_{k,0}^*$ and $V_{k,1}^1$ in the equation (2.10). We want to derive conditions, when it is optimal to serve class 1 and when it is optimal to serve class 2. For case, when it is optimal to serve class 1, i.e. first term in (2.10) is bigger or equal to the second. Directly from that inequality we obtain:

$$\nu \leq d_k\theta_k + (\mu_k - \theta) \left[c_k + \beta_k \frac{c_k(1 - \mu_k) + \nu + \beta\mu_k \frac{\nu}{(1-\beta)}}{1 - \beta + \beta\mu_k} - \beta \frac{\nu}{(1 - \beta)} \right].$$

We can derive from that:

$$\nu \leq \frac{c_k(\mu_k - \theta_k) + d_k\theta_k(1 - \beta + \beta\mu_k)}{1 - \beta + \beta\mu_k},$$

what holds under our assumptions in statements (1) and (2)

$$\nu \leq 0 \Leftrightarrow c_k(\mu_k - \theta_k) + d_k\theta_k(1 - \beta + \beta\mu_k) \leq 0$$

Case for serving class 2 we can obtained analogously.

Similarly goes the proof for statements 3 and 4, where the Bellman's equation for state $n = 0$ (job is already completed or abandoned) is:

$$V_{k,0}^* = -\max\{R_{k,0}^1 - \nu W_{k,0}^1 - \beta V_{k,0}^*; R_{k,0}^0 - \nu W_{k,0}^0 - \beta V_{k,0}^*\}.$$

If we choose that we are serving we can derive value that $V_{k,0}^* = \frac{\nu}{(1-\beta)}$. In the case, that we chose that we are not serving, we can derive that $V_{k,0}^* = 0$.

Therefore, when it is optimal to serve, action 1 is \geq action 2, we obtain condition:

$$-\nu - \beta \left(\frac{\nu}{(1-\beta)} \right) \geq -\beta \left(\frac{\nu}{(1-\beta)} \right)$$

We have assumed that $\nu \leq \nu_{k,0}^{AJN}$ and we know that $\nu_{k,0}^{AJN} = 0$. From the condition, we can observe that $\nu \leq 0$, so it is satisfied.

Analogous for the case, when it is optimal not to serve (action 1 is \leq action 2), we obtain similar condition that is satisfied when $\nu \geq 0$. This is true, because in the 2.5.1, we assumed that $\nu \geq \nu_{k,0}^{AJN}$, where $\nu_{k,0}^{AJN} = 0$

The proof for the last two statements 5 and 6 is intuitive. We know that ν is a price for using a server. Due to that it is obvious that when $\nu \leq \nu_{K,0}^{AJN}$, we choose to allocate the server to the alternative task K . Because we want to choose a bigger price. Analogously, when $\nu \geq \nu_{K,0}^{AJN}$, we choose not to allocate the server to the alternative task K .

□

2.5.2 Optimal Solution to Relaxations

The vector of policies $\boldsymbol{\pi}^* := (\tilde{\pi}_k^*)_{k \in \mathcal{K}}$ identified in Proposition (2.5.1) is formed by mutually independent single-user optimal policies, therefore this vector is an optimal policy to the Lagrangian relaxation (P_ν^L) .

Since a finite-state MDP admits an LP formulation using the standard *state-action frequency* variables (as observed in Niño-Mora (2001)), strong LP duality implies that there exists ν^* (possibly depending on the joint initial state) such that the Lagrangian relaxation $(P_{\nu^*}^L)$ achieves the same objective value as (P^W) . Further, if $\nu^* \neq 0$, then LP complementary slackness ensures that the β -average capacity constraint (2.6) is satisfied by any optimal solution to $(R_{\nu^*}^L)$.

2.5.3 AJN Rule for Original Problem

Since the original problem requires to allocate the server to exactly one option (one of the users or the alternative task), then at any time epoch t we propose to allocate the server to the user $k^*(t)$ with the highest actual price, i.e.,

$$k^*(t) := \arg \max_{k \in \mathcal{K}} \nu_{k, X_k(t)}^{\text{AJN}}.$$

Notice that any class without abandonment (i.e., having $\theta_k = 0$) has the index

$$\nu_{k,1}^{\text{AJN}} := \frac{c_k \mu_k}{1 - \beta}, \quad (2.13)$$

which is just the $c\mu$ index scaled by a constant. Moreover, under the time-average criterion such a class gets an absolute priority over any class with positive abandonment rate.

Under $\beta = 1$, we obtain the time-average version of the AJN index:

$$\nu_{k,1}^{\text{AJN}} := \begin{cases} \frac{c_k(\mu_k - \theta_k) + d_k \theta_k \mu_k}{\theta_k}, & \text{if } \nu_k^{1\text{U}} \geq 0 \\ \frac{c_k(\mu_k - \theta_k) + d_k \theta_k \mu_k}{\mu_k}, & \text{if } \nu_k^{1\text{U}} < 0 \end{cases} \quad (2.14)$$

where

$$\nu_k^{1\text{U}} := c_k(\mu_k - \theta_k) + d_k \theta_k \mu_k.$$

which we implement in our computational experiments in the next section.

Finally, we just remark that $\beta = 0$ gives rise to the myopic version of the AJN index:

$$\nu_{k,1}^{\text{AJN}} := c_k(\mu_k - \theta_k) + d_k \theta_k. \quad (2.15)$$

Chapter 3

Computational Study

In this chapter we report on an exhaustive study of numerical experiments. We consider a system with two classes of users. Each class is characterized by a set of values for the parameters μ , θ , c and d as before, and the mean λ of Poisson arrivals. For higher relevance in applications we consider a continuous-time model (so λ , μ and θ are rates). Therefore, we first introduce computational methods we use and theoretical background for using computational methods for the continuous-time model. In the second section we report on obtained results for idling and non-idling systems.

3.1 Computational Methods

In the dynamic programming exists two main approaches for solution: the value iteration algorithm and the policy iteration algorithm. Throughout this thesis we use the value iteration algorithm with truncated state space.

3.1.1 Value Iteration and State Space Truncation

Value iteration (aka backward induction) was first proposed by Bellman in 1957 (see Bellman (1957)). In this technique we do not compute policy π_n , for horizons $n = 0, 1, \dots$, (if it is needed we can compute it), but we compute value functions $V_n(\cdot)$ for each state.

In our experiments we consider a continuous-time model with the infinite horizon, by a reason that we want to increase relevance for possible applications. In addition, we focus on an average-cost criteria in our computational study, because problems that occur in the telecommunication have this setting. The following definitions are set in the environment proposed in the section (2.2) and are based on the book Ross (1983).

Definition 3.1.1. *Average-cost criteria is to find a policy that minimizes*

$$\limsup_{n \rightarrow \infty} \frac{1}{n} \mathbb{E} \left(\sum_{t=0}^{n-1} C(X_t) \right)$$

Theorem 3.1.2. *Let $(g, V(\cdot))$ be a solution of the average-cost optimality equation*

$$g + V(x) = C(x) + \min_{a \in A(x)} \sum_y p_{x,y}^a, \text{ for all states } x \quad (3.1)$$

then g equals the minimum average cost.

