Queueing models with admission and termination control : monotonicity and threshold results

Queueing models with admission and termination control

Monotonicity and threshold results

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN

Brouns, Gido A.J.F.

Queueing models with admission and termination control — Monotonicity and threshold results / by Gido A.J.F. Brouns. - Eindhoven : Technische Universiteit Eindhoven, 2003.
Proefschrift. - ISBN 90-386-0732-6
NUR 919
Keywords: Markov decision processes / dynamic programming / queueing theory
2000 Mathematics Subject Classification: 90C40, 90C39, 60K25, 90B22

Printed by Universiteitsdrukkerij Technische Universiteit Eindhoven

Queueing models with admission and termination control

Monotonicity and threshold results

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr. R.A. van Santen, voor
een commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen op
donderdag 30 oktober 2003 om 16.00 uur

door

Gido Antonius Johannes Fransiscus Brouns

geboren te Amsterdam

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. J. van der Wal
en
prof.dr. K.M. van Hee

"I hope I can make it across the border. I hope to see my friend and shake his hand. I hope the Pacific is as blue as it has been in my dreams.———I hope."

–*Red*, *The Shawshank Redemption*

Contents

1

Introduction

Someone once said that "The only person getting his work done by Friday was Robinson Crusoe". Ideally, the planning of activities is simple: one's schedule comprises oceans of time, in which one need not look any further than the next activity. In this primary world, finishing a task ahead or behind of time does not influence one's schedule, apart from pushing subsequent duties to an earlier or rather future point in time, nor does it have an impact on one's environment. It goes without saying that this does not accord with workaday reality. Insurance companies, tax offices, banks, criminal investigation bureaus and administrative departments of industrial organizations are not situated on desert islands. They are part of a larger system, which they influence and which they are influenced by. Moreover, the environment they interact with may pursue totally different and conflicting objectives. Still, they are expected to reckon and comply with wishes expressed and demands enforced by this environment, coping with additional limiting conditions at the same time. In practice, this means facing deadlines, dealing with unforeseen circumstances—e.g., unexpected outcomes of affairs, delay of work, and the sudden arrival of even more work—and commonly possessing insufficient resource capacity to serve all jobs completely, or to grant every single request for service. Hence, both time and capacity are precious. On-line decisions have to be taken on which jobs to serve and which to leave unattended, and on how to employ the available capacity, i.e., how much capacity to assign to the work-in-process and which resources to assign new work to. This will depend on the pressure of work on the one hand, and the benefit of carrying on with work-in-process on the other.

In this monograph, we focus on two essential on-line decisions that must be taken in scant-capacity operating environments:

- *admission* (and subsequent allocation) or *rejection* of new work, and
- *continuation* or *termination* of work-in-process.

By setting up a mathematical framework, in which the construction of queueing models that feature these decisions takes a prominent place, we can study and characterize the structure of optimal control policies with respect to these two decisions. This leads to a new class of decision models, which fits naturally within the field of dynamic control of queues.

1.1 Motivation

The organization of work within organizations continues to become more complex. This has given rise to the development of a general framework for the (automated) control of business processes: workflow management (WfMC [59, 60], Lawrence [38], Van der Aalst and Van Hee [3]). Stimulated by the IT boom of the 90s, the use of workflow management systems is now widespread across the service industry. These are sophisticated information systems that are capable of regulating the division and execution of work-in-process and future work. On the one hand, these systems dispose of a precise description of each flow of work, which can easily be translated to a stochastic network model. On the other hand, the current generation of workflow management systems does not provide suitable or sufficient functionality to satisfactorily account for the special characteristics of workflow processes (Brouns [9]). An important shortcoming is the lack of means for real-time quantitative decision support. Dynamic (resource) control on the basis of the true status of work-in-process, the expected supply of work and the available capacity is not possible, even if the required information is available to the controller. There remains a clear need for intelligent mechanisms and methodologies for workflow process control. In this respect, it is important to study the application of quantitative analysis techniques to the specific problems that can be identified in workflow environments. In particular, in this thesis, we focus on a mathematical study of the subtle relationship between the factors *resource capacity*, *throughput time* and *Quality of Service*.

1.2 Onset of a mathematical model formulation

The aim of this section is to bridge the gap between workflow problem characteristics and the formulation of quantitative models for dynamic workflow control.

1.2.1 Work execution

Business processes in the service industry are typically case-based: whenever work arrives, in terms of or as a result of an order, a request or a compulsory return, a *case* is opened. Every case induces one or more *tasks* to be executed. Usually, the amount of work-in-process varies heavily over time, and the moments at which work arrives are often so random that the arrival process of work can be described as a Poisson process. Alternatively, one could think of processes in which work typically arrives in large *batches* that have to be processed before some given deadline. Furthermore, the nature and extent of cases may differ greatly. The first will usually give rise to classification of work. This means that new work is directly associated with a specific *type* of case. For example, in crime investigation, many different types of crime can be distinguished, such as grand theft, vandalism, drug trafficking, assault, and murder. However, in general, the exact amount of work contained in cases of the same type is typically unknown beforehand. It is often unclear along which exact lines the investigation of a case has to be carried out. This becomes clear gradually during the execution of the process. Also the eventual outcome of a case is often unknown beforehand. Apparently similar cases may have dissimilar outcomes, and there does not exist a one-to-one correspondence between the development of a case and the outcome of the case. For example, in crime investigation, a homicide might be solved in a couple of months, might turn out to be in fact a case of suicide, or might never be solved. As a result of these uncertainties, it is unclear what total capacity engagement is required for a specific case. Different outcomes of intervening (whether or not sudden) events and investigations lead to different demands for capacity.

1.2.2 Deficient resource capacity

Service organizations are typically completely dependent on clients (or the public) as regards the supply of work. In other words, the supply of work cannot be controlled. Because of random arrivals, running out of

work at times is theoretically possible. However, in practice, a far more common phenomenon is a superfluous (and unavoidable) supply of work. Despite a range of possible actions, such as the introduction of overtime, temporary staff employment and multi-skilling of staff, the resource capacity will commonly be insufficient to execute all (or even the majority of all) tasks, let alone all cases. If we let ρ denote the *gross workload*, defined as the ratio between—on the one hand—the required processing capacity if all tasks involved with all arriving case-inducing work were to be executed, and—on the other hand—the actual processing capacity, then we are in principle interested in the situation that $\rho > 1$.

1.2.3 Throughput time

We define the throughput time of a case as the total time the case resides in the system. It is crucial to have control over the throughput times of cases, since work-in-process is directly related to cost. Consequently, the available resources need to be managed efficiently and deployed with care. In this thesis, throughput time will be associated with variable as well as fixed costs. The first are represented by means of *holding* costs. In practice, these can represent various types of 'costs'. For example, in the world of criminal justice, keeping a suspect in custody entails (literal) holding costs. Another example can be found in service organizations, where the notion of holding costs can be used to model a loss of goodwill, as experienced by customers (vainly) awaiting response. Besides holding costs, a second type of investment costs can commonly be identified. These are the fixed costs associated with a number of more or less standard tasks that must be carried out when it is decided that a new case is eligible for attention. These tasks include, for example, preparing, opening and eventually closing the case, and keeping case files. The respective costs are collectively termed *consideration* costs in this thesis.

1.2.4 Quality of Service

Any restriction on the attention (i.e., amount of consideration time) a case will receive—by limiting the capacity assigned to it—is inevitably negatively correlated with the *Quality of Service* (QoS). This is a performance measure that indicates the quality of the actual outcome of a case compared to the most favourable outcome achievable for that case. For instance, a taxation

office cannot expect to detect large-scale tax fraud without thoroughly examining tax returns. On the other hand, sifting through a tax return does not necessarily lead to the detection of fraud. This is an example of the so-called absence of strict causal connections between capacity and value added service. Furthermore, an increase in capacity with respect to some particular case will inevitably go at the expense of fresh cases awaiting and requiring attention.

Quality of Service is naturally associated with rewards. Whenever one of the tasks belonging to a certain case is completed, a new or additional (finite) reward is earned with respect to this case. A common assumption is that the overall reward for a case is non-decreasing and concave, in the following sense. Non-decreasing means that putting more work into a case does not leave us with a lower overall reward for that case. Concavity means that we have diminishing marginal returns, i.e., the longer we work on a case, the less rewarding it becomes to continue.

1.2.5 On-line decision-making

Rejection of work is commonly undesirable, viz., it can lead to adverse consequences such as negative publicity or an unintentional invitation to misconduct. But to alleviate the workload, it will frequently be a sheer necessity to reject new cases, hereby yielding the lowest possible (i.e., extremely poor) QoS for those cases, or to terminate running cases, either already under consideration or waiting for attention.[1] On a continuous basis, decisions must be taken on the progress of cases, based on the current amount of work-in-process and related costs on the one hand, and the QoS and related revenues on the other. Each time the state of the process changes—e.g., because of the arrival of new work or the completion of a work item—new decisions need to be taken. In this decision-making process, throughput time and QoS have to be weighed against each other. Under the

[1] Consider, as an illustration, the following extract from a Dutch news report of 2001. « Voorburg, The Netherlands - Official sources assess the total number of offences committed against citizens in 2000 at 4.7 million, of which 1.6 million were reported to the police, which, on their part, recorded 1.3 million. Further figures show that approximately one out of every seven (recorded) crimes was solved. The percentage of crimes that are solved has been decreasing steadily over the years. In explanation, a spokesman of the Netherlands Police Institute (NPI) states that investigations become more and more complex, which makes its demands on capacity, and that there is an apparent shift to violent crimes, which are reported more often. Almost half of these crimes are solved, but the investigations are particularly labour-intensive, said the spokesman. »

acute constraint that the available processing capacity will not be sufficient to treat each case to the full extent, the aim is to find an optimal trade-off (i.e., compromise) between the factors throughput time (and associated costs) and QoS (and associated revenues). This leads to what we call *partial execution* of work.

In this thesis, we provide an impetus to the modelling and analysis of control problems that feature the opportunity to either *accept* or *reject* new work (i.e., fresh cases) and to either *continue* or *abort* work-in-process (i.e., running cases) on a dynamic basis, such that partial execution of work is allowed for. Any other model components—e.g., the number of resources, their working speed, characteristics of the arrival process, characteristics concerning the content of cases—are not eligible for control. Note that by comparing different models, it will still be possible to evaluate the effect of more or rather less resources, a lighter or rather heavier workload, and so on, but this will not be further explored in this thesis.

Our goal is to contribute to the construction of a framework for on-line decision-making in workflow environments with deficient resource capacity, using modelling and analysis techniques from queueing theory and stochastic decision theory. The formulation of a workflow process in terms of a queueing system is fairly straightforward. However, to be able to formally analyse such a system, we are forced to make some simplifications. In particular, when analysing large systems, one encounters the problem that all parts of the system interact with one another in a complex way. This usually causes a serious impediment to a formal analysis. A common step to reduce the complexity is to decompose the system into separate parts, and to analyse each part separately. In this thesis, we will only consider so-called single-station problems; see Section 1.3. As it turns out, the analysis of such problems regularly proves to be already hard in itself.

Workflow problems encountered in practice are far more complicated than those that can be represented by means of the models we consider in this thesis, which are in the first place generic, i.e., they will be viewed and analysed on a stand-alone (not necessarily workflow related) basis. Nevertheless, our analysis forms an essential step towards more practical extensions. We are convinced that a better mathematical understanding of the structure of the optimal strategies for our simplified problems will be very helpful in more complex situations where good heuristics for dealing with deficient capacity are needed.

1.3 Queueing terminology

Systems in which the arrival and execution of work suffer from variability, are typically subject to *queueing*. Queueing theory was built on the foundations of probability theory, and finds application in a wide range of areas, where it is used to model and analyse real-life systems, such as production systems, communication systems, transport systems and computer systems. Below, we discuss briefly the basic concepts of queueing theory that are relevant in the context of this thesis. For a thorough introduction to queueing theory, we refer to Kleinrock [32], Cohen [15], and Takagi [54].

In general, a queueing model describes a situation where customers, or *jobs*, are to be served by a limited set of resources, or *servers*. The service requirements of a job comprise a number of tasks to be performed by one or more of the servers. Jobs awaiting service reside in a *queue*, which has a certain *buffer* capacity. Together with their joint buffer or their own private buffers, the servers at a particular location form a *station*. A queueing system may consist of one or more interconnected stations. In the latter case, we speak of a queueing *network*. In a queueing network, jobs can visit more than one queue and more than one server in succession after admission to the system. However, in this thesis, we concentrate on single-station queueing systems: any job will visit at most one queue.

The most basic queueing model is the single-server system shown in Figure 1.1. Jobs arrive at the station, one at a time, according to some arrival process (e.g., a Poisson process). Jobs may be of different types (e.g., may have different priorities assigned to them) and the amount of work jobs bring to the system may vary from job to job.

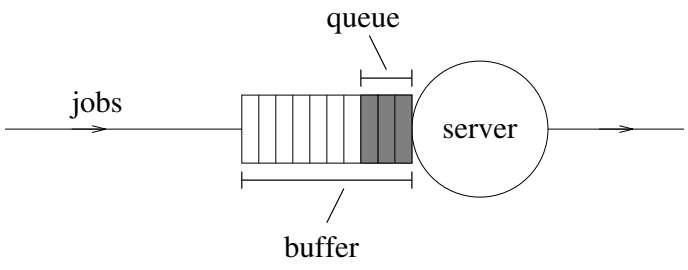

Figure 1.1: Single-server queueing model

The way the server allocates its capacity to the jobs in the system is determined by the *service discipline*. The most natural service discipline is FCFS ('First Come First Served'), which means that jobs are served in order of arrival. Many different service disciplines have been proposed and studied in literature. The two that are most relevant in the light of this thesis are FCFS and PR ('Preemptive Resume'). Under the PR discipline, the service of a job may be interrupted at any time in order to start serving a job of higher priority that has just entered the system. The lower priority job then joins the queue again, and its service may be resumed later.

1.4 Dynamic control of queues

The emphasis of queueing theory has originally been on performance evaluation. For a given set of predefined system characteristics (and possibly a set of parameter values), the behaviour and performance of the system can be evaluated analytically. However, many problems in practice involve systems in which certain characteristics are typically not fixed, and which may be adjusted continuously in order to obtain better performance. By introducing on-line decision features with respect to the way the system is operated, we allow for *dynamic stochastic control* of the queueing system. The operational procedures are described by a control *policy*, or *strategy*. Such a policy consists of a set of control rules that assign a certain *decision*, or *action*, to each state the system may find itself in. The focus is commonly on the determination or characterization of *optimal* control policies, i.e., policies that yield optimal performance with respect to some objective function, such as minimizing the sojourn times of jobs in the system, or maximizing profit in case of operational costs and job rewards.

1.4.1 Individually versus socially optimal policies

In general, we distinguish between *individually* optimal policies and *socially* optimal policies. Individually optimal policies consider the system from an individual customer point of view. In this view, customers aim for personal optimization. Socially optimal policies consider the system from an average customer point of view. In this view, the aim is at optimization over all customers collectively, which is usually in the best interest of the system itself. In our models, we are solely interested in social optimization.

To illustrate the basic concepts of dynamic control of queues, consider again the single-server queueing system of Figure 1.1. Note that this system automatically accepts and (eventually) serves any new job. Now suppose that jobs arrive at the station according to a Poisson process with arrival rate λ and that each job brings a reward r to the system and incurs holding costs h for each unit of time it resides in the system. Service times are exponential, and the service rate is fixed and equal to μ. We allow for (dynamic) *admission* control of the system: jobs may be either accepted or rejected upon arrival. Accepted jobs join the queue, whereas rejected jobs are discarded from the system immediately. See Figure 1.2. This model dates back to Naor [43]. The (social) objective is to find a policy that maximizes the average long-run profit (reward minus cost) obtained by the system.

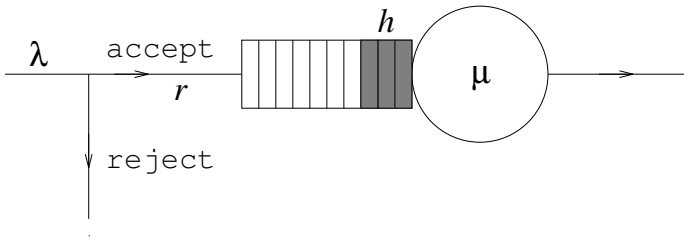

Figure 1.2: Single-server queueing model with admission control

If we let j denote the number of jobs already in the system upon arrival of a new job, then the individually optimal policy is to enter the system if $r > (j+1)h\mu^{-1}$ and to balk if $r < (j+1)h\mu^{-1}$. If $r = (j+1)h\mu^{-1}$, then either decision is optimal. This already shows that, in general, optimal policies need not be unique; hence the following remark.

Remark 1.1 Whenever we use the phrase 'the optimal policy', we do not suggest that there exists a *unique* optimal policy for the corresponding model.

The individually optimal policy for Naor's model has a *threshold*, or *switch-over* structure: there is an integer m such that it is optimal to enter if $j \leq m$ and to balk if $j > m$. It can be shown that the socially optimal policy has a threshold structure as well, and its critical number—which clearly also depends on the arrival rate—can be derived explicitly, although it requires some calculation effort; see [43].

1.4.2 Formalizing intuition

Intuitively, the threshold characterizations of the (individually and socially) optimal policies in the example of Figure 1.2 are 'obvious'. However, intuition may be misleading. Some well-known examples are due to Whitt [61], who considered a system with two or more parallel identical servers, each with its own infinite capacity queue. He showed that from a customer point of view, it can be optimal in certain circumstances to take the counterintuitive decision of joining the *longest* queue. So, although a good sense of intuition may be very helpful in the thought process, we cannot rely on it. Assertions on (probable) properties of the model and of the optimal policy for the model need to be substantiated, and occasionally, counterintuitive results are found. In fact, we will come across some counterintuitive results, or results that appear to be so at first sight, at a couple of instances in this thesis.

In the remainder of this introduction, we give an outline of literature on models for the dynamic control of queues, highlighting the types of control that have been investigated in literature so far, and indicating which ones are relevant in the light of our research. We only address models and types of control concerning single-station queueing systems and the objective of social optimization. The outline below is merely descriptive; the theory and methodology behind the models will be discussed in Section 1.7.