The function $V(\cdot)$ from previous theorem is the value function.

In the Value Iteration algorithm we investigate the value functions defined as follows

Definition 3.1.3. *Value functions $V_n(\cdot)$ satisfy*

$$V_0(x) = 0$$

$$V_{n+1}(x) = C(x) + \min_{a \in A(x)} \left\{ \sum_y p_{x,y}^a V_n(y) \right\}, n = 0, 1, \dots, \quad (3.2)$$

Under certain conditions, minimizing actions in (3.2) converge to actions that generate an average-cost optimal policy (Verloop (2009)), moreover for $n \rightarrow \infty$ it holds that $V_n(\cdot) - ng \rightarrow V(\cdot)$ and $V_{n+1} - V_n \rightarrow g$ (Hernández-Lerma and Lasserre (1996)). In addition, under some assumptions (Kooze (2006)) for $V_n(\cdot)$ it converge to an average-cost optimal value function $V(\cdot)$ (Verloop (2009), Sennott (2005)).

The value iteration algorithm can be used for the numerical determination of an approximation of the average-cost optimal policy, when the state space is finite. This is based on a recursively computing the function $V_{n+1}(\cdot)$ until difference between the maximal and the minimal difference is sufficiently small. In the thesis we consider the infinite state space, hence in our computational study we apply the value iteration algorithm after the appropriate truncation of the state space.

In all our numerical experiments we set the appropriate truncation similarly as in van Dijk (1991).

3.1.2 The Uniformization Approach

In previous chapters we have established the discrete-time MDP model. In our computational study, we can equivalently consider the uniformized Markov process, as it was proposed in Puterman (2005): "The uniformization may be viewed as an equivalent process, in which the system state is observed at random times which are exponentially distributed with some parameter. Because of the Markov property, it begins anew at each observation point. Alternatively, this transformation may be viewed as inducing extra or "fictitious" transitions from a state to itself."

After the uniformization all transition epochs are generated by a Poisson process. Thus we can reformulate model as a discrete-time MDP.

3.2 Numerical Experiments

We are interested in the time-average performance of the whole system, i.e., including the user in service. Recall that in the case without abandonments, the system is work-conserving what implies that taking or not into account the user in service is equivalent. However, in the case of abandonments these two models are not equivalent, and we believe that the total cost in the system is a more relevant measure. As a consequence, in some cases it may not necessarily be optimal to serve, but rather to idle. This is not captured by the model in Atar et al. (2010), in which only the waiting users are considered and the $c\mu/\theta$ -rule derived for it. It was shown in Atar et al. (2010) that for a non-zero abandonment penalty d , the $c\mu/\theta$ -rule is

$$\frac{c_k\mu_k}{\theta_k} + d_k\mu_k. \quad (3.3)$$

We truncate the state space by allowing a maximum number of users in each class, and we then use the uniformization technique in order to obtain a discrete-time representation of the model. Using value iteration Puterman (2005) we obtain numerically the optimal policy and the worst possible policy. Then we calculate the relative suboptimality gap and the adjusted relative suboptimality gap produced by the rules AJN, $c\mu/\theta$, $c\mu$, 2U and myopic version of AJN (myopic rule) .

Definition 3.2.1. *Relative suboptimality gap is a difference between values of the chosen rule and the optimal strategy, over value for optimal strategy.*

Adjusted relative suboptimality gap is a difference between values chosen rule and the optimal strategy, over a difference between values of the worst strategy and the optimal strategy.

The myopic rule is

$$\nu_{k,1}^{\text{Myo}} := c_k(\mu_k - \theta_k) + d_k\theta_k.$$

We take care that the truncation levels are large enough so that the optimal policy obtained in this way is almost surely the optimal policy in the untruncated problem.

We present obtained results both for idling and non-idling system.

3.2.1 Idling System

In this type of system we consider that system can idle. That influence the optimal, the worst policy and also AJN index. Other rules mentioned before: $c\mu$, $c\mu/\theta$, 2U and myopic are always serving some class.

The time-average version of AJN index for idling system is:

$$\nu_{k,1}^{\text{AJN}} := \frac{c_k(\mu_k - \theta_k) + d_k\theta_k\mu_k}{\theta_k}$$

We do not need to concern about a case for $\nu_k^{1U} < 0$, because in idling system it is more efficient to idle.

Table 3.1: Parameters for all the scenarios in computational experiments for the idling system.

	μ_1	μ_2	θ_1	θ_2	c_1	c_2	d_1	d_2	λ_1	λ_2
Scenario 1	0.7	0.3	[0, 2]	0.2	1	1	1	1	1	1
Scenario 2	0.4	0.59	[2, 3]	4	1	1	1	1	1	1
Scenario 3	0.8	0.7	1.2	2.7	1	1	[0, 50]	1	1	1
Scenario 4	0.8	0.7	1.2	2.7	1	1	[0, 10]	5	1	1
Scenario 5	0.4	0.1	0.5	0.8	1	[0.01, 20]	1	1	1	1
Scenario 6	0.4	0.22	0.1	0.2	1	[1, 40]	1	1	1	1
Scenario 7	0.4	0.3	0.001	0.03	1	[1, 60]	1	1	1	1
Scenario 8	0.1	0.4	0.12	0.1	[0.01, 20]	1	50	1	1	1

We have investigated a wide range of settings for the parameters in around 200 scenarios, and we report here the results of 6 representative scenarios in order to provide a global panorama. We further present 2 additional scenarios with unique and peculiar results. In each of the scenarios, only a single parameter is varied in order to easily depict the effect. In 3.2 are presented the parameters considered in each of the scenarios. Note that in all the scenarios the system is in overload (as in Atar et al. (2010)), having

$$\frac{\lambda_1}{\mu_1} + \frac{\lambda_2}{\mu_2} > 1,$$

which implies that abandonment are required in order to stabilize the system.

Given the number of parameters that we can choose from, the number of scenarios one can construct is virtually unbounded. Nevertheless, there are some general conclusions that we can draw:

- Almost always the AJN rule is equivalent or outperforms the $c\mu/\theta$ rule;
- In cases in which the optimal policy chooses to idle instead of serving, then AJN is much better than $c\mu/\theta$ or $c\mu$;
- In many scenarios AJN is equal to the optimal policy for almost all values of the varied parameter;
- The switching point of the 2U-rule is often very close to AJN, but usually its suboptimality region is larger.
- If both the 2U and AJN index for class 1 are greater than both for class 2, then it is almost always optimal to serve class 1.

The first 6 scenarios illustrate these general conclusions. In all following figures AJN rule is represented by blue color, $c\mu/\theta$ by red color, $c\mu$ by green line, 2U by yellow dotted line and Myopic index is represented with purple line.

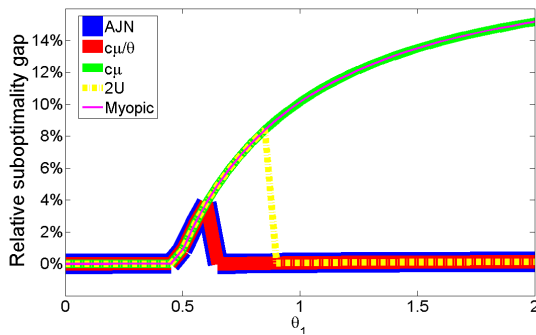


Figure 3.1: Relative suboptimality gap - Scenario 1

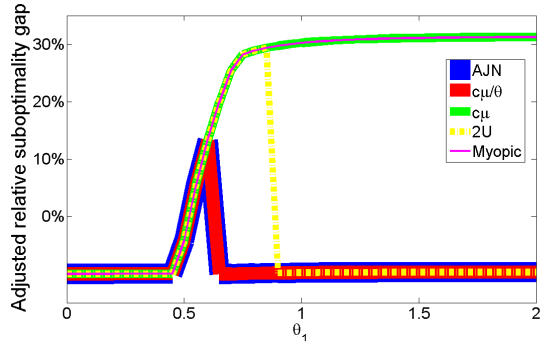


Figure 3.2: Adjusted relative suboptimality gap - Scenario 1

Scenario 1. (3.1)(3.2) In this scenario the performance of AJN and $c\mu/\theta$ is equivalent, and optimal except for a small interval. We also observe that the $c\mu$ rule with myopic rule performs very poorly. The performance of 2U is quite good, and the only difference with respect to AJN and $c\mu/\theta$ is the switching point where the policy starts serving class-2 users.

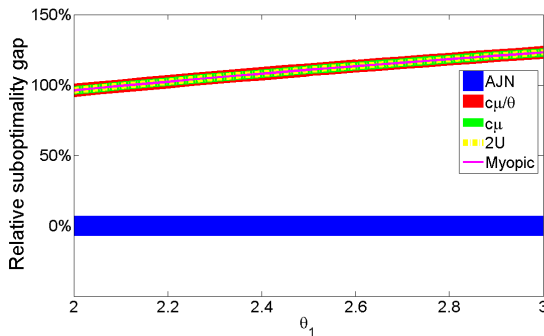


Figure 3.3: Relative suboptimality gap - Scenario 2

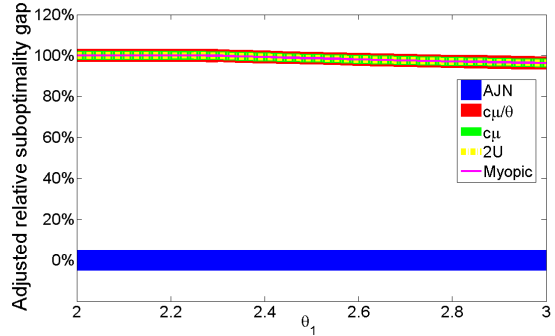


Figure 3.4: Adjusted relative suboptimality gap - Scenario 2

Scenario 2. (3.3)(3.4) For this scenario it is optimal not to serve any user, and to leave them abandon. The 2U and AJN policies capture this feature (and are optimal), but the $c\mu$, $c\mu/\theta$ and myopic rules do not. As a consequence the performance of the former two is much better than the latter two.

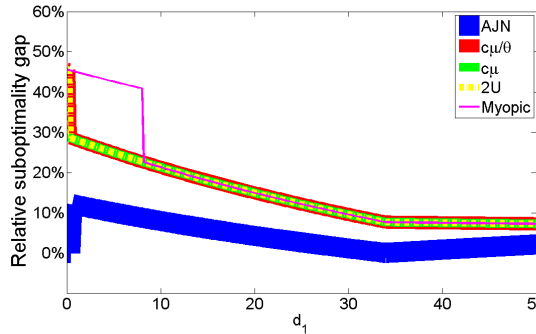


Figure 3.5: Relative suboptimality gap - Scenario 3

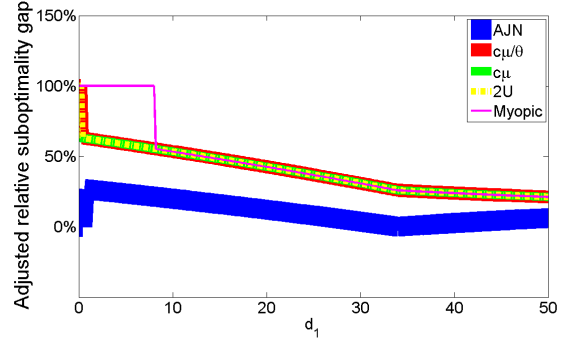


Figure 3.6: Adjusted relative suboptimality gap - Scenario 3

Scenario 3. (3.5)(3.6) For values of d_1 smaller than 35, the optimal policy does not serve any user, and for values larger than 35 it serves class 1 users (or idles if no class 1 user is waiting). All the other policies give priority to class 1, being the only difference that AJN and 2U idle if there is no class 1 user, whereas $c\mu$ and $c\mu/\theta$ serve class 2 in such a case.

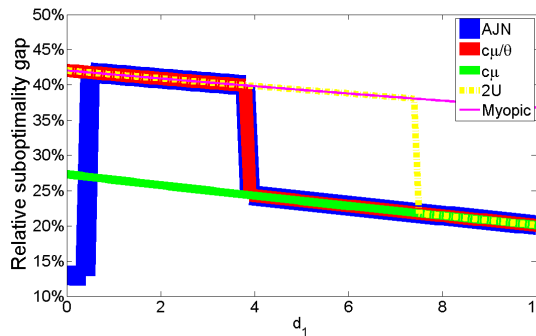


Figure 3.7: Relative suboptimality gap - Scenario 4

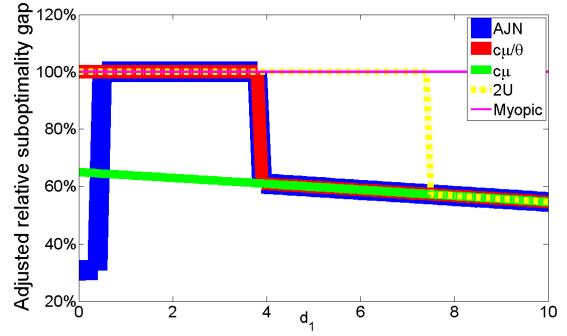


Figure 3.8: Adjusted relative suboptimality gap - Scenario 4

Scenario 4. (3.7)(3.8) Compared to the previous scenario, only the value of d_2 is 5 instead of 1, and this produces a significant difference in the results. Even though θ 's are still larger than μ 's, in this case the performance of 2U and AJN differ during a non-negligible range of values for d_1 . Interestingly, the $c\mu$ -rule outperforms (or matches) all the other policies. For this range of values, the optimal policy is not to serve any user. The $c\mu$ rule always serve class 1, myopic rule always serve class 2 and the other policies start serving class 2 users (as a consequence of d_2), and switch to serve class 1 (first AJN together with $c\mu/\theta$, and afterwards 2U) as the value of d_1 becomes larger.