1.5 Outline of literature on the dynamic control of queues

There is an extensive literature on the dynamic control of queues. The early work of Crabill et al. [16] gives a review of research on the dynamic control of queues in its pioneering stage, ranging from the late 60s to the late 70s. A comprehensive overview of research on the dynamic control of queues up to the beginning of the 90s is given by Stidham and Weber [53]. Both surveys provide an extensive list of references to literature devoted to the analysis of specific queueing control models. Topics include optimal admission control, optimal routing (or flow) control (which is equivalent to optimal server allocation in case of a single-station system), optimal service rate control, optimal control of the number of servers, and optimal control of the service discipline (which includes optimal scheduling control in case of a single-station system). These topics can be classified into two main categories: the first two topics concern control of the arrival process, whereas

the other three concern control of the service process. A survey specifically oriented towards research on the latter is also given by Teghem [55]. Our classification is somewhat different than the one presented in [16] and [55], because we expressly consider the service discipline to be a service process control tool too.

1.5.1 Scope of the models

In both [55] and [53], the emphasis is on the use of models based on Markov decision theory (see Section 1.7) to examine the structure of optimal control policies. Sometimes, an explicit form of the optimal policy in terms of the system parameters can be distilled, e.g., the individually as well as the socially optimal policy in the example of Figure 1.2. Often, however, such an explicit characterization is far from straightforward, or even impossible to give. In such a case, we are satisfied with structural results. In particular, a characterization of the structure of the optimal policy would be of great interest. Namely, if one can show that the optimal policy is determined only by a (limited) set of critical states in the model, then the optimal decision rules are of an intuitive and practical nature. Furthermore, the fact that one can restrict oneself to a particular subset of states enables one to compute the optimal decisions recursively, e.g., via the method of successive approximations or via policy iteration.

1.5.2 Control of the arrival process

In order to keep control of the throughput time of jobs and the burdening of resources, we can allow for control of the arrival process of jobs. Here, we distinguish admission control and routing control. In admission control problems, either the arrival rate of jobs may be modified dynamically or jobs may be rejected upon arrival. The first will typically not be an option in our models; cf. Section 1.2.5. The second, on the other hand, will be an option in most of our models. Note that the basic example of Figure 1.2 featured this kind of control. This particular model and several generalizations as well as some other more complex admission control models are discussed by Stidham [52]. These also include a number of routing control problems. In single-station systems, routing control can be seen as a special kind of admission control. It is concerned with the question which resource to assign a newly arrived job to. This is often the main issue in models that feature a

number of parallel servers, each having its own queue. For further treatment of this topic, we refer to Chapter 2. There, a specific routing control problem featuring parallel queues is considered.

1.5.3 Control of the service process

For a given or optimized arrival process, the system can be further optimized by means of control of the service process. Here, we distinguish control of the number of servers, control of the service rates, and control of the service discipline.

If servers can be turned on or off, then we allow for control of the number of servers. The corresponding models are commonly termed *vacation* models, because in these models we can let servers 'go on vacation' for a certain time. As already stressed, in this thesis, we concentrate on scant-capacity models. The workload is typically very high, and a (temporary) removal of servers from the process is basically out of the question. Therefore, this type of control will not be considered any further. Similarly, we do not allow for service rate control in our models either. Servers are expected to work at their nominal speed. Nothing will be gained by working at a slower pace than this nominal speed—on the contrary—and, in our models, 'working faster' will not correspond to actions such as overtime (cf. Section 1.2.2), but to 'providing jobs with less attention than they require or request'. This is the subject of Section 1.6. However, although we do not consider any models with service rate control, we will use the concept of a variable service rate to construct a (near-optimal) heuristic for a particular multi-server system whose analysis turns out to be intractable. This is the subject of Chapter 5.

A type of control we do allow for in some models, is scheduling control. Scheduling problems arise when one may decide in which order to process jobs. In single-station systems, this is of interest when multiple types of jobs can be identified. In this case, it may be clear that any specific order can be established by choosing the service discipline appropriately. This gives rise to so-called priority models. In these models, each job belongs to a certain priority class of jobs. A newly arrived job will immediately overtake all jobs belonging to lower priority classes that were already in the system, awaiting service. Under the PR service discipline (cf. Section 1.3), a new job may also immediately interrupt the service of a lower priority job being served at one of the servers, if there is any, and take over its place at the server. The lower priority job is placed in the queue again—where it must wait until its service

can be resumed—or its service is terminated. A specific model studied by various authors in literature, is the two-class preemptive priority queue. See, e.g., the recent work of Groenevelt et al. [22]. This model concerns a single server serving two customer classes with holding and switching costs. The corresponding control problem involves the objective to switch between classes in such a way that the sum of expected holding and switching costs is minimized. In Chapter 6, we consider a similar model, but with a different cost structure and additional types of control, namely admission as well as *termination* control.

1.6 Termination control

An important characteristic of almost all optimal control problems studied in literature, either with or without admission control, and including all models covered by Section 1.5, is that admission is final, i.e., once new work has been accepted for service, it *must* be processed by the system, and must be processed to a finish, before it can be considered to be out of the system. Models subject to clearing control count as an exception. These are models in which at any time it may be decided to instantaneously remove *all* work content from the system. See, e.g., page 149 of [55]. As stressed before, one of the main problems in many (workflow) operating environments is the lack of capacity to deal with all jobs and to treat all jobs to the full extent. It must be decided which jobs to serve and when to stop. The types of control studied in literature do not cover this type of decision. Clearing, i.e., either removing the complete workload or keeping all work in the system, is far too rigorous. Scant-capacity problems call for a more subtle control with respect to the admission and disposal of jobs.

An initial effort to model the disposal of jobs was made by Xu and Shanthikumar [63], who introduced a new approach for determining the optimal admission control policy in a FCFS M|M|m ordered-entry queueing system with nonidentical servers.[2] The idea of this approach is to construct a dual system: a preemptive LCFS ('Last Come First Served') M|M|m ordered-entry system without admission control, but with *expulsion* control. A system is subject to expulsion control if customers—which may not be denied entry to the system—may be expelled from the system, with the restriction that one can only expel customers—one after another—from

[2] Throughout this thesis, we use the classical notation of Kendall [31] to specify essential system characteristics, which we supplement with new notation wherever convenient.

the end of the queue. The authors show that the two systems induce the same probabilistic behaviour for the departure process and the number of customers in the system under any given policy. Hence, the optimal policy in the original system agrees with its counterpart in the dual system.

Xu [64] employs the dual approach to determine the optimal admission and routing control policy in a FCFS M|M|2 queueing system with nonidentical servers. The corresponding dual system is subject to expulsion and scheduling control. Using the dual approach, Righter [46] extends the results of [64] to an M|M|2 queueing system with nonidentical servers and multiple classes of customers, where preemption is allowed. Further extensions are given to models with finite buffers and models with deadlines for customer service completion.

In the aforementioned literature, expulsion control models are used as a tool rather than a goal. Within the framework of workflow control, expulsion control is too restrictive, since one may only expel a job from the end of the queue and not, for example, the job currently in service. Apart from either serving a job completely or not at all, there is no control of the service times of the jobs in the system.

Johansen and Larsen [28] consider a FCFS single-server one-class workload model in which a key feature of the control policy is its ability to let the service time of a job depend on the actual number of jobs in the system, and to remove jobs from the queue. Each job entering service is assigned a service time in advance, which may not be altered during service. So, service may not be aborted before the pre-assigned service time has elapsed and service may not be extended either.

In this thesis, we introduce the concept of *termination* control, studying a collection of workload models in which the service of a job may be aborted before the job has received full service, and in which work may be removed from the queue as well, at any point in time. This offers a more dynamic service control policy than that of [28]. We will show that there exist optimal threshold policies for both the decision to accept or reject a new job and the decision to continue or abort the service of a job. In some of the models, these results hold under certain regularity conditions, e.g., diminishing marginal returns; cf. Section 1.2.4.

1.7 Methodology

In this section, we discuss the mathematical framework behind the models discussed in Section 1.5 and the models we will consider in this thesis. We then describe the main approach and techniques that will be used in the analysis of these models. Here, our focus is exclusively on the methodology itself. A discussion of computational issues and issues such as the existence of optimal (stationary) policies is beyond our scope.

1.7.1 Markov decision theory

Many dynamic control problems can be formulated as a Markov Decision Problem (MDP). The formulation of such models and the examination of structural properties of such models (e.g., a characterization of the structure of the optimal policy for the model) is the subject of Markov decision theory. Bellman [6] is commonly credited as founder of Markov decision theory. Its essential concepts were formulated in the 40s and 50s, within the framework of sequential game theory. Howard [27] was the first to consider infinite horizon MDPs with average cost criterion. He introduced the policy iteration algorithm, which can be used to compute optimal policies for average cost MDPs. For a fairly complete treatment of Markov decision theory, we refer to Ross [48], Bertsekas [8], and Puterman [44].

The basic ingredients of an MDP are *states*, *actions*, *transitions*, *rewards* and an *objective function*. At fixed and equidistant points in time, the state, $i \in S$ say, of the system is observed and an action $a \in A(i)$ is chosen, where $A(i)$ is the set of all possible actions in state i, and S is the state space. Action a in state i yields a direct reward—possibly in expectation—of $r(i, a)$, and causes the system to make a transition to state j with probability $p_{ij}(a)$. The (fixed) times between transitions are termed *periods*. Note that the next state is drawn from a distribution that depends only on the current state and the action chosen in that state, and not on previous actions and events. This is the Markov property of the process. The models we consider in this thesis will possess this property. We note that in many real-life situations, it is not important *how* a certain state was reached, but *that* it was reached. In fact, the 'how' is often unclear. Furthermore, our focus is on models with complete state observation. This means that the state of the system is available to the decision maker at any time.

Given these basic ingredients, the basic question is how to choose the actions on a dynamic basis such that the objective function reaches a maximum value. We can distinguish between finite horizon and infinite horizon models, and in both cases also between models with and models without discounting. Under reasonable assumptions, and using standard techniques, finite horizon results can be extended to the infinite horizon case.

1.7.2 Dynamic Programming

Using Dynamic Programming (DP), finite horizon MDPs can be solved recursively in n, the number of periods 'to go'. If we let $V_n(i)$ denote the maximum expected return for an n-stage problem starting in state i, and denote by β the discount factor (where $0 < \beta \leq 1$), then

$$V_{n+1}(i) = \max_{a \in A(i)} [r(i,a) + \beta \sum_{j \in S} p_{ij}(a) V_n(j)]. \tag{1.1}$$

Equation (1.1) is a special type of Dynamic Programming Equation (DPE), known as the *optimality equation* for the MDP. It is also referred to as the Bellman equation in literature. $V_n(\cdot)$ is termed the *value function*.

If a control problem can be formulated in terms of a set of DPEs, then DP can be used as a means to prove properties of the optimal control policy, by induction on properties of $V_n(\cdot)$. This is particularly of interest if $V_n(\cdot)$ cannot be explicitly solved for; cf. Section 1.5.1. This approach is known as inductive Dynamic Programming; see, e.g., Hajek [23].

1.7.3 Uniformization

The MDP defined in Section 1.7.1 concerns a *discrete-time* model, i.e., decisions are taken and transitions occur at fixed and equidistant points in time. However, many control problems involve systems which are typically not observed at fixed and equidistant points in time, but *continuously*. For example, in the admission control problem of Figure 1.2, the system is observed at arrival times of jobs, which are generated according to a continuous Poisson process. We concentrate on control problems in which the times between decision epochs are exponential, and whose probabilistic structure is a semi-Markov Decision Process. Characteristic of such a process is that if action a is chosen in state i, then an immediate reward $r(i,a)$ is obtained and, in addition, a cost rate $c(i,a)$ is imposed until the next transition occurs.

If, in addition, the expected times between decision epochs are uniformly bounded, then by allowing transitions that do not result in a change of state, we can obtain that the times between consecutive transitions are exponential with a constant (i.e., state-independent) parameter. This enables us to consider the system as a discrete-time model, and hence enables a representation of the system by means of a set of DPEs. This discretization technique, which is due to Lippman [40], and which was later formalized by Serfozo [50], is termed *uniformization*. Note that after uniformization, periods do not have a fixed length, but have the same length in expectation.

Remark 1.2 Throughout this thesis, in the context of uniformized models, the term 'horizon' (or 'time horizon') is used synonymously with 'number of periods'. Thus, in a finite horizon problem, the expected time span rather than the real time span is considered fixed.

Uniformization also allows for the incorporation of discounting, which is represented by means of a discount rate $\alpha \geq 0$ (instead of a discount factor β). This means that a reward r received at time t has present value $re^{-\alpha t}$. The (exponential) discount rate can be treated as the rate by which the process *vanishes*. See, e.g., the exposition of inductive DP of Walrand [58].

1.7.4 Monotonicity

Inductive DP can be used as a means to obtain certain *monotonicity* properties of the value function $V_n(\cdot)$ which hold for every finite number of periods. Appropriately combined, these monotonicity properties imply certain monotonicity properties of the optimal control policy, e.g., a threshold structure. We now discuss briefly some relevant monotonicity properties of the value function $V_n(\cdot)$. We distinguish between models with a one-dimensional state space and models with a two-dimensional state space.

1.7.4.1 One-dimensional state space models

In continuous-time queueing models with dynamic control, the main (and possibly only) component of the state is usually the number of jobs in the system. Having introduced uniformization, Lippman [40] was the first to use inductive DP to obtain a monotonic characterization of the optimal policy for a specific continuous-time control problem. In fact, he considers

three distinct models, which are all based on models that appeared earlier in literature. In each model, the value function $V_n(i)$ represents the maximum expected n-period reward, starting from state i, where i denotes the number of jobs in the system. Further, in each model, the intended monotonic characterization of the optimal policy is obtained by proving, by means of induction, that $V_n(i)$ is *concave* in i, the number of jobs in the system.[3] We remind that a function $f(x)$ defined on some domain in $\mathbb{N}$ is *concave* if

$$f(x+1) - f(x) \geq f(x+2) - f(x+1) \tag{1.2}$$

for all x for which the four states appearing in the inequality exist. In the models of Lippman, in words, concavity means that the value of an additional job is non-increasing in the number of jobs. This monotonicity property implies a monotonic characterization of the optimal policy in terms of the number of jobs in the system.

For example, let us consider his third model, which concerns an M|M|c queue with finite or infinite buffer capacity. The system features arrival rate control: on a dynamic basis, the arrival rate may be chosen from some closed subset A of $[0, \bar{\lambda}]$, where $\bar{\lambda} < \infty$. A reward r_λ is received when a job arrives in a time interval in which the arrival rate is λ. In addition, there are holding costs $h(i)$ per unit of time when there are i jobs in the system. It is assumed that $h(i)$ is non-decreasing and convex, and that r_λ is continuous and non-increasing on A, with $r_{\bar{\lambda}} \geq 0$ and $r_0 < \infty$. Finally, denote by $\lambda_n(i)$ the optimal arrival rate when the current state is i and n periods remain. By induction on n, Lippman shows that $V_n(i)$ is concave in i for every n. From the concavity of $V_n(i)$, and the DPE for the control problem, it is obtained that $\lambda_n(i)$ is non-increasing in i. Put informally, this monotonic characterization says that the more crowded the system becomes, the lower the selected arrival rate will be.

1.7.4.2 Two-dimensional state space models

In control models in which the dimension of the state space is larger than 1, concavity by itself will not be a sufficient condition to establish the desired threshold results. However, in two-dimensional state space models,

[3] Probably in order to be consistent with the original models, Lippman actually considers the control problem in the second and third model as a *minimization* problem, and establishes *convexity* of the value function. Clearly, this is equivalent to establishing concavity in the equivalent maximization problem.

concavity can be complemented by *submodularity* (see Topkis [56]) to obtain monotonicity properties of the optimal policy. A function $f(x_1, x_2)$ defined on some domain in $\mathbb{N} \times \mathbb{N}$ is said to be *submodular* if

$$f(x_1, x_2 + 1) - f(x_1, x_2) \geq f(x_1 + 1, x_2 + 1) - f(x_1 + 1, x_2) \tag{1.3}$$

for all x_1, x_2 for which the four states appearing in the inequality exist.

One of the first to apply inductive DP to a two-dimensional model was Davis [17]. He considers a queueing system with two identical exponential servers in parallel, each with its own queue, and independent and identically distributed interarrival times. The system is controlled by means of admission control: an arriving job may be either rejected, admitted to queue 1, or admitted to queue 2, based on the state $\underline{x} = (x_1, x_2)$ at the time of arrival, where x_j is the number of jobs at queue j, including the position at the server, $j = 1, 2$. Jobs admitted to the system generate a reward r, which is received upon entrance, and there are non-decreasing and convex holding costs $h_j(x_j)$ per unit of time when there are x_j jobs at queue j, $j = 1, 2$. Defining $V_n(\underline{x})$ as the maximum expected n-period reward, starting from state $\underline{x}$, Davis gives an inductive proof to show that the value function satisfies the condition

$$V_n(\underline{x} + \underline{e}_j) - V_n(\underline{x}) \quad \geq \quad V_n(\underline{x} + \underline{e}_i + \underline{e}_j) - V_n(\underline{x} + \underline{e}_i) \tag{1.4}$$

for $i, j = 1, 2$, where $\underline{e}_1 := (1, 0)$ and $\underline{e}_2 := (0, 1)$. Note that (1.4) comprises submodularity (if $i \neq j$) and componentwise concavity (if $i = j$) of $V_n(\underline{x})$.