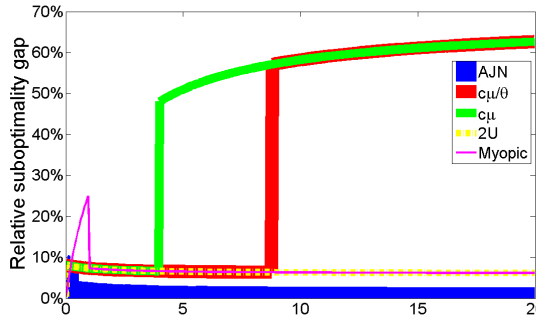


Figure 3.9: Relative suboptimality gap - Scenario 5

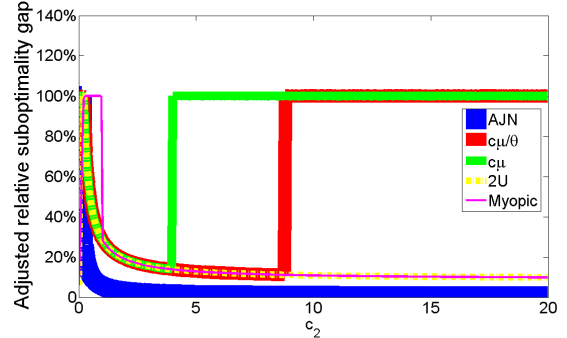


Figure 3.10: Adjusted relative suboptimality gap - Scenario 5

Scenario 5. (3.9)(3.10) The optimal policy is to idle. Policies AJN and 2U give priority to class 1, and to idling if there no user of class 1 (class 2 is never served). As the value of c_2 increases, the policies $c\mu$ and $c\mu/\theta$ switch to give priority to class 2, what makes a sudden increase in the cost function. The key difference is that AJN depends on the difference $\mu - \theta$, and thanks to this it chooses not to serve class 2 regardless of the value of c_2 . AJN’s performance is very close to optimal.

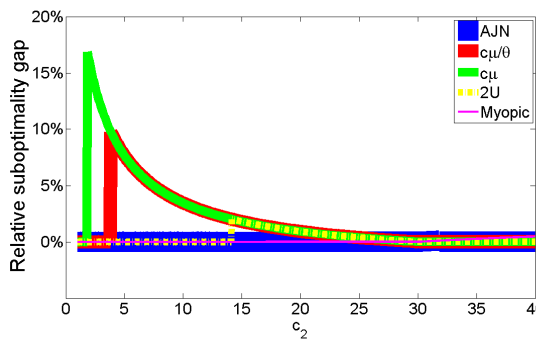


Figure 3.11: Relative suboptimality gap - Scenario 6

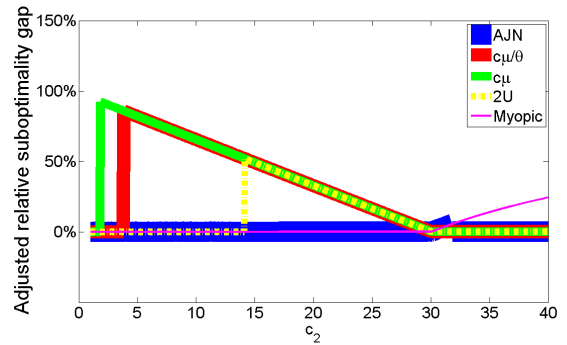


Figure 3.12: Adjusted relative suboptimality gap - Scenario 6

Scenario 6. (3.11)(3.12) This is a particularly interesting scenario. Service rates are larger than abandonment rates, but the policy AJN shows a better performance than the other policies. In fact, AJN is optimal for all values of c_2 with the exception of a small range around 32. The optimal policy starts serving class 1 in almost all system states, but as the value of c_2 increases it starts serving class 2 in more states. The AJN policy serves class 1 with strict preference for values of c_2 smaller than 32, and class 1 from that moment on. The upward jump for the other indices happens when they start giving priority to class 2 (first $c\mu$ switches, then $c\mu/\theta$ and then 2U).

The following two scenarios illustrate some specific and uncommon phenomena we have found in our experiments.

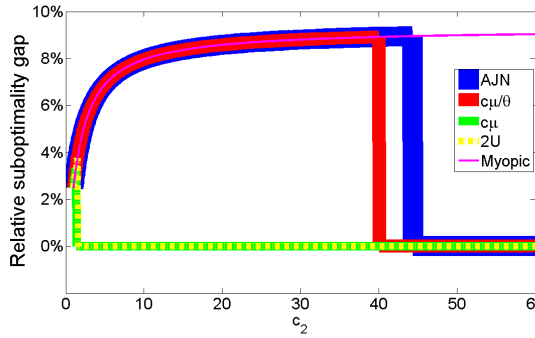


Figure 3.13: Relative suboptimality gap - Scenario 7

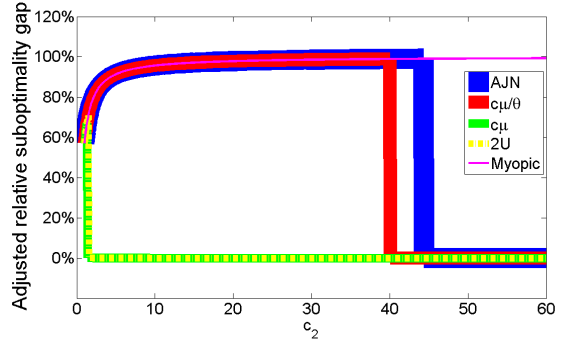


Figure 3.14: Adjusted relative suboptimality gap - Scenario 7

Scenario 7. (3.13)(3.14) In this scenario the abandonment rate is very small, say negligible. We recall that without abandonments, the $c\mu$ -rule is optimal Cox and Smith (1961); Buyukkoc et al. (1985). In the numerical experiments we see that, with a rather surprising exception for $c_2 = 1$, the $c\mu$ -rule is indeed optimal in this case and the 2U-rule is equivalent to $c\mu$. Policies AJN and $c\mu/\theta$ start serving class 1, and switch later on to class 2 when the value of c_2 becomes sufficiently large.

We emphasize that this is the only scenario we have found where the decision pattern of 2U and AJN differs completely and also the only one in which $c\mu/\theta$ outperforms AJN (for values of c_2 between 40 and 44).

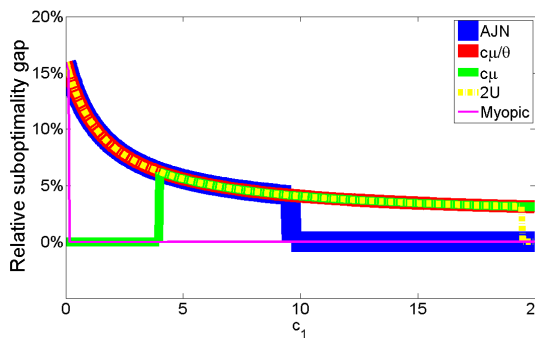


Figure 3.15: Relative suboptimality gap - Scenario 8

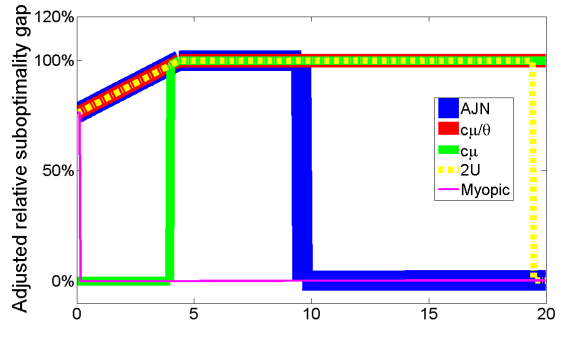


Figure 3.16: Adjusted relative suboptimality gap - Scenario 8

Scenario 8. (3.15)(3.16) This scenario has a mixed setting since $\mu_1 < \theta_1$, but $\mu_2 > \theta_2$. We vary c_1 . The optimal policy always serves class 2 users. For small values

of c_1 , the $c\mu$ -rule serves class 2, but it then switches to class 1, and therefore it is suboptimal for all larger values of c_1 . AJN and 2U start serving class 1, and when c_1 becomes sufficiently large they switch to class 2 (since $\mu_1 < \theta_1$).