To make an inductive proof work, two additional conditions are added to condition (1.4), namely,

$$V_n(\underline{x} + \underline{e}_1 + \underline{e}_2) - V_n(\underline{x} + \underline{e}_2) \quad \geq \quad V_n(\underline{x} + 2\underline{e}_1) - V_n(\underline{x} + \underline{e}_1), \tag{1.5}$$

$$V_n(\underline{x} + \underline{e}_1 + \underline{e}_2) - V_n(\underline{x} + \underline{e}_1) \quad \geq \quad V_n(\underline{x} + 2\underline{e}_2) - V_n(\underline{x} + \underline{e}_2). \tag{1.6}$$

Combined, inequalities (1.5) and (1.6) state that $V_n(\underline{x})$ is *subconcave*, after the following definition, used by Ghoneim and Stidham [53]. A function $f(x_1, x_2)$ defined on some domain in $\mathbb{N} \times \mathbb{N}$ is said to be *subconcave in* x_1 if

$$f(x_1 + 1, x_2 + 1) - f(x_1, x_2 + 1) \geq f(x_1 + 2, x_2) - f(x_1 + 1, x_2) \tag{1.7}$$

for all x_1, x_2 for which the four states appearing in the inequality exist, and is said to be *subconcave in* x_2 if

$$f(x_1 + 1, x_2 + 1) - f(x_1 + 1, x_2) \geq f(x_1, x_2 + 2) - f(x_1, x_2 + 1) \tag{1.8}$$

for all x_1, x_2 for which the four states appearing in the inequality exist. If $f(x_1, x_2)$ is subconcave in both x_1 and x_2, then $f(x_1, x_2)$ is called *subconcave*.

Together, inequalities (1.4), (1.5) and (1.6) imply that the optimal policy has a threshold structure. In particular, the optimal policy is admission monotonic as well as routing monotonic: (1.4) implies that if it is optimal to reject in state $\underline{x}$, then it is also optimal to reject in state $\underline{x} + \underline{e}_j$, $j = 1, 2$, and (1.5) [(1.6)] implies that if admitting to queue 1 [queue 2] is preferable to admitting to queue 2 [queue 1] in state $\underline{x}$, then this remains the case in state $\underline{x} + \underline{e}_2$ [$\underline{x} + \underline{e}_1$].

Since the late 70s, following the early work of Davis and the like, numerous two-dimensional models have been studied in literature. However, attempts to generalize structural results to higher than two dimensions have almost always been unsuccessful, and usually fail. Some examples are given by Stidham [53]. The question as of *why* higher-dimensional models are generally intractable is not easily answered. However, it may be clear that by increasing the number of state components, the number of inequalities will increase rapidly, and additional (dimension-dependent) monotonicity properties, beyond submodularity and subconcavity, will be required to establish a monotonic characterization of the optimal policy via inductive DP. This is an intriguing subject to explore. Some results have been established by Koole [35], who studies higher-dimensional tandem queues. However, in this thesis, we confine ourself to models that can be formulated as (semi-)MDPs with a two-dimensional state space. As will be demonstrated in Chapter 2, DP already faces limitations in two dimensions.

In our inductive proofs of particular monotonicity properties, such as submodularity of the value function $V_n(\cdot)$, we will frequently make use of the following universal lemma. Here, and in the remainder of this thesis, $V_n(i; a)$ is generally defined as the maximum expected n-period (α-discounted) reward when the current state is i, and given authorized (but not necessarily optimal) decision a in that state.

Lemma 1.1 *Let $s_m \in S$ for $m = 1, \ldots, 4$, and $\phi \in A(s_1)$ and $\psi \in A(s_4)$. Then*

$$V_n(s_1; \phi) - V_n(s_2) \geq V_n(s_3) - V_n(s_4; \psi) \tag{1.9}$$

implies

$$V_n(s_1) - V_n(s_2) \geq V_n(s_3) - V_n(s_4). \tag{1.10}$$

Proof. Immediate from $V_n(s_1) \geq V_n(s_1;\phi)$ and $V_n(s_4) \geq V_n(s_4;\psi)$ for all ϕ, ψ.

□

We will use Lemma 1.1 in the following way. When distinguishing between all possible combinations of optimal decisions in certain states s_2 and s_3, we choose ϕ and ψ such that (1.9) holds. Then (1.10) holds as well. This enables us to consider *self-selected*, appropriate decisions in states s_1 and s_4, instead of decisions which are necessarily optimal.

Finally, in our inductive proofs, we will occasionally make use of *sample path* arguments. For an exposition of the sample path approach, we refer to Liu et al. [41], and El-Taha and Stidham [19]. Our sample path arguments rely on the use of stochastic coupling of processes. Wherever such arguments appear in this thesis, a separate proof based on inductive DP could most probably have been given instead, yet a sample path approach seemed more convenient or insightful to us.

1.8 Outline of the thesis

The remainder of this thesis is organized as follows. In Chapter 2, we consider two models that feature admission control, and in particular routing control, but not yet termination control. More specifically, we consider two closely related systems, both consisting of two parallel sub-systems to which arriving jobs must be routed. There are dedicated and flexible arrivals, and both systems are subject to blocking. Considering the objective to minimize the total number of blocked jobs, we show for both systems that the optimal routing control policy has a threshold structure. We also show that 'Least Loaded Routing' is the optimal routing policy if the system is symmetrical. The analysis conducted in Chapter 2 already demonstrates the capabilities as well as some 'incapabilities' (i.e., limitations) of our inductive DP approach.

Subsequently, in Chapter 3, we extend the dynamic control structure to include termination control. We introduce the notion of termination control by means of studying an M|E$_N$|1 one-class queueing model in which the service of a job may be aborted before the job has received full service, and in which jobs may be removed from the queue as well, at any point in time. Under certain regularity conditions on the cost and reward structure, we

derive various monotonicity properties of the value function and show that there exist optimal threshold policies for both the decision to accept or reject a new job and the decision to continue or abort the service of a job.

In Chapter 4, we discuss several extensions of the basic dynamic control model considered in Chapter 3. These include batch arrivals, phase-type arrivals, and a more general service process in which a job that completes its current phase is automatically routed to some downstream phase, according to a Markov feed-forward routing mechanism. By means of inductive DP, we derive generalized monotonicity and threshold results for each of these extensions. We also capture the structure of the optimal policy for a purely discrete-time model in which the workload of a job is the sum of at most N geometric service phases and in which the state of the system is observed at deterministic decisions epochs.

In Chapter 5, we discuss a multi-server version of the $\mathrm{M}|\mathrm{E}_N|1$ model studied in Chapter 3. This multi-server extension proves to be analytically as well as computationally intractable. Although some basic monotonicity properties can be derived, our focus is mainly on numerical aspects surrounding this $\mathrm{M}|\mathrm{E}_N|s$ queue and its optimal control policy. In particular, we present a heuristic for the computation of the optimal policy for this multi-server model. The heuristic is based on a closely related model, namely, a slightly modified version of the single-server model studied in Chapter 3, whose optimal policy is readily computed. We evaluate and refine the heuristic by means of a numerical study. The results of this study indicate that our heuristic yields near-optimal performance.

Thus far, we considered one-class models only. One-class models are suitable in situations where, *upon arrival*, jobs are mutually indistinguishable. With this, individual job characteristics such as the actual service requirements and the outcome of a job may vary from job to job (cf. Section 1.2.1), but such distinctions will only become clear during the service process. If, on the other hand, new jobs can be classified into distinct classes of jobs, based on characteristics that can already be recognized before any capacity engagement (again, cf. Section 1.2.1), then one-class models cannot be used to accurately represent the system, and multi-class models are called for.

In Chapter 6, we first consider a two-class $\mathrm{M}^{\lambda_1,\lambda_2}|\mathrm{M}^{\mu}|1$ preemptive priority queue. The system has the same admission and termination control features as the model studied in Chapter 3. This means that one has the option to either accept or reject new type-1 or type-2 jobs, and, at any time, one has

the option to remove any number of type-1 or type-2 jobs from the system. We show that there exist optimal threshold policies for these two types of decisions. Subsequently, under certain restrictions on the cost structure or (admission) control structure, we extend our results to the multi-server case.

Each of the Chapters 2 through 6 concludes with a brief summary of the results obtained, a discussion of some straightforward or rather improbable extensions, as well as some suggestions for further research. In addition, the thesis concludes with a separate chapter, in which we make some final remarks and indicate some natural directions for further research. In this chapter, we also discuss the relation between the models we studied in this thesis and a general framework for the derivation of monotonicity properties using inductive DP, as developed by Koole [34]. We give a brief overview of this unified treatment, and indicate how our models fit into this framework.

2

Two routing control problems

In this chapter, which is based on Brouns [13], we consider two closely related systems, both featuring two parallel sub-systems to which arriving jobs must be routed. Both systems are subject to blocking. In particular, the first system concerns two parallel exponential servers, each having its own finite capacity queue. The second system concerns two parallel Erlang loss (sub-)systems, i.e., each sub-system has its own set of parallel exponential servers and there is no waiting room at any of the sub-systems or servers. Both systems feature *dedicated* and *flexible* arrivals. Dedicated arrivals automatically join a particular sub-system, whereas flexible arrivals may join and be routed to either sub-system. Considering the objective to minimize the total number of blocked jobs, we show for both systems that the optimal routing control policy has a threshold structure. We also show that *Least Loaded Routing* is the optimal routing policy if the system is symmetrical.

The analysis of the second model is less straightforward than that of the first model, which—from an analytical point of view—serves mainly as a set-up for the second model. In the light of the issue of deficient resource capacity, the first model is of little practical interest, because it puts focus on a lack of *buffer* capacity rather than *service* capacity. The second model originates from literature on telecommunications network analysis and design (as will be touched upon in the next section). However, its use can be expanded to more general deficient-capacity environments. This will be illustrated further on in this chapter. Before describing and analysing our models in detail, we give an overview of literature on the routing of jobs to parallel queues that is relevant in the context of our models.

2.1 Literature on routing to parallel queues

There is a vast literature on the routing of jobs to parallel queues. We refer to Hariharan et al. [24] for an extensive overview. The perhaps most basic routing control problem in parallel queues is studied by Winston [62], who considers a queueing system consisting of a finite number of identical exponential servers in parallel, each having its own queue. Jobs arrive at the system according to a Poisson process. Upon arrival of a job it must be assigned to one of the queues. Under the assumption that jockeying between queues is not permitted, it is shown that the *shortest line discipline* is optimal in terms of maximizing throughput.

Hordijk and Koole [26] prove that the shortest line discipline maximizes stochastically the number of jobs served at any time t when the queues have finite buffers. The servers are assumed identical but the buffers may have different capacities. Towsley et al. [57] also consider identical servers and finite buffers with unequal capacities. They also allow for buffer space to be available at the controller, which enables the controller to delay the routing of a job to one of the parallel queues.

Koole et al. [37] show that the shortest line discipline is optimal with respect to various cost functions. Their results cover systems with two parallel queues with infinite or finite capacity, arrivals that are independent of the state of the system but otherwise arbitrary, and ILR ('Increasing Likelihood Ratio') service time distributions, including the exponential distribution.

Koole [36] studies the static assignment of jobs to parallel, exponential, heterogeneous servers. There is no waiting room at any of these servers, and blocked jobs are lost. The objective is to minimize the average number of blocked jobs. In the case of dynamic assignment it is optimal to route to the fastest available server; see Koole [33]. The static version of the problem is formulated as a stochastic control problem with partial observation. Numerical experiments are conducted and the structure of the optimal policy is studied.

Johri [29] considers state-dependent service rates. Under certain regularity conditions on these rates, the shortest line discipline minimizes stochastically the number of jobs at any time t. Menich and Serfozo [42] allow the service and arrival rates to be functions of all queue lengths. This includes the case of a number of parallel service facilities, each having s identical exponential servers. They do not allow for finite buffers.

Hajek [23] considers two interacting parallel stations with two servers at each station and a fifth server that is shared by the two stations. Both stations have an infinite capacity queue. There are three Poisson arrival streams: two *dedicated* streams and one *flexible* stream. Jobs in the first dedicated stream always join station 1 and jobs in the second dedicated stream always join station 2. Jobs in the flexible stream may join either queue. So for each arriving flexible job it must be decided to which queue it is routed.

By combining the models of [23] and [42], and by disallowing buffering, we can obtain systems with a number of parallel M|M|s|s sub-systems and dedicated as well as flexible arrivals. These are highly suitable for modelling wireless networks; see Alanyali and Hajek [4]. Such a network consists of a number of base stations and of users. The users require communication channels, which are available at the base stations. A station may only serve users that are within geographical range of the station. Users may be in range of several stations and the resource allocation problem concerns the question of station selection. If each location has finite capacity, i.e., a finite number of channels, then a consumer is lost if upon its arrival all channels of all stations in its neighbourhood are already in use. The goal of the allocation policy is to minimize the fraction of lost consumers. The authors provide a lower bound for the consumer loss probability under any allocation policy. Structural properties of the optimal policy are not addressed.

For the specific case of two parallel stations with c_1 and c_2 channels, respectively, and exp(μ)-distributed service times at any of the $c_1 + c_2$ channels, Van Leeuwaarden et al. [39] consider various optimal static routing policies. They also consider dynamic routing, for which they discuss a one-step policy improvement algorithm and its performance. They conclude with a brief discussion of three open problems, the first and second of which are of interest to us. In particular, their first open problem, although intuitively clear, is to show that the optimal routing policy for the model, termed *Model II* in the remainder of this chapter, is of a switch-over type, i.e., has a threshold structure. We will prove this conjecture. However, we will first establish the same threshold result for a closely related model, termed *Model I*, and then extend this property to *Model II*.

Model I will be described in Section 2.2. In Section 2.3, we state and prove our main results for this model. In Section 2.4, we shift our attention to *Model II*. Its description is taken from [39]. In Section 2.5, the threshold characterization obtained for *Model I* is extended to *Model II*.

The second open problem posed in [39] is to show that *Least Loaded Routing* is the optimal routing policy in the symmetrical case, i.e., in the case of equal dedicated arrival rates and equal capacities. This property is readily obtained as a corollary of the threshold characterization of the optimal policy for the general model.

2.2 Model description *Model I*

We consider the queueing system depicted in Figure 2.1. The system features two identical parallel servers. Service times are exp(μ)-distributed. The servers have separate queues, with finite capacity. Station 1 is formed by server 1 and its queue. Station 2 is formed by server 2 and its queue. The maximum number of jobs at station 1 is $c_1 \geq 1$ and the maximum number of jobs at station 2 is $c_2 \geq 1$. So the buffer sizes of stations 1 and 2 are $c_1 - 1$ and $c_2 - 1$, respectively. Note that we use the term 'station' to indicate either of the two sub-systems the system consists of. Nonetheless, in conformity with what has been said in Section 1.3, the system *as a whole* can still be regarded as a single station, as jobs visit at most one of the two sub-systems.

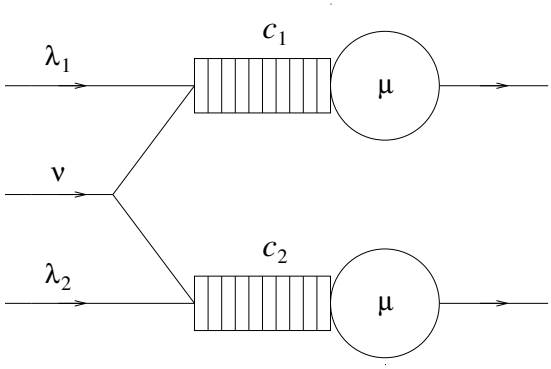

Figure 2.1: Queueing system corresponding to *Model I*

There are three arrival streams: two dedicated streams and one flexible stream. Jobs in dedicated stream k ($k = 1, 2$) arrive according to a Poisson process with rate λ_k and automatically join station k. Jobs in the flexible stream arrive according to a Poisson process with rate ν and may join either station. Upon arrival of a flexible job it must be decided to which of the two stations it is routed. We assume that the decision maker has complete

information, i.e., he knows the number of jobs at each of the two stations. The structure of the system is that of a (semi-)Markovian decision process. It can be described as follows.

States: The state of the system is described by the tuple (i, j), where i $(0 \leq i \leq c_1)$ is the number of jobs at the first station and j $(0 \leq j \leq c_2)$ is the number of jobs at the second station.

Events: We distinguish two possible events: (i) the arrival of a new job and (ii) a service completion.

Decisions: If the event is an arrival and it concerns a flexible job, then it has to be decided to which of the two stations the job is routed: decision '1' if station 1, decision '2' if station 2. If the event is an arrival and it concerns a dedicated job from stream 1 (or stream 2), then decision '1' (or decision '2') is taken automatically. If the decision is such that the arriving job is routed to a station that is loaded to capacity, then the job leaves the system immediately. If the event is a service completion, then no decision has to be taken.

Costs and rewards: If an arriving job is routed to a station that is loaded to capacity, then *blocking* costs of 1 are incurred. Alternatively, one can say that blocking yields a reward of -1. These are the only costs; there are no holding costs for jobs residing in the system.

Criterion: The objective is to minimize the expected (blocking) costs (i.e., number of blocked jobs) over an n-period time horizon. Alternatively stated, the objective is to maximize the expected reward over an n-period time horizon.

Uniformization: Applying uniformization (see Section 1.7.3), we can consider that transitions occur at the jump times of a Poisson process with rate $\lambda_1 + \lambda_2 + \nu + 2\mu$. By scaling time, we take $\lambda_1 + \lambda_2 + \nu + 2\mu = 1$ without loss of generality. Then, with probability λ_k $(k = 1, 2)$ a transition concerns the arrival of a dedicated job from stream k, with probability ν it concerns the arrival of a flexible job, with probability μ a service completion at station 1 and with the same probability a service completion at station 2. A service completion is either a real service completion or an *artificial* service completion when the server idles because there are no jobs at the station.

Uniformization enables us to use inductive Dynamic Programming to prove our results for any finite time horizon. Using a standard argument, these results can then be extended to the infinite time horizon case for the criterion

of average cost per unit of time. See, e.g., Denardo [18]. Note that since c_1 and c_2 are finite, the system is a finite state system.

2.2.1 Dynamic Programming formulation

We will now complete the model in terms of a mathematical formulation. After that, we successively state and prove our main results.

Recapitulating, i and j denote the number of jobs at station 1 and 2, respectively, and (i, j) is the state of the system, $0 \leq i \leq c_1$ and $0 \leq j \leq c_2$. We will use the following notation:

- $V_n(i, j)$ denotes the maximum expected n-period reward when the current state is (i, j). State (i, j) may be the result of an arrival—where the system is observed immediately after the new job has been routed to one of the two stations—or a real or artificial service completion.
- $V_n(i, j; \pi)$ denotes the maximum expected n-period reward when the current state is (i, j), given that there is an arrival event at this point in time and given that decision π is chosen with respect to the new job; $\pi = \mathtt{1}$ if this job belongs to dedicated stream 1, $\pi = \mathtt{2}$ if the job belongs to dedicated stream 2 and $\pi \in \{\mathtt{1}, \mathtt{2}\}$ if the job is a flexible job. Let π^* denote the optimal decision. Note that in the notation π^* the dependence on i, j and n is suppressed.

Further, for any condition Φ, define $\mathbf{1}[\Phi] := \begin{cases} 1 & \text{if } \Phi, \\ 0 & \text{else.} \end{cases}$

Then our model is defined by the following set of DPEs.

For $n \geq 0$ and all $0 \leq i \leq c_1$ and $0 \leq j \leq c_2$:

$$
\begin{aligned}
V_0(i,j) &= 0 \\
V_{n+1}(i,j) &= \lambda_1 V_n(i,j;\mathtt{1}) + \lambda_2 V_n(i,j;\mathtt{2}) + \\
&\qquad \nu \, \mathbf{max}\{V_n(i,j;\mathtt{1}),\ V_n(i,j;\mathtt{2})\} + \\
&\qquad\quad \mu V_n(\mathbf{max}\{i-1,0\},j) + \mu V_n(i,\mathbf{max}\{j-1,0\}) \\
V_n(i,j;\mathtt{1}) &= -\mathbf{1}[i = c_1] + V_n(\mathbf{min}\{i+1, c_1\}, j) \\
V_n(i,j;\mathtt{2}) &= -\mathbf{1}[j = c_2] + V_n(i, \mathbf{min}\{j+1, c_2\})
\end{aligned}
$$

Remark 2.1 The vertical line in front of the set of DPEs indicates that the equalities represent *programming* equations, i.e., program code that can be used to generate instances of the model that can be fed to an optimization program. Throughout this thesis, we will make use of this notation whenever we write down a set of DPEs or a single DPE.

2.3 Main results for *Model I*

We will prove the following theorem.

Theorem 2.1 {Characterization of the optimal routing policy} *For any remaining number of periods n, the optimal routing policy can be characterized as follows. If it is optimal to route an arriving flexible job to station 1 in state (i, j), then it is optimal as well to route it to station 1 in all states $(i, j + k)$ with $0 < k \leq c_2 - j$ and in all states $(i - k, j)$ with $0 < k \leq i$.*

Remark 2.2 Alternatively stated, Theorem 2.1 reads that if it is optimal to route an arriving flexible job to station 2 in state (i, j), then it is optimal as well to route it to station 2 in all states $(i + k, j)$ with $0 < k \leq c_1 - i$ and in all states $(i, j - k)$ with $0 < k \leq j$.

In terms of a graphical representation, Theorem 2.1 states that the optimal routing policy can be characterized by a switch-over curve in the shape of a non-decreasing step-function, so that for every i there exists a threshold j of i and for every j there exists a threshold i of j. The following example provides such a graphical representation of the structure of a typical routing policy.

Example 2.1 Consider the following instance of our model: $\lambda_1 = 0.2$, $\lambda_2 = 0.1$, $\nu = 0.2$, $\mu = 0.25$, $c_1 = 18$ and $c_2 = 12$. The average reward optimal routing policy for this system is depicted below. We employed the successive approximation algorithm to calculate the optimal policy. The desired relative and absolute accuracy of 10^{-4} was reached after 1,034 iterations, and the optimal average blocking costs per unit of time are approximately 0.0198.

For comparison, the optimal *static* policy (which has no state information and which routes an arriving flexible job to station 1 with probability p

and to station 2 with probability $1-p$) incurs average blocking costs of approximately 0.0324 per unit of time. This minimal loss is obtained for $p \approx 0.253$.

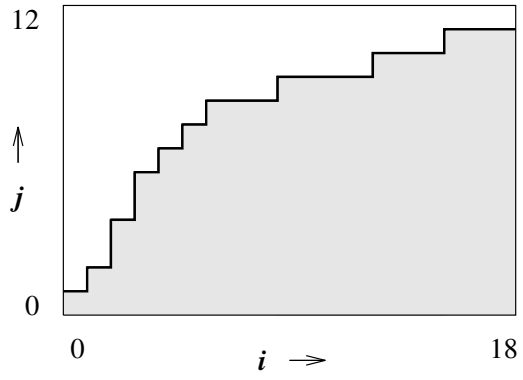

Figure 2.2: Optimal routing policy for Example 2.1

The white area covers states in which it is optimal to route an arriving flexible job to station 1, whereas the shaded area covers states in which it is optimal to route an arriving flexible job to station 2. Note that the switch-over curve—the solid polyline in the figure—is indeed non-decreasing.

2.3.1 The line of proof

In order to establish Theorem 2.1, we will prove the following monotonicity results, from which Theorem 2.1 will be obtained as a corollary.

Proposition 2.1 {Key Proposition}
For $n \geq 0$,

$$V_n(i,j+1) - V_n(i,j+2) \geq V_n(i+1,j) - V_n(i+1,j+1), \quad (2.1)$$

$$V_n(i+1,j) - V_n(i+1,j+1) \geq V_n(i,j) - V_n(i,j+1), \quad (2.2)$$

$$V_n(i+1,j) - V_n(i+2,j) \geq V_n(i,j+1) - V_n(i+1,j+1), \quad (2.3)$$

for all i,j *for which the four states appearing in the respective inequality exist (i.e., (2.1) holds for all* $0 \leq i < c_1$ *and* $0 \leq j < c_2 - 1$*, (2.2) for all* $0 \leq i < c_1$ *and* $0 \leq j < c_2$*, and (2.3) for all* $0 \leq i < c_1 - 1$ *and* $0 \leq j < c_2$*).*

Together, inequalities (2.1) and (2.3) state that our value function is subconcave; cf. (1.7) and (1.8). Note that (2.1) and (2.3) correspond to properties (c2) and (c1) of [23], respectively, in which $V_n(\cdot)$ is the minimum expected cost, so in our maximization model the inequality signs are read the other way around. Further, inequality (2.2) states that our value function is submodular; cf. (1.3). Note that (2.2) corresponds to property (b) of [23], although the latter is more general.

Remark 2.3 Combining (2.1) with (2.2) for $i = c_1 - 1$, and combining (2.3) with (2.2) for $j = c_2 - 1$, we obtain, for $n \geq 0$,

$$V_n(c_1, j+1) - V_n(c_1, j+2) \geq V_n(c_1, j) - V_n(c_1, j+1), \tag{2.4}$$
$$V_n(i+1, c_2) - V_n(i+2, c_2) \geq V_n(i, c_2) - V_n(i+1, c_2), \tag{2.5}$$

where (2.4) holds for all $0 \leq j < c_2 - 1$ and (2.5) for all $0 \leq i < c_1 - 1$.

2.3.2 Proof of the Key Proposition

The proof of the Key Proposition uses induction on the remaining number of periods and runs as follows. *Step 0:* Observe that (2.1), (2.2) and (2.3) hold for $n = 0$. *Step 1:* Assuming (2.1), (2.2) and (2.3) to hold for some $n \geq 0$, prove that (2.1), (2.2) and (2.3) hold for $n + 1$ as well. Note that it suffices to prove (2.1) and (2.2) for $n+1$. Namely, once these two inequalities have been established, (2.3) follows by interchanging the names of the two stations (i.e., station 1 is now termed station 2 and vice versa).

In Step 1 of the proof we will make use of Lemma 1.1. In addition, we will need the following proposition, which contains inequalities of the form $V_n(\cdot) \leq 1 + V_n(\cdot)$. The idea to use such inequalities is also found in [26].

Proposition 2.2 *For $n \geq 0$,*

$$0 \leq V_n(i,j) - V_n(i,j+1) \leq 1, \qquad 0 \leq i \leq c_1,\ 0 \leq j < c_2, \tag{2.6}$$
$$0 \leq V_n(i,j) - V_n(i+1,j) \leq 1, \qquad 0 \leq i < c_1,\ 0 \leq j \leq c_2. \tag{2.7}$$

Proof. We use coupling and a sample path argument (although an inductive proof could also be given). We first consider the right-hand inequality of (2.6). Consider two n-period instances of our model, instance $\mathcal{I}_0$ starting in (i, j) and instance $\mathcal{I}_1$ starting in $(i, j+1)$. We couple all events and all decisions. Instance $\mathcal{I}_0$ follows the optimal policy and instance $\mathcal{I}_1$ copies all

decisions taken in $\mathcal{I}_0$. In particular, if $\mathcal{I}_0$ idles at station 2 because it has run out of jobs at that station, then $\mathcal{I}_1$ takes its additional job into service.

Then the costs are the same for both instances as long as $\mathcal{I}_0$ does not route a new job to station 2 in some state $(k, c_2 - 1)$ for $0 \leq k \leq c_1$. If this does not occur before time hits zero, then the difference in reward is 0. Now suppose it does occur before time zero, at time T say. If, even earlier in time, at time T' say, $\mathcal{I}_1$ completed service at station 2 while $\mathcal{I}_0$ was idling at that station, then $\mathcal{I}_0$ and $\mathcal{I}_1$ became identical at T', and hence their difference in reward is 0. Alternatively, assume $\mathcal{I}_0$ has not witnessed an artificial service completion at station 2 before T. Then, at T, the job routed to station 2 in $\mathcal{I}_0$ is blocked in $\mathcal{I}_1$, incurring a reward of -1 for $\mathcal{I}_1$ and causing $\mathcal{I}_0$ and $\mathcal{I}_1$ to become identical immediately afterwards. So the difference in reward between $\mathcal{I}_0$ and $\mathcal{I}_1$ is $0 - (-1) = 1$.

The reasoning is almost the same for the left-hand inequality of (2.6). Again, let instance $\mathcal{I}_0$ start in (i, j) and instance $\mathcal{I}_1$ in $(i, j+1)$. But now let $\mathcal{I}_1$ follow the optimal policy and let $\mathcal{I}_0$ copy all decisions taken in $\mathcal{I}_1$. In particular, if $\mathcal{I}_1$ starts serving its additional job at station 2, then $\mathcal{I}_0$ idles at station 2.

Finally, (2.7) follows from (2.6) by interchanging the names of the two stations.

□

We note that the left-hand inequalities of (2.6) and (2.7) correspond to property (a) of [23]. We further note that one may easily verify that from Proposition 2.2 the following intermediate result can be obtained; cf. Remark 3 in [39].

Corollary 2.1 {The optimal routing policy is greedy}
For any n, the optimal routing policy will route an arriving flexible job to station 1 if $i < c_1$ and $j = c_2$ and to station 2 if $i = c_1$ and $j < c_2$.

Remark 2.4 For Corollary 2.1 to hold, it is essential that all blocked jobs incur the same costs, irrespective of which of the arrival streams they come from. Namely, if these costs are heterogeneous, then blocking on purpose may be beneficial in order to keep buffer space available for jobs from another stream, which have higher blocking costs associated with them. See Section 2.3.4 for a concise treatment of heterogeneous blocking costs.

We now return to the Key Proposition.

Proof of the Key Proposition.

Step 0. Inequalities (2.1), (2.2) and (2.3) hold by definition for $n = 0$.

Induction hypothesis. Assume that for some $n \geq 0$, (2.1) holds for all $0 \leq i < c_1$ and $0 \leq j < c_2 - 1$, (2.2) for all $0 \leq i < c_1$ and $0 \leq j < c_2$ and (2.3) for all $0 \leq i < c_1 - 1$ and $0 \leq j < c_2$. This will be our induction hypothesis.

Step 1. Under the induction hypothesis, we show that (2.1) and (2.2) hold for $n + 1$ as well. Then (2.3) also holds for $n + 1$.

In the derivation below, and throughout this thesis, we will make use of the following notation, which we have adopted in part from the calculus of logic. In deductions, arguments will be put between braces ("{" and "}") and will be written in sans serif typeface. If the induction hypothesis is used as an argument, then we will either write down its full name, or use the abbreviation "ih" to save space.

Further, to make compound arguments more comprehensible, we will make use of special symbols to indicate individual terms. Symbols made up of hollow dots enfolding black numbers (e.g., ①) will be used for terms on the left-hand side of an (in)equality, whereas solid black symbols enfolding white numbers (e.g., ❶) will be used for terms on the right-hand side of the (in)equality. In addition to this, we use the (standard) notation $A \Rightarrow B$ to indicate that some property A implies some property B, and $A \equiv B$ to indicate that A is equivalent to B.

We now return to Step 1 of our proof.

Proof of (2.1). Let $0 \leq i < c_1$ and $0 \leq j < c_2 - 1$. Then,

$$
\begin{aligned}
& V_{n+1}(i, j+1) - V_{n+1}(i, j+2) \\
&\quad = \mu[V_n(\mathbf{max}\{i-1, 0\}, j+1) - V_n(\mathbf{max}\{i-1, 0\}, j+2)]\,① \; + \\
&\qquad \mu[V_n(i, j) - V_n(i, j+1)]\,② \; + \\
&\qquad\quad \lambda_1[V_n(i, j+1; \mathtt{1}) - V_n(i, j+2; \mathtt{1})]\,③ \; + \\
&\qquad\qquad \lambda_2[V_n(i, j+1; \mathtt{2}) - V_n(i, j+2; \mathtt{2})]\,④ \; + \\
&\qquad\qquad\quad \nu[\mathbf{max}\{V_n(i, j+1; \mathtt{1}),\ V_n(i, j+1; \mathtt{2})\} - \\
&\qquad\qquad\qquad \mathbf{max}\{V_n(i, j+2; \mathtt{1}),\ V_n(i, j+2; \mathtt{2})\}]\,⑤ \\
&\quad \geq \{\mathsf{ih};\ ① \geq ❶;\ ② \geq ❷;\ ③ \geq ❸;\ ④ \geq ❹;\ ⑤ \geq ❺;\ \mathsf{see\ below}\} \\
&\qquad \mu[V_n(i, j) - V_n(i, j+1)]\,❶ \; + \\
&\qquad\quad \mu[V_n(i+1, \mathbf{max}\{j-1, 0\}) - V_n(i+1, j)]\,❷ \; + \\
&\qquad\qquad \lambda_1[V_n(i+1, j; \mathtt{1}) - V_n(i+1, j+1; \mathtt{1})]\,❸ \; + \\
&\qquad\qquad\quad \lambda_2[V_n(i+1, j; \mathtt{2}) - V_n(i+1, j+1; \mathtt{2})]\,❹ \; + \\
&\qquad\qquad\qquad \nu[\mathbf{max}\{V_n(i+1, j; \mathtt{1}),\ V_n(i+1, j; \mathtt{2})\} - \\
&\qquad\qquad\qquad\quad \mathbf{max}\{V_n(i+1, j+1; \mathtt{1}),\ V_n(i+1, j+1; \mathtt{2})\}]\,❺ \\
&\quad = V_{n+1}(i+1, j) - V_{n+1}(i+1, j+1).
\end{aligned}
$$

$\boxed{① \geq ❶}$ By (2.1) if $i > 0$, and by (2.1) and subsequently (2.2) if $i = 0$.

$\boxed{② \geq ❷}$ By (2.1) if $j > 0$, and by (2.6) if $j = 0$.

$\boxed{③ \geq ❸}$ By executing decision $\mathtt{1}$ and subsequently by (2.1) if $i < c_1 - 1$, and by (2.4) if $i = c_1 - 1$.

$\boxed{④ \geq ❹}$ By executing decision $\mathtt{2}$ and subsequently by (2.6) if $j = c_2 - 2$, and by (2.1) if $j < c_2 - 2$.

$\boxed{⑤ \geq ❺}$ For ⑤ $\geq$ ❺, we need to show that

$$
V_n(i, j+1; \pi^*) - V_n(i, j+2; \pi^*) \quad \geq \quad V_n(i+1, j; \pi^*) - V_n(i+1, j+1; \pi^*). \tag{2.8}
$$

The optimal decision $\pi^* =: d_1$ corresponding to $V_n(i, j+2; \pi^*)$ is either $\mathtt{1}$ or $\mathtt{2}$. The same can be said about the optimal decision $\pi^* =: d_2$ corresponding to $V_n(i+1, j; \pi^*)$. There are at most four joint cases (d_1, d_2), namely: $(\mathtt{1}, \mathtt{1})$, $(\mathtt{1}, \mathtt{2})$, $(\mathtt{2}, \mathtt{1})$ and $(\mathtt{2}, \mathtt{2})$. We will show that (2.8) holds for each case.

Under $(1,1)$, we choose 1 in the other two states appearing in (2.8) as well. Then the result follows by ③ $\geq$ ❸ and Lemma 1.1. Analogously, under $(2,2)$, we choose 2 in the other two states as well, after which the result follows by ④ $\geq$ ❹ and Lemma 1.1.

Thirdly, under $(1,2)$,

$$\begin{aligned} V_n(i,j+1;1) - V_n(i,j+2;1) &= V_n(i+1,j+1) - V_n(i+1,j+2) \\ &= V_n(i+1,j;2) - V_n(i+1,j+1;2), \end{aligned}$$

to which we subsequently apply Lemma 1.1.

Fourthly, under $(2,1)$,

$$\begin{aligned} &V_n(i,j+1;2) - V_n(i,j+2;2) \\ &\quad = V_n(i,j+2) - V_n(i,\mathbf{min}\{j+3,c_2\}) + \mathbf{1}[j=c_2-2] \\ &\quad \geq \{\text{induction hypothesis; (2.1) twice if } j<c_2-2 \text{ and } i<c_1-1; \\ &\qquad \text{(2.1), (2.4) if } j<c_2-2 \text{ and } i=c_1-1; \text{ (2.6) if } j=c_2-2\} \\ &\qquad V_n(\mathbf{min}\{i+2,c_1\},j) - V_n(\mathbf{min}\{i+2,c_1\},j+1) \\ &\quad = V_n(i+1,j;1) - V_n(i+1,j+1;1), \end{aligned}$$

to which we subsequently apply Lemma 1.1.

This concludes our proof of (2.1) for $n+1$.