In this case $c\mu/\theta$ remains suboptimal for all values of c_1 . However, each of the other three policies is optimal on some subrange of the parameter space.

3.2.2 Non-Idling System

In this case we forbid system to idle. Indices: $c\mu$, $c\mu/\theta$, 2U and myopic are same as before. The optimal policy, the worst policy and AJN index are changing. The time-average version of the AJN index for the non-idling system is:

$$\nu_{k,1}^{\text{AJN}} := \begin{cases} \frac{c_k(\mu_k - \theta_k) + d_k\theta_k\mu_k}{\theta_k}, & \text{if } \nu_k^{\text{1U}} \geq 0 \\ \frac{c_k(\mu_k - \theta_k) + d_k\theta_k\mu_k}{\mu_k}, & \text{if } \nu_k^{\text{1U}} < 0 \end{cases}$$

where

$$\nu_k^{\text{1U}} := c_k(\mu_k - \theta_k) + d_k\theta_k\mu_k.$$

Also for the non-idling system we analyze a wide range of experiments. For better comparison we report same settings of experiments as for the idling system.

Table 3.2: Parameters for all the scenarios in computational experiments for the non-idling system.

	μ_1	μ_2	θ_1	θ_2	c_1	c_2	d_1	d_2	λ_1	λ_2
Scenario 1	0.7	0.3	[0, 2]	0.2	1	1	1	1	1	1
Scenario 2	0.4	0.59	[2, 3]	4	1	1	1	1	1	1
Scenario 3	0.8	0.7	1.2	2.7	1	1	[0, 50]	1	1	1
Scenario 4	0.8	0.7	1.2	2.7	1	1	[0, 10]	5	1	1
Scenario 5	0.4	0.1	0.5	0.8	1	[0.01, 20]	1	1	1	1
Scenario 6	0.4	0.22	0.1	0.2	1	[1, 40]	1	1	1	1
Scenario 7	0.4	0.3	0.001	0.03	1	[1, 60]	1	1	1	1
Scenario 8	0.1	0.4	0.12	0.1	[0.01, 20]	1	50	1	1	1

Given the number of parameters that we can choose from, the number of scenarios one can construct is virtually unbounded. Nevertheless, there are some general conclusions that we can draw:

- Almost always the AJN rule is equivalent or outperforms the $c\mu/\theta$ rule;
- Performance of the AJN rule is little bit worse as in the idling system;

In scenarios 3 and 5 we can observe the biggest change in AJN rule performance. In all following figures AJN rule is represented by blue color, $c\mu/\theta$ by red color, $c\mu$ by green line, 2U by yellow dotted line and Myopic index is represented with purple line.

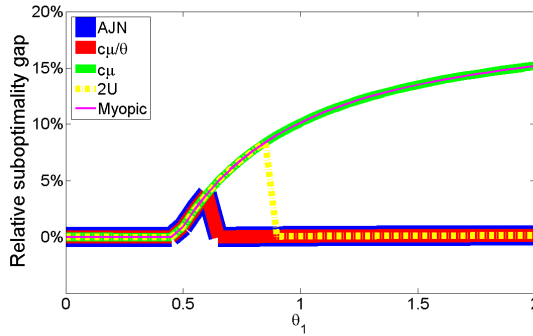


Figure 3.17: Relative suboptimality gap - Non-idling Scenario 1

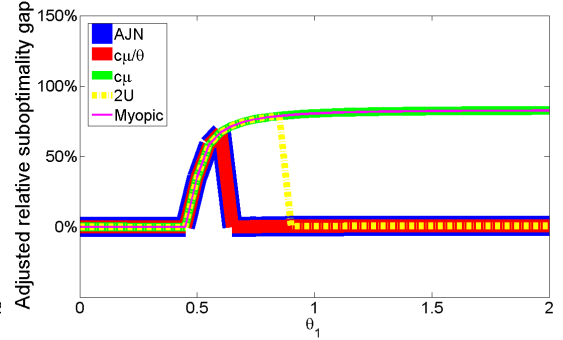


Figure 3.18: Adjusted relative suboptimality gap - Non-idling Scenario 1

Scenario 1. (3.17)(3.18) In this scenario the performance is almost same as in the idling case. AJN and $c\mu/\theta$ is equivalent, and optimal except for a small interval. We also observe that the $c\mu$ rule performs very poorly. The performance of 2U is quite good, and the only difference with respect to AJN and $c\mu/\theta$ is the switching point where the policy starts serving class-2 users.

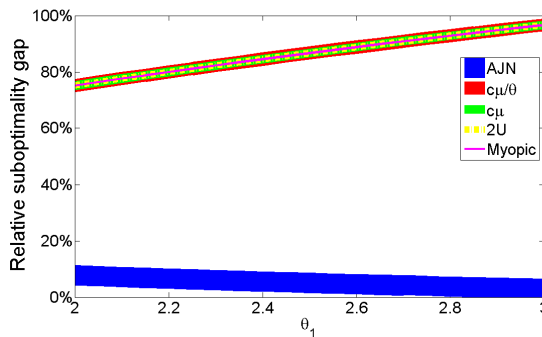


Figure 3.19: Relative suboptimality gap - Non-idling Scenario 2

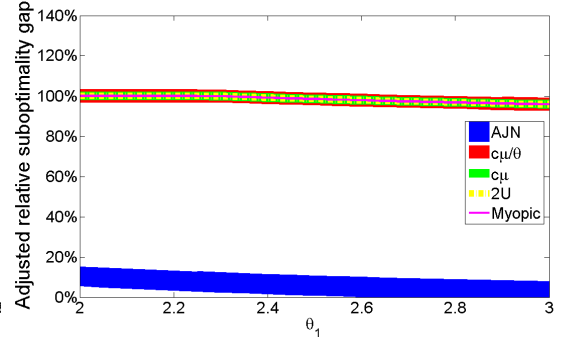


Figure 3.20: Adjusted relative suboptimality gap - Non-idling Scenario 2

Scenario 2. (3.19)(3.20) For this scenario in the idling case was optimal not to serve any user, and to leave them abandon. After system modifying to non-indling the AJN rule is not longer able to capture this feature (and are not optimal), but is better than the 2U, $c\mu$ and $c\mu/\theta$ rules.