Proof of (2.2). Let $0 \le i < c_1$ and $0 \le j < c_2$. Then,

$$
\begin{aligned}
& V_{n+1}(i+1,j) - V_{n+1}(i+1,j+1) \\
&\quad = \mu[V_n(i,j) - V_n(i,j+1)]\,① \;+ \\
&\qquad \mu[V_n(i+1,\mathbf{max}\{j-1,0\}) - V_n(i+1,j)]\,② \;+ \\
&\qquad\quad \lambda_1[V_n(i+1,j;1) - V_n(i+1,j+1;1)]\,③ \;+ \\
&\qquad\qquad \lambda_2[V_n(i+1,j;2) - V_n(i+1,j+1;2)]\,④ \;+ \\
&\qquad\qquad\quad \nu[\mathbf{max}\{V_n(i+1,j;1),\ V_n(i+1,j;2)\} - \\
&\qquad\qquad\qquad\quad \mathbf{max}\{V_n(i+1,j+1;1),\ V_n(i+1,j+1;2)\}]\,⑤ \\
&\quad \ge \{\mathsf{ih};\ ① \ge ❶;\ ② \ge ❷;\ ③ \ge ❸;\ ④ \ge ❹;\ ⑤ \ge ❺;\ \mathsf{see\ below}\} \\
&\qquad \mu[V_n(\mathbf{max}\{i-1,0\},j) - V_n(\mathbf{max}\{i-1,0\},j+1)]\,❶ \;+ \\
&\qquad\quad \mu[V_n(i,\mathbf{max}\{j-1,0\}) - V_n(i,j)]\,❷ \;+ \\
&\qquad\qquad \lambda_1[V_n(i,j;1) - V_n(i,j+1;1)]\,❸ \;+ \\
&\qquad\qquad\quad \lambda_2[V_n(i,j;2) - V_n(i,j+1;2)]\,❹ \;+ \\
&\qquad\qquad\qquad \nu[\mathbf{max}\{V_n(i,j;1),\ V_n(i,j;2)\} - \\
&\qquad\qquad\qquad\qquad \mathbf{max}\{V_n(i,j+1;1),\ V_n(i,j+1;2)\}]\,❺ \\
&\quad = V_{n+1}(i,j) - V_{n+1}(i,j+1).
\end{aligned}
$$

$\boxed{① \ge ❶}$ By (2.2) if $i > 0$, and with equality if $i = 0$.

$\boxed{② \ge ❷}$ By (2.2) if $j > 0$, and by $② = ❷ = 0$ if $j = 0$.

$\boxed{③ \ge ❸}$ By executing decision $\mathsf{1}$ and subsequently with equality if $i = c_1 - 1$, and by (2.2) if $i < c_1 - 1$.

$\boxed{④ \ge ❹}$ By executing decision $\mathsf{2}$ and subsequently by (2.2) if $j < c_2 - 1$, and by $④ = ❹ = 1$ if $j = c_2 - 1$.

$\boxed{⑤ \ge ❺}$ Analogous to the proof of $⑤ \ge ❺$ for (2.1) for $n+1$, we distinguish the cases $(\mathsf{1},\mathsf{1})$, $(\mathsf{2},\mathsf{2})$, $(\mathsf{1},\mathsf{2})$ and $(\mathsf{2},\mathsf{1})$ for the optimal decisions d_1 and d_2 corresponding to $V_n(i+1,j+1;\pi^*)$ and $V_n(i,j;\pi^*)$, respectively. Again, the first case can be dealt with by choosing $\mathsf{1}$ in the other two states as well, and the second by choosing $\mathsf{2}$ in the other two states as well.

Thirdly, under $(\mathtt{1},\mathtt{2})$,

$$\begin{aligned}
&V_n(i+1,j;\mathtt{2}) - V_n(i+1,j+1;\mathtt{1})\\
&\quad = \quad V_n(i+1,j+1) - V_n(\mathbf{min}\{i+2,c_1\},j+1) + \mathbf{1}[i=c_1-1]\\
&\quad \geq \quad \{\text{induction hypothesis; (2.2), (2.3) if } i<c_1-1;\ \text{(2.7) if } i=c_1-1\}\\
&\qquad\quad V_n(i,j+1) - V_n(i+1,j+1)\\
&\quad = \quad V_n(i,j;\mathtt{2}) - V_n(i,j+1;\mathtt{1}),
\end{aligned}$$

to which we subsequently apply Lemma 1.1.

Fourthly, under $(\mathtt{2},\mathtt{1})$,

$$\begin{aligned}
&V_n(i+1,j;\mathtt{2}) - V_n(i+1,j+1;\mathtt{2})\\
&\quad = \quad V_n(i+1,j+1) - V_n(i+1,\mathbf{min}\{j+2,c_2\}) + \mathbf{1}[j=c_2-1]\\
&\quad \geq \quad \{\text{induction hypothesis; (2.1), (2.2) if } j<c_2-1 \text{ and } i<c_1-1;\\
&\qquad\qquad \text{(2.4) if } j<c_2-1 \text{ and } i=c_1-1;\ \text{(2.6) if } j=c_2-1\}\\
&\qquad\quad V_n(i+1,j) - V_n(i+1,j+1)\\
&\quad = \quad V_n(i,j;\mathtt{1}) - V_n(i,j+1;\mathtt{1}),
\end{aligned}$$

to which we subsequently apply Lemma 1.1.

This concludes our proof of (2.2) for $n+1$ and hence our proof of the Key Proposition.

□

We now derive Theorem 2.1 from the Key Proposition by means of two corollaries. Note that Corollary 2.3 is exactly Theorem 2.1.

Corollary 2.2 *For $n \geq 0$,*

$$\begin{aligned}
V_n(i,j;\mathtt{2}) - V_n(i,j+1;\mathtt{2}) \quad &\geq \quad V_n(i,j;\mathtt{1}) - V_n(i,j+1;\mathtt{1}),\\
&\qquad\qquad 0 \leq i \leq c_1,\ 0 \leq j < c_2, \qquad (2.9)\\
V_n(i,j;\mathtt{1}) - V_n(i+1,j;\mathtt{1}) \quad &\geq \quad V_n(i,j;\mathtt{2}) - V_n(i+1,j;\mathtt{2}),\\
&\qquad\qquad 0 \leq i < c_1,\ 0 \leq j \leq c_2. \qquad (2.10)
\end{aligned}$$

Proof. One may easily verify that (2.9) follows from (2.6) for $0 \leq i \leq c_1$ and $j = c_2 - 1$, from (2.1) for $0 \leq i < c_1$ and $0 \leq j < c_2 - 1$, and from (2.4) for $i = c_1$ and $0 \leq j < c_2 - 1$. Analogously, (2.10) follows from (2.7) for $i = c_1 - 1$ and $0 \leq j \leq c_2$, from (2.3) for $0 \leq i < c_1 - 1$ and $0 \leq j < c_2$, and from (2.5) for $0 \leq i < c_1 - 1$ and $j = c_2$.

□

Corollary 2.3 *Let $n \geq 0$, $0 \leq i \leq c_1$ and $0 \leq j \leq c_2$. If it is optimal to route an arriving flexible job to station 1 in state (i, j), then it is optimal to route it to station 1 in state $(i, j+1)$, provided $j < c_2$, and in state $(i-1, j)$, provided $i > 0$.*

Proof. Let $n \geq 0$. It suffices to show that

$$V_n(i,j;1) \geq V_n(i,j;2) \implies V_n(i,j+1;1) \geq V_n(i,j+1;2), \quad 0 \leq i \leq c_1,\ 0 \leq j < c_2, \tag{2.11}$$

$$V_n(i,j;1) \geq V_n(i,j;2) \implies V_n(i-1,j;1) \geq V_n(i-1,j;2), \quad 0 < i \leq c_1,\ 0 \leq j \leq c_2. \tag{2.12}$$

It is easily verified that implications (2.11) and (2.12) are immediate from inequalities (2.9) and (2.10), respectively.

□

From Corollary 2.3 we can also obtain the following result, which states that in case of equal dedicated arrival rates and equal capacities, Least Loaded Routing, i.e., routing to the station with the least number of jobs, is the optimal routing policy. In *Model I*, this policy can also be referred to as the shortest line discipline.

Corollary 2.4 {Optimality of Least Loaded Routing}
Assume $\lambda_1 = \lambda_2$ and $c_1 = c_2 =: c$. Then, for any n, Least Loaded Routing is the optimal routing policy.

Proof. By symmetry it suffices to consider states (i, j) with $0 \leq i \leq j \leq c$. Let $n \geq 0$. Clearly, $V_n(i,i;1) = V_n(i,i;2)$ by symmetry for any $0 \leq i \leq c$. Hence, by (2.11), we also have $V_n(i,j;1) \geq V_n(i,j;2)$ for $0 \leq i \leq j \leq c$, i.e., it is optimal to route to station 1 in (i, j).

□

2.3.3 Extension to heterogeneous service rates

In our model we considered homogeneous service rates, i.e., the service rates at stations 1 and 2 are both equal to μ. Now assume the service rates are heterogeneous and equal to μ_1 and μ_2, respectively, where $\mu_1 \neq \mu_2$. Then, by replacing each occurrence of μ in the DPEs and all proofs by μ_1 or μ_2 (depending on which of the two is applicable there), it is readily verified that all results and proofs remain intact.

An even more general extension would be to consider heterogeneous service rates that also depend on the origin of jobs, i.e., when a job enters service, its rate of service depends on which of the three arrival streams the job stems from. However, assuming that the service order is not necessarily restricted to FCFS, this implies essentially a four-dimensional state space, involving states of the form (x_1, y_1, x_2, y_2), where x_k is the number of dedicated jobs at station k and y_k is the number of flexible jobs at station k ($k = 1, 2$). This will result in an intractable model as far as the derivation of threshold properties is concerned. The situation is even worse in case of FCFS, because then we also have to keep track of the order in which jobs arrive.

2.3.4 Heterogeneous blocking costs

In Remark 2.4, we noted that in case of heterogeneous blocking costs, intentional blocking at some station (when it is loaded to capacity) may be beneficial in order to keep buffer space available at the other station, hence reducing the probability of blocking dedicated jobs at that other station. Below, we discuss briefly three classes of routing policies that can be associated with general blocking costs of b_1 for jobs from dedicated stream 1, b_2 for jobs from dedicated stream 2 and b for jobs from the flexible stream.

Non-greedy routing Corollary 2.1 need no longer hold if intentional blocking is permitted (as already noted in Remark 2.4). The threshold structure described in Theorem 2.1 is lost as well, in general, as illustrated by the following basic counterexample.

Counterexample 2.1 For any model instance satisfying $\lambda_1 = \lambda_2$, $c_1 = c_2$, and $b_2 > b_1 > b = 0$, the average reward optimal routing policy when there are no jobs at station 1 is to route a flexible job to station 1 if $j < c_2$, but to station 2 (hence blocking it, at no cost) if $j = c_2$.

Compulsory greedy routing Suppose that a flexible job may not be routed to a station whose buffer is full if there is still buffer space available at the other station. So, a flexible job may only be blocked if both stations are loaded to capacity. Then the following counterexample shows that the threshold structure described in Theorem 2.1 need not hold any more.

Counterexample 2.2 Consider the following model instance: $\lambda_1 = 0.2$, $\lambda_2 = 0.35$, $\nu = 0.15$, $\mu = 0.15$, $c_1 = 6$, $c_2 = 10$, $b_1 = 2$, $b_2 = 0.5$ and $b = 0.1$. Computations reveal that the average reward optimal routing policy for this system when there are no jobs at station 1 is to route a flexible job to station 2 if $j = 0$, to station 1 if $1 \leq j \leq 8$, and to station 2 again if $j = 9$, violating Theorem 2.1. (Note that routing to station 1 is compulsory when $j = 10$.)

Unrestricted routing and termination In the models discussed so far, queued jobs had to be served. Now suppose that at any time we may decide to discard queued jobs from the system. Blocking a job (whether on purpose or not) and disposing of a job essentially both mean that the job is not going to be served, or at least not to completion. Assuming that the costs of not serving are the prime interest, and that blocking costs were introduced to represent these, it will be reasonable to assume that disposing of a job can be carried out at the tariff of blocking.

In *Model I*, this extended type of control implies that the state space becomes four-dimensional (involving states of the form (x_1, y_1, x_2, y_2), similar to the situation of stream-dependent service rates described in Section 2.3.3), rendering the model intractable. A related—yet more basic—model with multiple arrival streams and the option to remove queued jobs from the system is considered and analysed in Chapter 6.

This completes our discussion of *Model I*.

2.4 Model description *Model II*

We now shift our attention to *Model II*. Consider the system depicted in Figure 2.3. The system consists of two parallel Erlang loss stations. The first has $c_1 \geq 1$ parallel servers and the second has $c_2 \geq 1$ parallel servers. All $c_1 + c_2$ servers are identical, and service times are $\exp(\mu)$-distributed. Since the stations are loss stations, there is no queueing. The maximum number of jobs at station 1 is c_1 and the maximum number of jobs at station 2 is c_2, as in *Model I*. Note that we use the term 'station' again to indicate either of the two sub-systems the system is composed of.

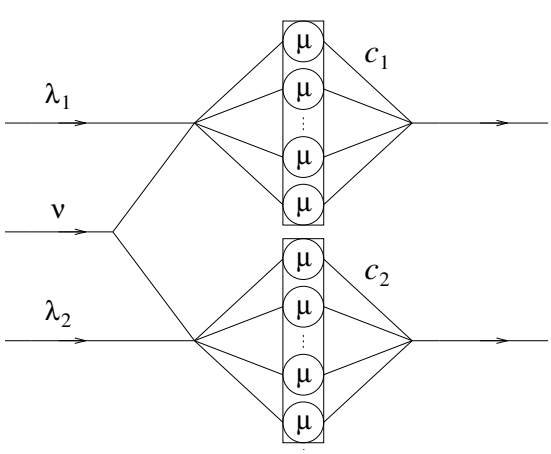

Figure 2.3: Queueing system corresponding to *Model II*

As in *Model I*, the arrival process is made up of three Poisson arrival streams: two dedicated streams (characterized by arrival rates λ_1 and λ_2) and one flexible stream (characterized by arrival rate ν). Upon arrival of a flexible job it must be decided to which of the two stations it is routed. The decision maker has complete information, i.e., he knows the number of jobs at each of the two stations.

The structure of the system is that of a (semi-)Markovian decision process, and its description is almost identical to that of *Model I*. We distinguish the same states, events, decisions, rewards of -1 and assume the same criterion, i.e., maximizing the expected reward over an n-period time horizon. The only essential difference lies in the total service rate of a station, which is $\mu \cdot \mathbf{1}[i > 0]$ in *Model I* when there are i jobs present at that station, and which now becomes $i\mu$. Consequently, also the uniformization is slightly different. We take $\lambda_1 + \lambda_2 + \nu + (c_1 + c_2)\mu = 1$ without loss of generality. Then, with probability $(i + j)\mu$ there is a real service completion, whereas with probability $(c_1 + c_2 - (i + j))\mu$ there is an artificial service completion.

Remark 2.5 Although *Model II* is tailored to base station versus user allocation problems in wireless networks (as discussed earlier in this chapter), its use is by no means limited to it. To give just an impression of what other areas of application one could think of, we mention engineering or consultancy offices, where orders (*jobs*) have to be assigned to specialized project teams (*servers*). Some orders are fairly general (*flexible*), whereas

others are of a highly specialized nature (*dedicated*) and hence require the attention of one particular team. A new order has to be considered a loss if there is no suitable project team available at the time of request, since running projects usually require a substantial amount of time and clients will place their order elsewhere if it cannot be taken on in the direct future. The objective of the office is to maximize the number of acquired projects.

2.4.1 Dynamic Programming formulation

The Dynamic Programming formulation for *Model II* is almost the same as for *Model I*. We employ the same notation, i.e., the same value functions $V_n(i,j)$ and $V_n(i,j;\pi)$, and we only need to modify the DPE for $V_{n+1}(i,j)$. Namely, under the convention that $i\mu V_n(i-1,j)$ is zero for $i=0$ and any j, and $j\mu V_n(i,j-1)$ is zero for $j=0$ and any i, we have, for $n \geq 0$ and all $0 \leq i \leq c_1$ and $0 \leq j \leq c_2$:

$$\begin{aligned} V_{n+1}(i,j) \;=\; & \lambda_1 V_n(i,j;\mathtt{1}) \;+\; \lambda_2 V_n(i,j;\mathtt{2}) \;+ \\ & \nu\, \mathbf{max}\{V_n(i,j;\mathtt{1}),\; V_n(i,j;\mathtt{2})\} \;+ \\ & i\mu V_n(i-1,j) \;+\; j\mu V_n(i,j-1) \;+ \\ & (c_1+c_2-(i+j))\mu V_n(i,j) \end{aligned}$$

2.5 Main results for *Model II*

We claim that Theorem 2.1 and Propositions 2.1 and 2.2 (and thus Corollaries 2.1, 2.2, 2.3 and 2.4) all remain intact. Consequently, the optimal routing policy for *Model II* has a threshold structure (and Least Loaded Routing is the optimal routing policy in the symmetrical case). Note that it suffices to show that Propositions 2.1 and 2.2 remain valid.

Proposition 2.3 *Proposition 2.2 holds for Model II as well. I.e., in words, the expected difference in reward between some instance of Model II and another instance starting with one job extra lies between 0 and 1.*

Proof. It can easily be verified that Proposition 2.2 holds for *Model II* as well. The proof is analogous to the proof given for *Model I*. For example, consider the right-hand inequality of (2.6). Consider two n-period instances

of *Model II*, instance $\mathcal{I}_0$ starting in (i, j) and instance $\mathcal{I}_1$ starting in $(i, j+1)$. We couple all events and all decisions, and also all servers. Instance $\mathcal{I}_0$ follows the optimal policy and instance $\mathcal{I}_1$ copies all decisions taken in $\mathcal{I}_0$. Let S denote the server that is idle in $\mathcal{I}_0$ at time n and serving a job in $\mathcal{I}_1$ at time n.

Then the costs are the same for both instances as long as $\mathcal{I}_0$ does not route a new job to server S. If this does not occur before time hits zero, then the difference in reward is 0. Now suppose it does occur before time zero, at time T say. If, even earlier in time, at time T' say, $\mathcal{I}_0$ witnessed an artificial service completion at server S and, consequently, $\mathcal{I}_1$ completed service at server S, then $\mathcal{I}_0$ and $\mathcal{I}_1$ became identical at T', and hence their difference in reward is 0. Alternatively, assume $\mathcal{I}_0$ has not witnessed an artificial service completion at server S before T. Then, at T, the job routed to server S in $\mathcal{I}_0$ is blocked in $\mathcal{I}_1$, incurring a reward of -1 for $\mathcal{I}_1$ and causing $\mathcal{I}_0$ and $\mathcal{I}_1$ to become identical immediately afterwards. So the difference in reward between $\mathcal{I}_0$ and $\mathcal{I}_1$ is $0 - (-1) = 1$.

□

Proposition 2.4 *The Key Proposition (i.e., Proposition 2.1) holds for Model II as well. I.e., in words, the value function of Model II is submodular and subconcave.*

Proof. We follow the proof of the Key Proposition for *Model I*, i.e., we perform the same steps and take the same induction hypothesis. Step 0 is trivial. For *Model I*, for both (2.1) and (2.2), Step 1 of the proof consisted of 5 parts, namely, establishing ① $\geq$ ❶ through ⑤ $\geq$ ❺. One may easily verify that the last three parts (③ $\geq$ ❸, ④ $\geq$ ❹ and ⑤ $\geq$ ❺), i.e., the parts concerning the three arrival processes, remain intact for *Model II*, since we have the same induction hypothesis and the same DPEs at arrival times as for *Model I*.

It remains to show that ① + ② $\geq$ ❶ + ❷ for the terms ①, ②, ❶ and ❷ corresponding to (2.1) and (2.2) for $n+1$ for *Model II*. These are the terms concerning the departure processes from stations 1 and 2. (Again, (2.3) will follow from (2.1) by interchanging the names of the two stations.)

We first consider (2.1). Let $0 \leq i < c_1$ and $0 \leq j < c_2 - 1$. Then,

$$
\begin{aligned}
①+② &= i\mu V_n(i-1,j+1) \ + \ (j+1)\mu V_n(i,j) \ + \\
&\quad (c_1+c_2-(i+j+1))\mu V_n(i,j+1) \ - \ [i\mu V_n(i-1,j+2) \ + \\
&\quad (j+2)\mu V_n(i,j+1) \ + \ (c_1+c_2-(i+j+2))\mu V_n(i,j+2)] \\
&= \{\text{rearrange terms}\} \\
&\quad i\mu[V_n(i-1,j+1)-V_n(i-1,j+2)]\,①_1 \ + \\
&\quad\quad \mu[V_n(i,j+1)-V_n(i,j+2)]\,①_2 \ + \\
&\quad\quad\quad (c_1-(i+1))\mu[V_n(i,j+1)-V_n(i,j+2)]\,①_3 \ + \\
&\quad\quad\quad\quad j\mu[V_n(i,j)-V_n(i,j+1)]\,②_1 \ + \\
&\quad\quad\quad\quad\quad \mu[V_n(i,j)-V_n(i,j+1)]\,②_2 \ + \\
&\quad\quad\quad\quad\quad\quad \mu[V_n(i,j+1)-V_n(i,j+1)]\,②_3 \ + \\
&\quad\quad\quad\quad\quad\quad\quad (c_2-(j+2))\mu[V_n(i,j+1)-V_n(i,j+2)]\,②_4 \\
&\geq \{\text{ih};\ ①_1 \geq ❶_1,\ ①_3 \geq ❶_3,\ ②_1 \geq ❷_1,\ ②_4 \geq ❷_4 \text{ by (2.1)}; \\
&\quad\quad ②_3 = ❷_2 = 0;\ ②_2 = ❶_2;\ ①_2 \geq ❷_3 \text{ by (2.1)}\} \\
&\quad i\mu[V_n(i,j)-V_n(i,j+1)]\,❶_1 \ + \\
&\quad\quad \mu[V_n(i,j)-V_n(i,j+1)]\,❶_2 \ + \\
&\quad\quad\quad (c_1-(i+1))\mu[V_n(i+1,j)-V_n(i+1,j+1)]\,❶_3 \ + \\
&\quad\quad\quad\quad j\mu[V_n(i+1,j-1)-V_n(i+1,j)]\,❷_1 \ + \\
&\quad\quad\quad\quad\quad \mu[V_n(i+1,j)-V_n(i+1,j)]\,❷_2 \ + \\
&\quad\quad\quad\quad\quad\quad \mu[V_n(i+1,j)-V_n(i+1,j+1)]\,❷_3 \ + \\
&\quad\quad\quad\quad\quad\quad\quad (c_2-(j+2))\mu[V_n(i+1,j)-V_n(i+1,j+1)]\,❷_4 \\
&= \{\text{rearrange terms}\} \\
&\quad (i+1)\mu V_n(i,j) \ + \ j\mu V_n(i+1,j-1) \ + \\
&\quad (c_1+c_2-(i+j+1))\mu V_n(i+1,j) \ - \ [(i+1)\mu V_n(i,j+1) \ + \\
&\quad (j+1)\mu V_n(i+1,j) \ + \ (c_1+c_2-(i+j+2))\mu V_n(i+1,j+1)] \\
&= ❶+❷.
\end{aligned}
$$

Remark 2.6 In the derivation above we have used explicitly that the service rates of the servers at station 1 are equal to the service rates of the servers at station 2.

Next, consider (2.2). Let $0 \leq i < c_1$ and $0 \leq j < c_2$. Then,

$$\begin{aligned}
① &= i\mu[V_n(i,j) - V_n(i,j+1)]\,①_1 + \\
&\quad \mu[V_n(i,j) - V_n(i,j+1)]\,①_2 + \\
&\quad\quad (c_1 - (i+1))\mu[V_n(i+1,j) - V_n(i+1,j+1)]\,①_3 \\
&\geq \{\text{induction hypothesis; } ①_2 = ❶_2;\ ①_1 \geq ❶_1,\ ①_3 \geq ❶_3 \text{ by (2.2)}\} \\
&\quad i\mu[V_n(i-1,j) - V_n(i-1,j+1)]\,❶_1 + \\
&\quad\quad \mu[V_n(i,j) - V_n(i,j+1)]\,❶_2 + \\
&\quad\quad\quad (c_1 - (i+1))\mu[V_n(i,j) - V_n(i,j+1)]\,❶_3 \\
&= ❶,
\end{aligned}$$

and

$$\begin{aligned}
② &= j\mu[V_n(i+1,j-1) - V_n(i+1,j)]\,②_1 + \\
&\quad \mu[V_n(i+1,j) - V_n(i+1,j)]\,②_2 + \\
&\quad\quad (c_2 - (j+1))\mu[V_n(i+1,j) - V_n(i+1,j+1)]\,②_3 \\
&\geq \{\text{induction hypothesis; } ②_2 = ❷_2 = 0;\ ②_1 \geq ❷_1,\ ②_3 \geq ❷_3 \text{ by (2.2)}\} \\
&\quad j\mu[V_n(i,j-1) - V_n(i,j)]\,❷_1 + \\
&\quad\quad \mu[V_n(i,j) - V_n(i,j)]\,❷_2 + \\
&\quad\quad\quad (c_2 - (j+1))\mu[V_n(i,j) - V_n(i,j+1)]\,❷_3 \\
&= ❷,
\end{aligned}$$

so $① \geq ❶$ and $② \geq ❷$, and thus $① + ② \geq ❶ + ❷$.

This concludes our proof of the Key Proposition for *Model II*.

□

2.5.1 Extension to heterogeneous service rates

In Section 2.3.3, we noted that all monotonicity results remain valid in *Model I* if the two stations have heterogeneous (but stream-independent) service rates. One may ask if this is also the case in *Model II*, i.e., when each of the c_1 servers at station 1 has service rate μ_1 and each of the c_2 servers at station 2 has service rate μ_2, where $\mu_1 \neq \mu_2$. As far as the threshold characterization is concerned, this remains an *open problem*. Our approach for (2.1) will not work directly; cf. Remark 2.6. In fact, (2.1) need not even hold in case $\mu_2 > \mu_1$. See the following counterexample.

Counterexample 2.3 Consider the following instance of *Model II* with heterogeneous service rates: $\lambda_1 = \lambda_2 = 0$, $\nu = \frac{5}{17}$, $\mu_1 = \frac{1}{17}$, $\mu_2 = \frac{3}{17}$ and $c_1 = c_2 = 3$. We have calculated the average reward optimal routing policy for this instance. The desired relative and absolute accuracy of 10^{-5} was reached after 116 iterations. We found

$$V_{116}(2,1) - V_{116}(2,2) = 0.094 < 0.097 = V_{116}(3,0) - V_{116}(3,1),$$

which violates inequality (2.1).

Despite the fact that (2.1) does not hold in general if $\mu_2 > \mu_1$, the threshold structure of the optimal policy may still very well apply, because Proposition 2.1 is not a *necessary* condition for Theorem 2.1. For example, the optimal policy for the instance considered in Example 2.3 (clearly) is to route to station 1 only if station 2 is loaded to capacity, so it is still of a threshold type.

Conjecture 2.1 *The threshold characterization of the optimal routing policy as given by Theorem 2.1 still holds for Model II if* $\mu_1 \neq \mu_2$.

It is important to note that any proof of the conjecture must follow a different approach than the general approach based on the notion of submodularity and subconcavity employed in this thesis, because, in general, subconcavity is lost when $\mu_1 \neq \mu_2$ (as demonstrated by Counterexample 2.3).

2.6 Conclusions

We have studied two closely related dynamic control models, both concerning queueing systems with two parallel sub-systems to which arriving jobs must be routed. The models featured routing control, but not yet termination control. For both models we have shown that the optimal routing control policy has a threshold structure. In addition, we have discussed two types of extensions, namely, heterogeneous blocking costs and heterogeneous (station-dependent) service rates. We have seen that, in general, the monotonicity and threshold results no longer remain valid in case of heterogeneous blocking costs. In the first model, the main monotonicity and threshold properties extend to the case of station-dependent service rates. By means of a counterexample we have

shown that in the second model, one of the main monotonicity properties need no longer hold if the service rates are station-dependent, revealing that the inductive DP approach already faces limitations in two dimensions.

A natural direction for further research would be to investigate the extension of both models to more than two sub-systems. Note, however, that this will essentially imply a higher-dimensional state space, i.e., m-dimensional in case of m sub-systems, $m > 2$, and that monotonicity properties beyond submodularity and subconcavity must be established to obtain a monotonic characterization of the optimal policy via inductive DP; cf. Section 1.7.4.2.

3

An M|E_N|1 queue with admission and termination control

In this chapter, which is based on Brouns and Van der Wal [10], we consider our first model incorporating both admission and termination control in the decision structure. In particular, we study an M|E_N|1 queue with FCFS service order and two on-line decision features. One type of decision concerns the arrivals: for each arrival we have to decide to either accept or reject it. The other type, termed termination control, concerns the decision to either continue the service of a job or to abort it. We will show that under some regularity conditions on the cost and reward functions, both the optimal admission control policy and the optimal termination control policy have a threshold structure.

The chapter is organized as follows. In Section 3.1, we describe the model in detail. Section 3.2 gives an overview of the results and the line of proof. Section 3.3 gives the (lengthy) proofs for the finite horizon case. Section 3.4 discusses the extension to the infinite horizon. Section 3.5 contains some counterexamples that illustrate the necessity of the regularity conditions we imposed. Finally, in Section 3.6, we discuss briefly a slightly modified version of the model, which is based on an alternative reward structure.

3.1 Model description

The basic model we study in this chapter is a single-server queueing system, operating according to the FCFS discipline and possessing infinite buffer capacity. New jobs arrive at the system according to a Poisson process with

arrival rate $\lambda \geq 0$. The workload of a job is Erlang distributed with $N \geq 1$ phases. Each phase takes an exponential time with mean $1/\mu$. The system is controlled in two ways: one has to decide to accept or reject new arrivals and one has to decide to continue or abort the job in service. See also Figure 3.1. We assume that the decision maker has complete information, i.e., he knows the number of jobs present and the number of (exponential) phases already completed for the job in service. The structure of the system is that of a (semi-)Markovian decision process. It can be described as follows.

States: The state of the system is described by the tuple (i, k), where i is the number of jobs in the system and k denotes the number of phases completed for the job in service. If $i = 0$, then k is indefinite. To indicate this, the empty system is denoted by $(0, \cdot)$. We also use the intermediate state (i, k, arr) immediately after an arrival. A newly arrived job is considered not to be part of the system until after it has been accepted.

We will also say that a job being served 'resides in node k' if it has passed through k service phases and, consequently, is in its $(k+1)$th phase, where $0 \leq k < N$. After the completion of this $(k+1)$th phase, the job moves on from node k to node $k+1$, provided the job is not aborted. The maximum number of service phases is N.

Events: We distinguish two possible events: (i) the arrival of a new job and (ii) the completion of a service phase.

Decisions: If the event is an arrival, then first the decision has to be taken to accept or reject the newly arrived job. This changes the state (i, k, arr) into $(i+1, k)$ or (i, k). Next, it is decided to either continue or abort service of the job in service. If the abort decision is taken, then the job is discarded from the system and service cannot be resumed later. One may discard more than one job at a time, so we allow for so-called *multi-aborts*. The reward structure will be such that in case of an abort one always aborts the job in service first; cf. Remark 3.3. If the event is a service phase completion, then only the continue/(multi-)abort decision has to be taken.

Costs and rewards: At any time one may decide to abort the job in service. The reward for a job depends on the number of service phases that have been completed. The reward corresponding to the completion of k phases is $r(k)$, $0 \leq k \leq N$, where $k = 0$ means the job has not (yet) passed the first phase, possibly because it has not been taken into service at all. Note that we have chosen a reward structure in which rewards are paid when a job leaves the system and not when a job completes a phase.

Apart from these rewards there are holding costs for the jobs residing in the system, either waiting to be taken into service or being served. We assume these costs are linear in the number of jobs, namely, $ih \geq 0$ per unit of time when there are i jobs present. (However, our results will also hold in the more general case of non-decreasing, convex holding costs $h(i) \geq 0$; see Section 3.3.1.) In addition, each time a job is admitted to the system, *consideration* costs $c \geq 0$ are incurred. If an arriving job is not admitted, then a (possibly negative) reward $r(0)$ is earned. So, jobs that are terminated before they have passed through the first service phase receive the same reward $r(0)$ as jobs that are rejected. See also Figure 3.1.

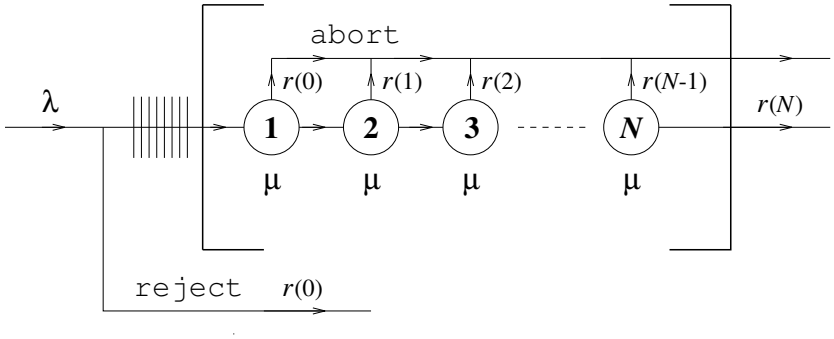

Figure 3.1: $M|E_N|1$ queue with admission and termination control

Discounting: We discount at a rate $\alpha \geq 0$, i.e., rewards and costs at time t are to be multiplied by $\exp(-\alpha t)$. We treat the discount rate α as the rate by which the process vanishes. In other words, the process will live for an exponential time with rate α, after which there will be no more arrivals, service phase completions, rewards or costs.

Criterion: The objective is to maximize the expected (discounted) reward over an n-period time horizon. We allow $\lambda > \mu/N$, as well as $\lambda = 0$. In the latter case, there is an initial number of jobs that await service, i.e., a batch, without any future arrivals.

Uniformization: The system evolves at arrival times, at service phase completion times, and eventually at the time the process vanishes. Applying the uniformization method, we can consider that transitions occur at the jump times of a Poisson process with rate $\lambda + \mu + \alpha > 0$. By scaling time, we take $\lambda + \mu + \alpha = 1$ without loss of generality. Then, with probability

$\lambda \geq 0$ a transition concerns an arrival, with probability $\mu > 0$ it concerns a service phase completion and with probability $\alpha \geq 0$ the process vanishes. A service phase completion is either a real service phase completion or an *artificial* service phase completion if the server idles, e.g., because the system is empty. If the system is indeed empty, then the state, which is $(0, \cdot)$, remains unaltered and we continue service by definition.

Uniformization enables us to use inductive DP to prove our results for any finite time horizon. These results can then be extended to the infinite time horizon case. This will be discussed in Section 3.4.

Remark 3.1 For technical reasons and notational simplicity, we specify that if a job has received maximum service and, as a consequence, has reached final node N (so the system is in some state (i, N)), then both the abort and the continue decision are allowed. Continuing implies that the server idles deliberately in the next period. Continuing in (i, N) also leads to artificial service phase completions. See Remark 3.2 for further comments.

We make the following important assumption; cf. Section 1.2.4.

Fundamental Assumption 3.1
{Structure of the reward function}
The reward function $r(k)$ is non-decreasing and concave in k.

We note (once more) that 'non-decreasing' means that putting more work into a job does not leave us with a lower overall reward for this job, and that 'concave' means that the longer we work on a job the less rewarding it becomes to continue. This implies that aborting the job in service in order to continue with a job awaiting service can be interesting. All rewards are assumed to be finite.

Example 3.1 Suppose the service phases represent independent checks without precedence constraints; see also Example 4.1. For each case, it may be decided dynamically how many of a maximum of N checks are performed, where check k yields an expected reward of $u(k)$. Then we can achieve that Fundamental Assumption 3.1 holds by arranging the checks in non-increasing order of $u(k)$, and by taking $r(k) = \sum_{j=1}^{k} u(j)$ for $0 \leq k \leq N$.

Remark 3.2 The assumption that the reward function is non-decreasing is essentially an implication of the assumption that the reward function is concave. If for some k we would have $r(k+1) < r(k)$ (and thus, because of the concavity, $r(k+l) < r(k)$ for all $l > 0$), then idling in k would always

be better than working in k. Hence, the phases 'over the top' will never be reached and can be deleted.

Additionally, one may show that aborting in N is always at least as good as idling, provided $\alpha r(N) + h \geq 0$. This is surely the case if $r(N)$ is non-negative, but if $r(N) < 0$, then the discount rate α reduces the loss on the job sitting in node N by a fraction α at the cost of h in one period. So, for this particular job, postponing the abort by one period is rewarding if $(1-\alpha)r(N) - h > r(N)$. Therefore, the condition $\alpha r(N) + h \geq 0$ implies that for the system as a whole, idling in N will never result in any gain. A formal proof can be given via sample path arguments, but we omit it here.