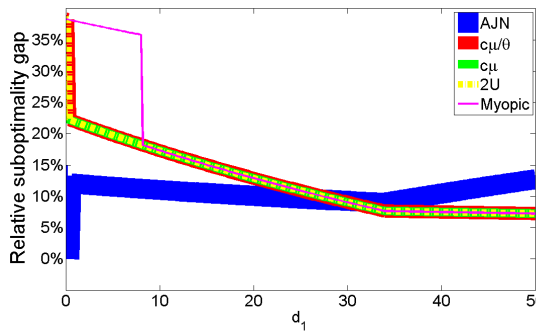


Figure 3.21: Relative suboptimality gap - Non-idling Scenario 3

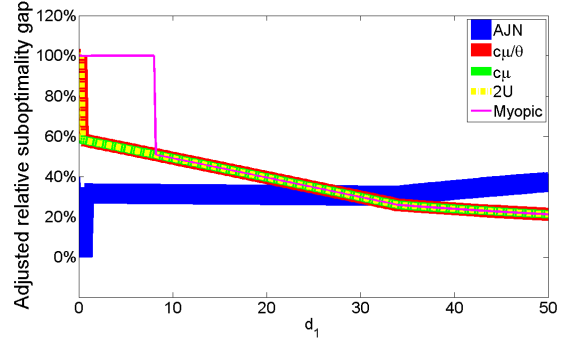


Figure 3.22: Adjusted relative suboptimality gap - Non-idling Scenario 3

Scenario 3. (3.21)(3.22) In this scenario we can observe the biggest difference between AJN performance in the idling system and the non-idling system. AJN is still serving same class and it is not able to switch serving class for d_1 bigger than 35. This is the only case, what we found, where AJN is the worst rule. Therefore, we want to study this scenario more detailed.

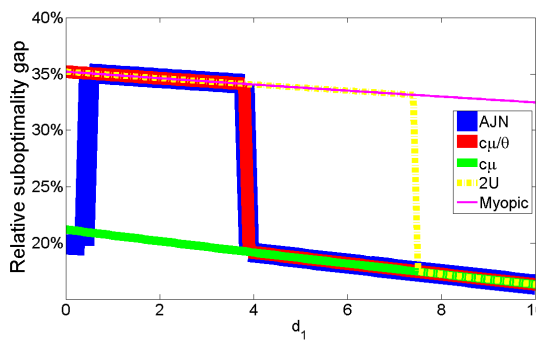


Figure 3.23: Relative suboptimality gap - Non-idling Scenario 4

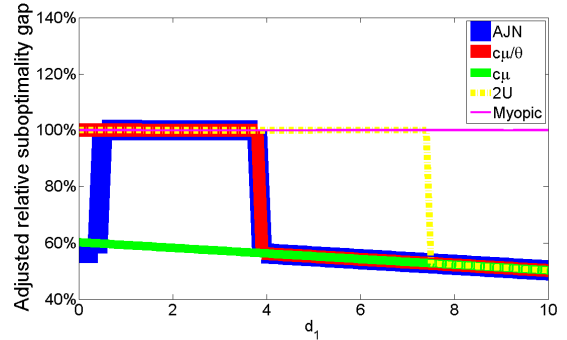


Figure 3.24: Adjusted relative suboptimality gap - Non-idling Scenario 4

Scenario 4. (3.23)(3.24) Compared to the previous scenario, only the value of d_2 is 5 instead of 1, and this produces a significant difference in the results. Even though θ 's are still larger than μ 's, in this case the performance of 2U and AJN differ during a non-negligible range of values for d_1 . Interestingly, the $c\mu$ -rule outperforms (or matches) all the other policies. The $c\mu$ rule always serve class 1, and the other policies start serving class 2 users (as a consequence of d_2), and switch to serve class 1 (first AJN together with $c\mu/\theta$, and afterwards 2U) as the value of d_1 becomes larger.

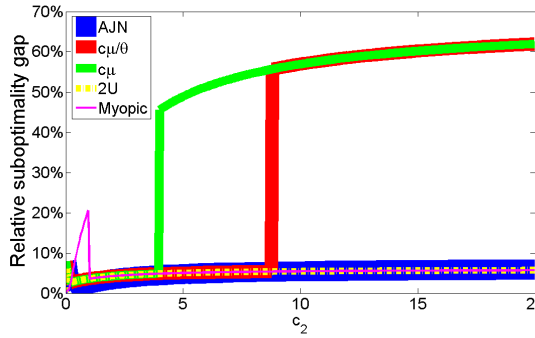


Figure 3.25: Relative suboptimality gap - Non-idling Scenario 5

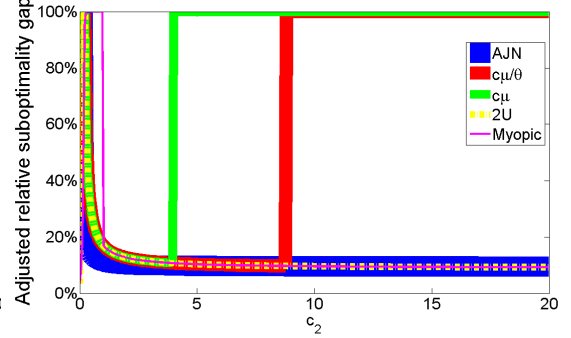


Figure 3.26: Adjusted relative suboptimality gap - Non-idling Scenario 5

Scenario 5. (3.25)(3.26) The optimal policy is to idle. Policies AJN and 2U give priority to class 1. As the value of c_2 increases, the policies $c\mu$ and $c\mu/\theta$ switch to give priority to class 2, what makes a sudden increase in the cost function. AJN's performance is not far from optimal, but it loses advantage of idling and become to has same performance as 2U and Myopic.

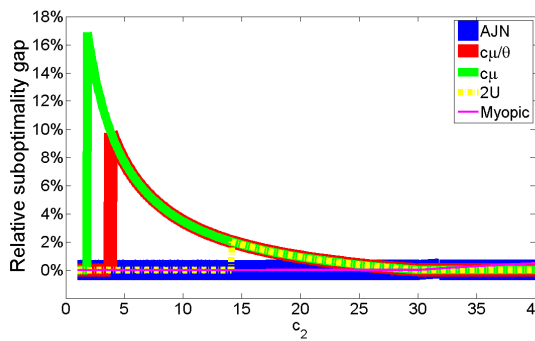


Figure 3.27: Relative suboptimality gap - Non-idling Scenario 6

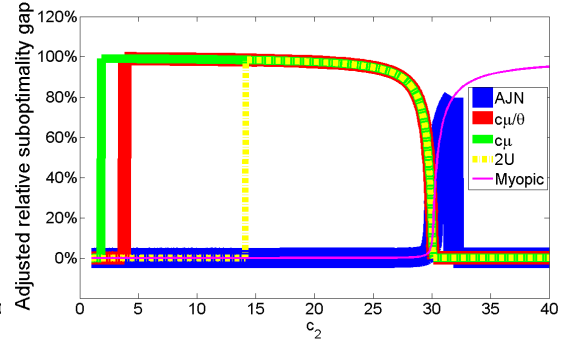


Figure 3.28: Adjusted relative suboptimality gap - Non-idling Scenario 6

Scenario 6. (3.27)(3.28) This is a particularly interesting scenario. Service rates are larger than abandonment rates, but the policy AJN shows a better performance than the other policies. In fact, AJN is optimal for all values of c_2 with the exception of a small range around 32. The optimal policy starts serving class 1 in almost all system states, but as the value of c_2 increases it starts serving class 2 in more states. The AJN policy serves class 1 with strict preference for values of c_2 smaller than 32, and class 1 from that moment on. The upward jump for the other indices happens when they start giving priority to class 2 (first $c\mu$ switches, then $c\mu/\theta$ and then 2U).