Remark 3.3 The concavity of the reward function implies that in case of an abort or a multi-abort, the job in service is always aborted first.

We will use this property immediately, but it is more convenient to postpone the proof until Section 3.3. See Corollary 3.1.

3.1.1 Dynamic Programming formulation

In this section, we summarize and complete the model in terms of a mathematical formulation. After that, we state and prove the main theorem, which provides a characterization of the optimal admission/termination control policy.

Recapitulating, let i denote the number of jobs in the system and k the node the job under service resides in. Let (i,k) be the state of the system for $i \geq 1$ and $0 \leq k \leq N$. If the system is empty, then the state is $(0,\cdot)$. Whenever $(0,k)$ appears in an expression, $(0,\cdot)$ can be substituted. We will use the following notation:

- $V_n(i,k)$ denotes the maximum expected n-period α-discounted reward when the current state, *just before* the next decision to continue or abort, is (i,k). Note that state (i,k) may be the result of a (real or artificial) service phase completion, but also of an arrival immediately after the accept/reject decision.
- $V_n(i,k;\pi)$ denotes the maximum expected n-period α-discounted reward when the current state, just before the next decision to continue or abort, is (i,k), and given that decision π is chosen in that state, where $\pi \in \{\texttt{continue}, \texttt{abort}\}$. Let π^* denote the optimal decision, so $V_n(i,k) = V_n(i,k;\pi^*)$. In the notation π^* the dependence on i, k and n is suppressed. Further, we use commas to separate state characteristics and a semi-colon to separate the decision from the state.

- $V_n(i,k,\text{arr})$ denotes the maximum expected n-period α-discounted reward when the current state is (i,k), given that at this very point in time an arrival event occurs.
- $V_n(i,k,\text{arr};\pi)$ denotes the maximum expected n-period α-discounted reward when the current state is (i,k), given that at this very point in time an arrival event occurs, and given that decision π is chosen in that state, where $\pi \in \{\texttt{accept},\texttt{reject}\}$. Again, let π^* denote the optimal decision, so $V_n(i,k,\text{arr}) = V_n(i,k,\text{arr};\pi^*)$.
- Finally, when time hits zero, all jobs in the queue yield their initial reward $r(0)$, and the job in service, residing in node k, yields its current cashable reward, which is $r(k)$.

Then our model is defined by the following set of DPEs. To save space, we will usually write `ab` for `abort` and `co` for `continue` in formal expressions (and also `ac` for `accept` and `rj` for `reject`).

$$V_0(0,\cdot) = 0$$

$$V_0(i,k) = r(k) + (i-1)r(0) \qquad i \geq 1,\ 0 \leq k \leq N$$

For $n \geq 0$:

$$V_n(0,\cdot,\text{arr}) = \mathbf{max}\{V_n(1,0) - c,\ V_n(0,\cdot) + r(0)\}$$

$$V_n(i,k,\text{arr}) = \mathbf{max}\{V_n(i+1,k) - c,\ V_n(i,k) + r(0)\} \qquad i \geq 1,\ 0 \leq k \leq N$$

For $n \geq 1$:

$$V_n(0,\cdot) = V_n(0,\cdot\ ;\texttt{co})$$

$$V_n(0,\cdot\ ;\texttt{co}) = \lambda V_{n-1}(0,\cdot,\text{arr}) + \mu V_{n-1}(0,\cdot)$$

$$V_n(i,k) = \mathbf{max}\{V_n(i,k;\texttt{co}),\ V_n(i,k;\texttt{ab})\} \qquad i \geq 1,\ 0 \leq k \leq N$$

$$V_n(i,k;\texttt{co}) = \lambda V_{n-1}(i,k,\text{arr}) + \mu V_{n-1}(i,k+1) - ih \qquad i \geq 1,\ 0 \leq k < N$$

$$V_n(i,N;\texttt{co}) = \lambda V_{n-1}(i,N,\text{arr}) + \mu V_{n-1}(i,N) - ih \qquad i \geq 1$$

$$V_n(i,k;\texttt{ab}) = r(k) + V_n(i-1,0) \qquad i \geq 1,\ 0 \leq k \leq N$$

3.2 Overview of the results

We will prove the following theorem.

Theorem 3.1
{Characterization of the optimal admission/termination policy}
For any remaining number of periods n, the optimal admission/termination policy can be characterized as follows:

1. *If it is optimal to reject an arriving job in state (i,k), then it is optimal as well to reject it in all states (j,l) with $j > i$, and in all states (i,l) with $l < k$.*
2. *If it is optimal to abort the service of a job in state (i,k), then it is optimal as well to abort service in all states (j,l) with $j \geq i$ and $l \geq k$.*
3. *If $c > 0$ and if it is optimal to accept an arriving job in state (i,k), then it is optimal as well to continue the service of a job in all states $(j,0)$ with $j \leq i$.*

The result is intuitive but the proof becomes rather involved. A graphical representation of the structure of typical admission and termination policies is given below. The solid dots represent states in which jobs are rejected or terminated, respectively, and the two solid polylines capture the respective rejection and termination regions.

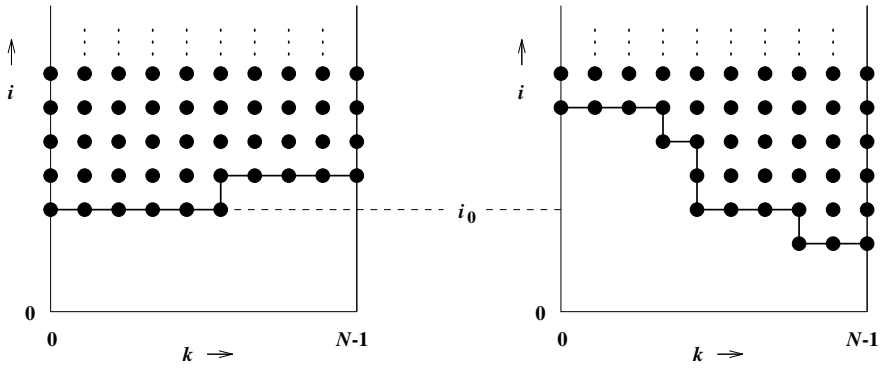

Figure 3.2: Characterization of the optimal admission (*left-hand side*) and termination (*right-hand side*) policies

We note that in the optimal admission policy either there is a unique (threshold) node (i_0, k_0) where the polyline moves up one level, or the polyline consists of a single horizontal line at some job level i_0. The occurrence of multiple threshold nodes or the polyline moving up more than one level would contradict Part 1 of Theorem 3.1.

If $c > 0$, then in the optimal termination policy the termination region does not cover nodes $(i, 0)$, $i \leq i_0$ or $i < i_0$, respectively. For all $c \geq 0$, the termination region is further determined by a non-increasing step-function (switch-over curve) such that for all i there exists a threshold node k of i and for all k there exists a threshold level i of k.

The following two examples show that indeed the polylines are as general as stated. The vector $\underline{r}$ denotes the reward function, i.e., $\underline{r} = (r(0), \ldots, r(N))$. Note that both examples comply with Fundamental Assumption 3.1.

Example 3.2 Consider the following instance of our model: $\lambda = \frac{1}{2}$, $\mu = \frac{1}{2}$, $\alpha = 0$, $h = \frac{1}{2}$, $c = 30$, $N = 3$ and $\underline{r} = (0, 25, 45, 60)$. The average reward optimal admission/termination policy for this system is given below. The hollow dots represent states in which we accept or continue, respectively. Note that the polyline capturing the rejection region is a straight line.

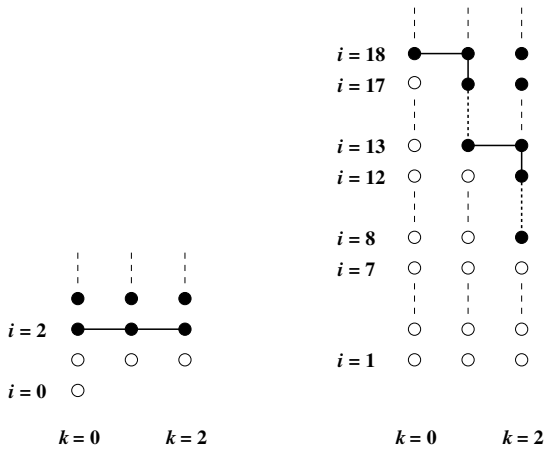

Figure 3.3: Optimal admission (*left-hand side*) and termination (*right-hand side*) policies for Example 3.2

Example 3.3 Consider the following instance of our model: $\lambda = \frac{1}{6}$, $\mu = \frac{5}{6}$, $\alpha = 0$, $h = \frac{5}{3}$, $c = 10$, $N = 4$ and $\underline{r} = (0, 25, 45, 60, 72)$. The average reward optimal admission/termination policy for this system is given below. Note that the polyline capturing the rejection region moves up one level at state $(4, 1)$.

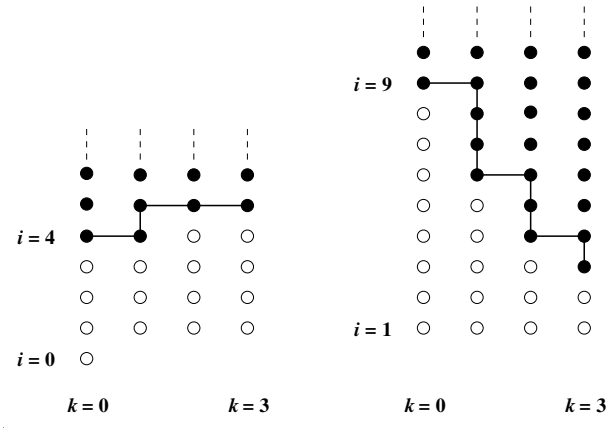

Figure 3.4: Optimal admission (*left-hand side*) and termination (*right-hand side*) policies for Example 3.3

Remark 3.4 Part 3 of Theorem 3.1 implies that if $c > 0$, then immediately after the admission of a new job, at most one job is aborted. So in this case there will only occur so-called *single-aborts* and no multi-aborts.

Further, if $c = 0$, then it is always optimal to accept an arriving job, since it may be aborted at the same moment in time at no additional cost.

3.2.1 The line of proof

In order to establish Theorem 3.1, we will prove the following monotonicity results, from which Theorem 3.1 will be obtained as a corollary.

Proposition 3.1 {Key Proposition}
Let $r(k)$ be non-decreasing and concave in k. Then, for $i \geq 0$,

$$V_n(i{+}1,l) - V_n(i,l) \geq V_n(i{+}2,k) - V_n(i{+}1,k), \tag{3.1}$$
$$V_n(i{+}1,k{+}1) - V_n(i,k{+}1) \geq V_n(i{+}1,k) - V_n(i,k), \tag{3.2}$$
$$\nearrow \tag{3.3}$$
$$V_n(i{+}1,l,\mathrm{arr}) - V_n(i,l,\mathrm{arr}) \geq V_n(i{+}2,k,\mathrm{arr}) - V_n(i{+}1,k,\mathrm{arr}),$$
$$V_n(i{+}1,k{+}1,\mathrm{arr}) - V_n(i,k{+}1,\mathrm{arr}) \geq V_n(i{+}1,k,\mathrm{arr}) - V_n(i,k,\mathrm{arr}),$$
$$\searrow \tag{3.4}$$
$$\nearrow \tag{3.5}$$
$$V_n(i{+}1,l;\mathtt{co}) - V_n(i,l;\mathtt{co}) \geq V_n(i{+}2,k;\mathtt{co}) - V_n(i{+}1,k;\mathtt{co}),$$
$$V_n(i{+}1,k{+}1;\mathtt{co}) - V_n(i,k{+}1;\mathtt{co}) \geq V_n(i{+}1,k;\mathtt{co}) - V_n(i,k;\mathtt{co}),$$
$$\searrow \tag{3.6}$$

where

- *(3.1) and (3.3) hold for all $n \geq 0$, $0 \leq k \leq N$ and $0 \leq l \leq N$,*
- *(3.2) and (3.4) hold for all $n \geq 0$ and $0 \leq k < N$,*
- *(3.5) holds for all $n \geq 1$, $0 \leq k \leq N$ and $0 \leq l \leq N$,*
- *(3.6) holds for all $n \geq 1$ and $0 \leq k < N$.*

In words, inequality (3.1) states that the value of an additional job in state $(i+1,k)$ does not surpass the value of an additional job in state (i,l) for any k and l. By taking $l = k$, one can immediately see that the inequality also states that our value function $V_n(i,k)$ is concave in i for all n and k, implying that the value of an extra job is non-increasing in i.

Analogously, inequality (3.2) states that the value of an additional job in state (i,k) does not surpass the value of an additional job in state $(i,k+1)$ for $k<N$, i.e., the value of an extra job is non-decreasing in k. More general, the inequality states that $V_n(i,k)$ is submodular; cf. (1.3) and (2.2).

3.3 Proof of the Key Proposition

The proof of the Key Proposition uses induction on the remaining number of periods and runs as follows.

Step 0: Verify the first two inequalities (3.1) and (3.2) for $n = 0$.

Step 1: Assuming (3.1) and (3.2) to hold for some $n \geq 0$, prove (3.3) and (3.4) for n.

Step 2: Using this result, prove that (3.5) and (3.6) hold for $n+1$.

Step 3: Finally, prove that (3.1) and (3.2) also hold for $n+1$.

In the proof we will make use of Lemma 1.1, and of a proposition which we will state and prove in Step 3 of the proof, where it is used. Note that $S = \{(i,k),\, (i,k,\mathrm{arr}) \,|\, i \geq 0,\, 0 \leq k \leq N\}$ in our M|E$_N$|1 model.

Proof of the Key Proposition.

Step 0. We verify that (3.1) and (3.2) hold for $n = 0$.

First, let $i = 0$. Then, for $0 \leq k \leq N$ and $0 \leq l \leq N$, using the monotonicity of $r(\cdot)$,

$$V_0(1,l) - V_0(0,\cdot) = r(l) \geq r(0) = V_0(2,k) - V_0(1,k)$$

and, for $0 \leq k < N$,

$$V_0(1,k+1) - V_0(0,\cdot) = r(k+1) \geq r(k) = V_0(1,k) - V_0(0,\cdot).$$

Next, let $i \geq 1$. Then, for $0 \leq k \leq N$ and $0 \leq l \leq N$,

$$V_0(i+1,l) - V_0(i,l) = r(0) = V_0(i+2,k) - V_0(i+1,k)$$

and, for $0 \leq k < N$,

$$V_0(i+1,k+1) - V_0(i,k+1) = r(0) = V_0(i+1,k) - V_0(i,k).$$

Thus, inequalities (3.1) and (3.2) hold for $n = 0$.

Induction hypothesis. Assume that for some $n \geq 0$, inequality (3.1) holds for all $i \geq 0$, $0 \leq k \leq N$ and $0 \leq l \leq N$, and (3.2) for all $i \geq 0$ and $0 \leq k < N$. This will be our induction hypothesis.

Step 1. Under the induction hypothesis, we show that (3.3) and (3.4) hold for n.

We will first prove (3.3). Let $i \geq 0$, $0 \leq k \leq N$ and $0 \leq l \leq N$.

The next decision, d_1 say, prescribed by the (optimal) policy corresponding to $V_n(i,l,\mathrm{arr})$, is either to accept or to reject the new job. Clearly, this also holds for the next decision, d_2 say, prescribed by the (optimal) policy

corresponding to $V_n(i+2,k,\mathrm{arr})$. There are at most four joint cases (d_1,d_2). We will show that inequality (3.3) holds for each case, irrespective of the question whether that case will actually occur; cf. Remark 3.5.

The four cases can be presented as follows, where $\mathcal{A}$ indicates that `accept` is optimal and $\mathcal{R}$ indicates that `accept` is not optimal:

$$\begin{array}{lcl}
\mathcal{AA} & : & V_n(i,l,\mathrm{arr};\mathtt{ac}) \geq V_n(i,l,\mathrm{arr};\mathtt{rj}) \quad \wedge \\
 & & \qquad V_n(i+2,k,\mathrm{arr};\mathtt{ac}) \geq V_n(i+2,k,\mathrm{arr};\mathtt{rj}), \\
\mathcal{AR} & : & V_n(i,l,\mathrm{arr};\mathtt{ac}) \geq V_n(i,l,\mathrm{arr};\mathtt{rj}) \quad \wedge \\
 & & \qquad V_n(i+2,k,\mathrm{arr};\mathtt{ac}) < V_n(i+2,k,\mathrm{arr};\mathtt{rj}), \\
\mathcal{RA} & : & V_n(i,l,\mathrm{arr};\mathtt{ac}) < V_n(i,l,\mathrm{arr};\mathtt{rj}) \quad \wedge \\
 & & \qquad V_n(i+2,k,\mathrm{arr};\mathtt{ac}) \geq V_n(i+2,k,\mathrm{arr};\mathtt{rj}), \\
\mathcal{RR} & : & V_n(i,l,\mathrm{arr};\mathtt{ac}) < V_n(i,l,\mathrm{arr};\mathtt{rj}) \quad \wedge \\
 & & \qquad V_n(i+2,k,\mathrm{arr};\mathtt{ac}) < V_n(i+2,k,\mathrm{arr};\mathtt{rj}).
\end{array}$$

Then,

- under $\mathcal{AA}$,

$$\begin{array}{rcl}
V_n(i+1,l,\mathrm{arr};\mathtt{ac}) - V_n(i,l,\mathrm{arr}) & = & V_n(i+2,l) - c - [V_n(i+1,l) - c] \\
 & = & V_n(i+2,l) - V_n(i+1,l) \\
 & \geq & \{\textsf{induction hypothesis; (3.1)}\} \\
 & & V_n(i+3,k) - V_n(i+2,k) \\
 & = & V_n(i+3,k) - c - [V_n(i+2,k) - c] \\
 & = & V_n(i+2,k,\mathrm{arr}) - V_n(i+1,k,\mathrm{arr};\mathtt{ac});
\end{array} \tag{3.7}$$

- under $\mathcal{AR}$,

$$\begin{array}{rcl}
V_n(i+1,l,\mathrm{arr};\mathtt{rj}) - V_n(i,l,\mathrm{arr}) & = & V_n(i+1,l) + r(0) - [V_n(i+1,l) - c] \\
 & = & r(0) + c \\
 & = & V_n(i+2,k) + r(0) - [V_n(i+2,k) - c] \\
 & = & V_n(i+2,k,\mathrm{arr}) - V_n(i+1,k,\mathrm{arr};\mathtt{ac});
\end{array} \tag{3.8}$$

- under $\mathcal{RA}$,

$$\begin{aligned} V_n(i+1,l,\mathrm{arr};\mathtt{rj}) - V_n(i,l,\mathrm{arr}) &= V_n(i+1,l) - V_n(i,l) \\ &\geq \{\text{induction hypothesis; (3.1) twice}\} \\ &\quad V_n(i+3,k) - V_n(i+2,k) \\ &= V_n(i+2,k,\mathrm{arr}) - V_n(i+1,k,\mathrm{arr};\mathtt{ac}); \end{aligned} \tag{3.9}$$

- under $\mathcal{RR}$,

$$\begin{aligned} V_n(i+1,l,\mathrm{arr};\mathtt{rj}) - V_n(i,l,\mathrm{arr}) &= V_n(i+1,l) - V_n(i,l) \\ &\geq \{\text{induction hypothesis; (3.1)}\} \\ &\quad V_n(i+2,k) - V_n(i+1,k) \\ &= V_n(i+2,k,\mathrm{arr}) - V_n(i+1,k,\mathrm{arr};\mathtt{rj}). \end{aligned} \tag{3.10}$$

Finally, apply Lemma 1.1 to each of the relations (3.7) through (3.10) to obtain (3.3). This ends our proof of (3.3).