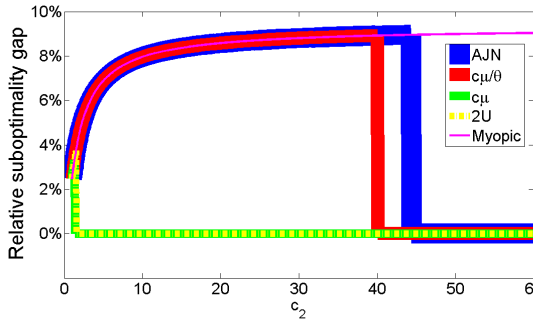


Figure 3.29: Relative suboptimality gap - Non-idling Scenario 7

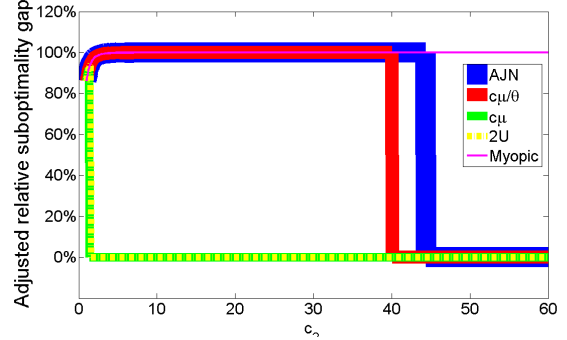


Figure 3.30: Adjusted relative suboptimality gap - Non-idling Scenario 7

Scenario 7. (3.29)(3.30) In this scenario the abandonment rate is very small, say negligible. We recall that without abandonments, the $c\mu$ -rule is optimal Cox and Smith (1961); Buyukkoc et al. (1985). In the numerical experiments we see that, with a rather surprising exception for $c_2 = 1$, the $c\mu$ -rule is indeed optimal in this case and the 2U-rule is equivalent to $c\mu$. Policies AJN and $c\mu/\theta$ start serving class 1, and switch later on to class 2 when the value of c_2 becomes sufficiently large.

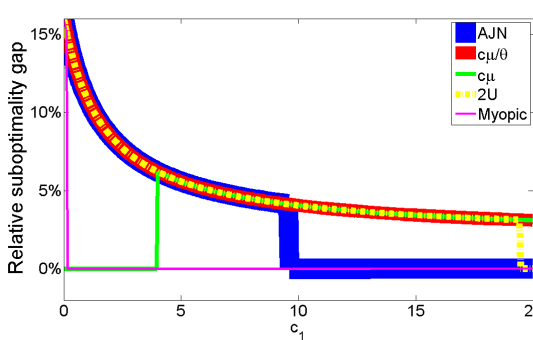


Figure 3.31: Relative suboptimality gap - Non-idling Scenario 8

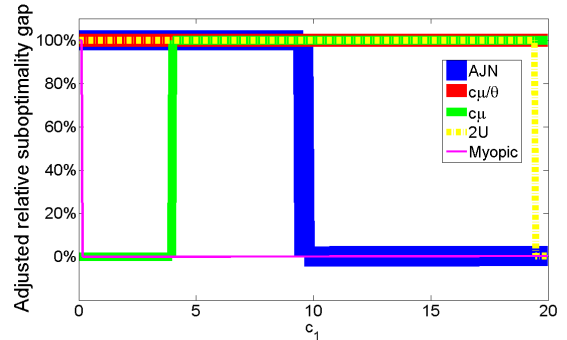


Figure 3.32: Adjusted relative suboptimality gap - Non-idling Scenario 8

Scenario 8. (3.31)(3.32) This scenario has a mixed setting: $\mu_1 < \theta_1$, but $\mu_2 > \theta_2$. We vary c_1 . The optimal policy always serves class 2 users. For small values of c_1 , the $c\mu$ -rule serves class 2, but it then switches to class 1, and therefore it is suboptimal for all larger values of c_1 . AJN and 2U start serving class 1, and when c_1 becomes sufficiently large they switch to class 2 (since $\mu_1 < \theta_1$).

In this case $c\mu/\theta$ remains suboptimal for all values of c_1 . However, each of the other three policies is optimal on some subrange of the parameter space.

Conclusion

In this work we focus on development of index policies for dynamic and stochastic problems. We introduce techniques for these systems description and for Whittle's index policies derivation. Moreover, we have investigated the problem of job scheduling with user abandonments. This is an important problem in several application fields, for which no general solution is known. We have proposed a comprehensive model accounting for both the linear holding costs and the abandonment penalties. For the problem with one or two users in the system, we have obtained an optimal solution.

For the more general case with multiple users, we have applied Whittle's relaxation methodology to derive the AJN-rule, a heuristic scheduling rule which has a very simple structure. It was derived for two system option with idling and non-idling behavior. This rule is under some conditions equivalent to the $c\mu/\theta$ -rule that was proven asymptotically optimal in Atar et al. (2010).

We report on an exhaustive numerical study for the idling and the non-idling system. In the idling system AJN performs exceptionally well: it is often optimal, and if not, then its suboptimality is small. Numerical results also indicate that in the most cases the AJN-rule outperforms the $c\mu/\theta$ and $c\mu$ rules.

The biggest improvement of the AJN over $c\mu/\theta$ and $c\mu$ is achieved when it is optimal to idle, that is for instance, when the abandonment probability is relatively large to the service-departure probability. In this case the suboptimality of $c\mu/\theta$ can be larger than 100%, while AJN may be optimal at the same time. Another important differences can be observed when the holding costs differ across classes. Interestingly, our scheduling rule recovers known optimal policies in some special cases of the problem, for instance, our rule becomes the $c\mu$ -rule if there is no abandonment.

In the non-idling system the AJN performance is not as good as in the idling system. Often its performance is same as in the idling case, but it is not so often optimal and also number of cases in which the AJN performance is worse than $c\mu/\theta$ is bigger. Therefore we propose to use AJN- rule mainly in idling systems.

An important question for the future research is to determine under what conditions the AJN-rule is optimal in asymptotic and non-asymptotic cases and whether AJN-rule for non-idling system could be improved.

Resumé

V práci sa venujeme formulácii, a za použitia Bellmanovej rovnice a Whittlovej metódy, aj riešeniu konkrétneho modelu pre rozvrhovanie úloh užívateľov viacerých tried. Keďže problém rozvrhovania úloh užívateľov viacerých tried je veľmi náročný, zameriavame sa na odvodenie indexových stratégií pre tieto problémy. Tie nám poskytujú ľahko implementovateľné a elegantné heuristické pravidlá na riešenie týchto problémov, vyznačujúcich sa veľmi dobrými, takmer optimálnymi výsledkami. V skúmanom modeli máme zahrnutý aj fenomén odchodov užívateľov. Ten má pre praktické použitie nesmierny význam, pre jeho častý výskyt a súčasne nevedomosť, ako najlepšie matematicky tento problém riešiť. Okrem odvodenia nového heuristického pravidla poukazujeme aj na rozsiahlu štúdiu zaoberajúcu sa výkonnosťou daného pravidla v porovnaní s ostatnými známymi a dosiaľ často používanými pravidlami.

V prvej kapitole predstavujeme základné princípy a charakteristiky metódy, ktoré sú potrebné na pochopenie problémov rozvrhovania úloh užívateľov viacerých tried a na odvodenie indexových stratégií. Na jednoduchom príklade ukazujeme spôsob formulácie takejto úlohy v prostredí Markovovských rozhodovacích procesov.

V druhej kapitole predstavujeme skúmaný problém, v ktorom našim cieľom je minimalizácia celkových nákladov a pokút za odchody užívateľov. Ten následne formulujeme v prostredí Markovovských rozhodovacích procesov. Ide o rozšírenie dosiaľ známeho problému rozvrhovania úloh užívateľom rôznych tried. Problém je rozšírený o možné odchody používateľov a súčasne sa zaoberáme aj dvoma rôznymi špecifikáciami serverov. V jednej je dovolená nečinnosť serveru a v druhej zakázaná. Okrem iného, na rozdiel od posledných obdobných výsledkov predstavených v Atar et al. (2010), v našom modeli aj práve obsluhovaný užívateľ prispieva k nákladovej funkcii. To má opodstatnenie v praxi a ako sa ukázalo aj signifikantný dopad na dosiahnuté výsledky.

Vyššie spomenutý model v práci najprv analyticky riešime pre prípady s jedným alebo dvoma užívateľmi. Keďže pre vyššie počty užívateľov problém už nie je možné analyticky riešiť, problém relaxujeme, nasledujúc Whittlovu metódu a na základe optimálneho riešenia relaxovaného problému odvádzame nové heuristické pravidlo, pre pôvodný problém. V závere kapitoly predstavujeme niekoľko verzií tohto pravidla v závislosti od špecifikácie optimalizačného problému. Taktiež uvádzame aj scenáre, v ktorých naše pravidlo dostáva tvar známych a v daných scenároch optimálnych

stratégií.

V tretej kapitole predstavujeme rozsiahlu výpočtovú štúdiu pre oba typy serverov, v ktorých našu novú indexovú stratégiu numericky porovnávame s dosiaľ známymi alternatívami: $c\mu$ -stratégiou, o ktorej je dobre známe, že je optimálna pre systém s príchodmi aj bez príchodov užívateľov, avšak bez uvažovania odchodov užívateľov. Porovnávame ju aj s $c\mu/\theta$ - stratégiou, o ktorej bolo prednedávnom ukázané, že je asymptoticky optimálna stratégia v preťaženom systéme s viacerými servermi.

Výsledky ukazujú, že pre analyzovaný problém je odvodená stratégia takmer vždy lepšia, alebo porovnateľná s ostatnými. Lepšie výsledky dosahuje v prípade, keď je dovolená nečinnosť servera. Vďaka za to schopnosti uviesť server do nečinnosti v prípadoch, keď je to optimálne, čo je aj jeden z jej najväčších prínosov. Mnoho dosiaľ známych stratégií nie je schopných zachytiť takéto optimálne správanie. V prípade, keď je zakázaná nečinnosť servera, preukazuje odvodená indexová stratégia dobré výsledky, avšak mierne sa zvyšuje počet situácií, v ktorých je neoptimálna, alebo v ktorých je horšia ako niektorá z alternatív.

Bibliography

- Atar, R., Giat, C., and Shimkin, N. (2010). The $c\mu/\theta$ rule for many-server queues with abandonment. *Operations Research*, 58(5):1427–1439.
- Ayesta, U., Jacko, P., and Novak, V. (2011). A nearly-optimal index rule for scheduling of users with abandonment. In *Proceedings of IEEE INFOCOM 2011*, pages 2835–2843.
- Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.
- Bertsekas, D. P. (2005). *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, 3 edition.
- Buyukkoc, C., Varaya, P., and Walrand, J. (1985). The $c\mu$ rule revisited. *Adv. Appl. Prob.*, 17:237–238.
- Cox, D. R. and Smith, W. L. (1961). *Queues*. Methuen & Co. LTD, London.
- Fife, D. (1965). Scheduling with random arrivals and linear loss functions. *Management Science*, 11(3):429–437.
- Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B*, 41(2):148–177.
- Gittins, J. C. (1989). *Multi-Armed Bandit Allocation Indices*. J. Wiley & Sons, New York.
- Hernández-Lerma, O. and Lasserre, J. (1996). *Discrete-Time Markov Control Processes: Basic Optimality Criteria*. Springer-Verlag, New York.
- Jacko, P. (2009). Adaptive greedy rules for dynamic and stochastic resource capacity allocation problems. *Medium for Econometric Applications*, 17(4):10–16. Available online at <http://www.met-online.nl>. Invited paper.
- Jacko, P. (2010a). *Dynamic Priority Allocation in Restless Bandit Models*. Lambert Academic Publishing. Invited book.

- Jacko, P. (2010b). Restless bandits approach to the job scheduling problem and its extensions. In Piunovskiy, A. B., editor, *Modern Trends in Controlled Stochastic Processes: Theory and Applications*, pages 248–267. Luniver Press, United Kingdom.
- Koole, G. (2006). Monotonicity in Markov reward and decision chains: Theory and applications. *Foundations and Trends in Stochastic Systems*, 1:1–76.
- Niño-Mora, J. (2001). Restless bandits, partial conservation laws and indexability. *Advances in Applied Probability*, 33(1):76–98.
- Niño-Mora, J. (2007). Dynamic priority allocation via restless bandit marginal productivity indices. *TOP*, 15(2):161–198.
- Papadimitriou, C. H. and Tsitsiklis, J. N. (1999). The complexity of optimal queueing network. *Mathematics of Operations Research*, 24(2):293–305.
- Puterman, M. L. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.
- Ross, S. (1983). *Introduction to Stochastic Dynamic Programming*. Academic Press.
- Sennott, L. (2005). Value iteration in countable state average cost Markov decision processes with unbounded costs. *Annals of Operations Research*, 28:261–271.
- Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66.
- van Dijk, N. M. (1991). Truncation of markov chains with applications to queueing. *Operations Research*, 39:1018–1026.
- Verloop, I. (2009). *Scheduling in Stochastic Resource-Sharing Systems*. Ponsen&Looijen b.v.
- Whittle, P. (1980). Multi-armed bandits and the Gittins index. *Journal of the Royal Statistical Society, Series B*, 42(2):143–149.
- Whittle, P. (1988). Restless bandits: Activity allocation in a changing world. *A Celebration of Applied Probability, J. Gani (Ed.), Journal of Applied Probability*, 25A:287–298.