We now shift our attention to (3.4). The proof resembles the one of (3.3). Let $i \geq 0$ and $0 \leq k < N$. Again, we distinguish four cases: $\mathcal{AA}$, $\mathcal{AR}$, $\mathcal{RA}$, and $\mathcal{RR}$. For example, $\mathcal{AR}$ means that

$$\begin{aligned} V_n(i,k+1,\mathrm{arr};\mathtt{ac}) \;\geq\; V_n(i,k+1,\mathrm{arr};\mathtt{rj}) \quad \wedge \\ V_n(i+1,k,\mathrm{arr};\mathtt{ac}) \;<\; V_n(i+1,k,\mathrm{arr};\mathtt{rj}). \end{aligned}$$

Then,

- under $\mathcal{AA}$,

$$\begin{aligned} V_n(i+1,k+1,\mathrm{arr};\mathtt{ac}) - V_n(i,k+1,\mathrm{arr}) & \\ &= V_n(i+2,k+1) - V_n(i+1,k+1) \\ &\geq \{\text{induction hypothesis; (3.2)}\} \\ &\quad V_n(i+2,k) - V_n(i+1,k) \\ &= V_n(i+1,k,\mathrm{arr}) - V_n(i,k,\mathrm{arr};\mathtt{ac}); \end{aligned} \tag{3.11}$$

- under $\mathcal{AR}$,

$$\begin{aligned} V_n(i+1,k+1,\mathrm{arr};\mathtt{rj}) - V_n(i,k+1,\mathrm{arr}) & \\ &= V_n(i+1,k+1) + r(0) - [V_n(i+1,k+1) - c] \\ &= V_n(i+1,k) + r(0) - [V_n(i+1,k) - c] \\ &= V_n(i+1,k,\mathrm{arr}) - V_n(i,k,\mathrm{arr};\mathtt{ac}); \end{aligned} \tag{3.12}$$

- under $\mathcal{RA}$,

$$\begin{aligned} V_n(i+1,k+1,\mathrm{arr};\mathtt{rj}) - V_n(i,k+1,\mathrm{arr}) & \\ = \;& V_n(i+1,k+1) - V_n(i,k+1) \\ \geq \;& \{\textsf{induction hypothesis;}\;\; (3.1)\} \\ & V_n(i+2,k) - V_n(i+1,k) \\ = \;& V_n(i+1,k,\mathrm{arr}) - V_n(i,k,\mathrm{arr};\mathtt{ac}); \end{aligned} \tag{3.13}$$

- under $\mathcal{RR}$,

$$\begin{aligned} V_n(i+1,k+1,\mathrm{arr};\mathtt{rj}) - V_n(i,k+1,\mathrm{arr}) & \\ = \;& V_n(i+1,k+1) - V_n(i,k+1) \\ \geq \;& \{\textsf{induction hypothesis;}\;\; (3.2)\} \\ & V_n(i+1,k) - V_n(i,k) \\ = \;& V_n(i+1,k,\mathrm{arr}) - V_n(i,k,\mathrm{arr};\mathtt{rj}). \end{aligned} \tag{3.14}$$

Finally, apply Lemma 1.1 to each of the relations (3.11) through (3.14) to obtain (3.4). This ends our proof of (3.4).

Remark 3.5 Case $\mathcal{RA}$ in the proof of (3.3) constitutes a violation of the Key Proposition, because under $\mathcal{RA}$,

$$\begin{aligned} r(0)+c \;\; > \;\; & \{\mathcal{RA}\ \textsf{assumed}\} \\ & V_n(i+1,l) - V_n(i,l) \\ \geq \;\; & \{\textsf{induction hypothesis;}\;\; (3.1)\ \textsf{twice}\} \\ & V_n(i+3,k) - V_n(i+2,k) \\ \geq \;\; & \{\mathcal{RA}\ \textsf{assumed}\} \\ & r(0)+c, \end{aligned}$$

which leads to a contradiction. Given the correctness of the Key Proposition, this proves that case $\mathcal{RA}$ in the proof of (3.3) is an *impossible* joint case.

Step 2. Assuming (3.1) through (3.4), we show that (3.5) and (3.6) hold for $n+1$. For this purpose, we define $m^- := \mathbf{min}\{m, N\}$ for all integers $m \geq 0$.

Let $i \geq 0$. Then, for $0 \leq k \leq N$ and $0 \leq l \leq N$,

$$\begin{aligned}
&V_{n+1}(i+1,l;\mathsf{co}) - V_{n+1}(i,l;\mathsf{co}) \\
&\qquad = \lambda[V_n(i+1,l,\mathrm{arr}) - V_n(i,l,\mathrm{arr})] \; + \\
&\qquad\quad \mu[V_n(i+1,(l+1)^-) - V_n(i,(l+1)^-)] \; - \; h \\
&\qquad \geq \{\text{induction hypothesis; (3.3); (3.1)}\} \\
&\qquad\quad \lambda[V_n(i+2,k,\mathrm{arr}) - V_n(i+1,k,\mathrm{arr})] \; + \\
&\qquad\quad \mu[V_n(i+2,(k+1)^-) - V_n(i+1,(k+1)^-)] \; - \; h \\
&\qquad = V_{n+1}(i+2,k;\mathsf{co}) - V_{n+1}(i+1,k;\mathsf{co})
\end{aligned}$$

and, for $0 \leq k < N$,

$$\begin{aligned}
&V_{n+1}(i+1,k+1;\mathsf{co}) - V_{n+1}(i,k+1;\mathsf{co}) \\
&\qquad = \lambda[V_n(i+1,k+1,\mathrm{arr}) - V_n(i,k+1,\mathrm{arr})] \; + \\
&\qquad\quad \mu[V_n(i+1,(k+2)^-) - V_n(i,(k+2)^-)] \; - \; h \\
&\qquad \geq \{\text{induction hypothesis; (3.4); (3.2)}\} \\
&\qquad = \lambda[V_n(i+1,k,\mathrm{arr}) - V_n(i,k,\mathrm{arr})] \; + \\
&\qquad\quad \mu[V_n(i+1,k+1) - V_n(i,k+1)] \; - \; h \\
&\qquad = V_{n+1}(i+1,k;\mathsf{co}) - V_{n+1}(i,k;\mathsf{co}).
\end{aligned}$$

Step 3. In order to perform the third step of our proof, we will need the following proposition, which exploits the concavity of $r(\cdot)$.

Proposition 3.2 *For all $n \geq 0$, $i \geq 0$ and $0 \leq k < N$,*

$$\left.\begin{array}{l} V_n(i,k+1) - V_n(i,k) \\ V_{n+1}(i,k+1;\mathsf{co}) - V_{n+1}(i,k;\mathsf{co}) \\ V_n(i,k+1,\mathrm{arr}) - V_n(i,k,\mathrm{arr}) \end{array}\right\} \leq r(k+1) - r(k). \qquad (3.15)$$

Proof. We could give an inductive proof that runs along the lines of the proof of the Key Proposition, but we choose not to reproduce this proof here.[4] Instead, we will prove the proposition by means of coupling and sample path arguments. We only consider the first inequality; the other two can be treated in exactly the same way.

Consider two n-period process instances of our model, instance $\mathcal{I}_{k+1}$ starting in $(i,k+1)$ and instance $\mathcal{I}_k$ starting in (i,k). We couple all events (all arrival

[4] An inductive proof will be given in the course of Chapter 4.

and service phase completion events and, if applicable, the event that the process vanishes) and all decisions (`accept` versus `reject` and `continue` versus `abort`). To be precise, instance $\mathcal{I}_{k+1}$ follows the optimal policy and instance $\mathcal{I}_k$ copies all the actions taken in instance $\mathcal{I}_{k+1}$. This is feasible, because $\mathcal{I}_{k+1}$ and $\mathcal{I}_k$ feature the same number of jobs in the system and the same remaining number of periods.

Then both instances are always completely identical, except for the job in service at the beginning, i.e., with n periods to go. This job will always be one phase behind in instance $\mathcal{I}_k$ until the job is aborted. (There is one exception: if in instance $\mathcal{I}_{k+1}$ the job has reached node N and the job is not aborted and there is an artificial service phase completion, then $\mathcal{I}_k$ becomes identical to $\mathcal{I}_{k+1}$.) If the job is never aborted because the system vanishes, then the difference in reward between the two instances is zero. If, after m service phase completions, the job is aborted (possibly at time $n = 0$), then the difference in reward is $r((k+m+1)^-) - r((k+m)^-)$, which, because $r(\cdot)$ is non-decreasing and concave, is at most $r(k+1) - r(k)$.

□

We can now prove the property mentioned in Remark 3.3, which we have already been using.

Corollary 3.1 *If a job is aborted, then aborting the job in service is at least as good as aborting a job from the queue.*

Proof. Consider state $(i+1, k)$ with n periods to go. Compare aborting the job in service, which gives a maximal reward of $r(k) + V_n(i, 0)$, to aborting a job from the queue, which results in a maximal reward of $r(0) + V_n(i, k)$. By repeated application of (3.15) we have $r(k) + V_n(i, 0) - r(0) - V_n(i, k) \geq 0$, so aborting the job in service is at least as good as aborting a job from the queue.

□

Now, assuming (3.1) through (3.6), we show that (3.2) and (3.1) hold for $n + 1$ as well. The proofs are similar to the ones of (3.3) and (3.4).

We first consider (3.2) and prove that this inequality holds for $n + 1$. Let $i \geq 0$ and $0 \leq k < N$.

The next decision, d_1 say, prescribed by the (optimal) policy corresponding to $V_{n+1}(i, k+1)$, is either to continue or to abort the job under service.

Clearly, this also holds for the next decision, d_2 say, prescribed by the (optimal) policy corresponding to $V_{n+1}(i+1,k)$. There are at most four joint cases (d_1,d_2). We will show that inequality (3.2) holds for each case.

The four cases can be presented as follows, where $\mathcal{C}$ indicates that `continue` is optimal and $\mathcal{A}$ indicates that `continue` is not optimal:

$$\begin{array}{lcl}
\mathcal{CC} & : & V_{n+1}(i,k+1;\mathtt{co}) \geq V_{n+1}(i,k+1;\mathtt{ab}) \quad \wedge \\
 & & \qquad V_{n+1}(i+1,k;\mathtt{co}) \geq V_{n+1}(i+1,k;\mathtt{ab}), \\
\mathcal{CA} & : & V_{n+1}(i,k+1;\mathtt{co}) \geq V_{n+1}(i,k+1;\mathtt{ab}) \quad \wedge \\
 & & \qquad V_{n+1}(i+1,k;\mathtt{co}) < V_{n+1}(i+1,k;\mathtt{ab}), \\
\mathcal{AC} & : & V_{n+1}(i,k+1;\mathtt{co}) < V_{n+1}(i,k+1;\mathtt{ab}) \quad \wedge \\
 & & \qquad V_{n+1}(i+1,k;\mathtt{co}) \geq V_{n+1}(i+1,k;\mathtt{ab}), \\
\mathcal{AA} & : & V_{n+1}(i,k+1;\mathtt{co}) < V_{n+1}(i,k+1;\mathtt{ab}) \quad \wedge \\
 & & \qquad V_{n+1}(i+1,k;\mathtt{co}) < V_{n+1}(i+1,k;\mathtt{ab}).
\end{array}$$

The cases $\mathcal{AC}$ and $\mathcal{AA}$ vanish for $i=0$, because `abort` is not an option in state $(0,\cdot)$.

Then,

- case $\mathcal{CC}$ follows, with `continue` chosen in $(i+1,k+1)$ and (i,k) as well, using (3.6) and Lemma 1.1, immediately from the induction hypothesis;

- with respect to case $\mathcal{CA}$, we need some extra notation: if the optimal decision is to first abort j jobs and then to continue, then we denote this by $\mathcal{A}^j$ and we write $\mathtt{ab}^j$ for the not necessarily optimal copied decision; then, for $\mathcal{CA}^{j+1}$,

$$\begin{aligned}
&V_{n+1}(i+1,k+1;\mathtt{ab}^{j+1}) - V_{n+1}(i,k+1) \\
&\quad = r(k+1) + jr(0) + V_{n+1}(i-j,0;\mathtt{co}) - V_{n+1}(i,k+1;\mathtt{co}) \\
&\quad \geq \{(3.15)\} \\
&\qquad r(k) + jr(0) + V_{n+1}(i-j,0;\mathtt{co}) - V_{n+1}(i,k;\mathtt{co}) \\
&\quad = V_{n+1}(i+1,k;\mathtt{ab}^{j+1}) - V_{n+1}(i,k;\mathtt{co}) \\
&\quad = V_{n+1}(i+1,k) - V_{n+1}(i,k;\mathtt{co}); \qquad (3.16)
\end{aligned}$$

- under $\mathcal{A}^j\mathcal{C}$,

$$\begin{aligned}
&V_{n+1}(i+1,k+1;\mathtt{ab}^{j}) - V_{n+1}(i,k+1) \\
&\quad = V_{n+1}(i+1-j,0;\mathtt{co}) - V_{n+1}(i-j,0;\mathtt{co}) \\
&\quad \geq \{\text{induction hypothesis; } (3.5)\} \\
&\qquad V_{n+1}(i+1,k;\mathtt{co}) - V_{n+1}(i,k;\mathtt{co}) \\
&\quad = V_{n+1}(i+1,k) - V_{n+1}(i,k;\mathtt{co}); \qquad (3.17)
\end{aligned}$$

- under $\mathcal{AA}$,

$$\begin{aligned} & V_{n+1}(i+1,k+1;\mathtt{ab}) - V_{n+1}(i,k+1) \\ & \quad = \quad V_{n+1}(i,0) - V_{n+1}(i-1,0) \\ & \quad = \quad V_{n+1}(i+1,k) - V_{n+1}(i,k;\mathtt{ab}). \end{aligned} \tag{3.18}$$

Finally, apply Lemma 1.1 to each of the relations (3.16), (3.17) and (3.18) to obtain (3.2) for $n+1$. This ends our proof of (3.2) for $n+1$.

Next we prove that (3.1) holds for $n+1$. The proof resembles the one of (3.2) for $n+1$. Let $i \geq 0$, $0 \leq k \leq N$ and $0 \leq l \leq N$. Again, we distinguish four cases: $\mathcal{CC}$, $\mathcal{CA}$, $\mathcal{AC}$, and $\mathcal{AA}$. For example, $\mathcal{AC}$ means that

$$V_{n+1}(i,l;\mathtt{co}) < V_{n+1}(i,l;\mathtt{ab}) \quad \wedge \quad V_{n+1}(i+2,k;\mathtt{co}) \geq V_{n+1}(i+2,k;\mathtt{ab}).$$

Again, the cases $\mathcal{AC}$ and $\mathcal{AA}$ vanish for $i=0$ by definition.

Then,

- case $\mathcal{CC}$ follows, with `continue` chosen in $(i+1,l)$ and $(i+1,k)$ as well, using (3.5) and Lemma 1.1, immediately from the induction hypothesis;

- under $\mathcal{CA}$ or $\mathcal{AA}$,

$$\begin{aligned} & V_{n+1}(i+1,l) - V_{n+1}(i,l) \\ & \qquad \geq \quad \{\text{induction hypothesis; } (3.2) \text{ for } n+1\} \\ & \qquad\quad V_{n+1}(i+1,0) - V_{n+1}(i,0) \\ & \qquad = \quad V_{n+1}(i+2,k;\mathtt{ab}) - V_{n+1}(i+1,k;\mathtt{ab}) \\ & \qquad = \quad V_{n+1}(i+2,k) - V_{n+1}(i+1,k;\mathtt{ab}); \end{aligned} \tag{3.19}$$

- under $\mathcal{A}^j\mathcal{C}$,

$$\begin{aligned} & V_{n+1}(i+1,l;\mathtt{ab}^j) - V_{n+1}(i,l) \\ & \qquad = \quad V_{n+1}(i+1,l;\mathtt{ab}^j) - V_{n+1}(i,l;\mathtt{ab}^j) \\ & \qquad = \quad V_{n+1}(i+1-j,0;\mathtt{co}) - V_{n+1}(i-j,0;\mathtt{co}) \\ & \qquad \geq \quad \{\text{induction hypothesis; } (3.5)\} \\ & \qquad\quad V_{n+1}(i+2,k;\mathtt{co}) - V_{n+1}(i+1,k;\mathtt{co}) \\ & \qquad = \quad V_{n+1}(i+2,k) - V_{n+1}(i+1,k;\mathtt{co}). \end{aligned} \tag{3.20}$$

Finally, apply Lemma 1.1 to (3.19) and (3.20) to obtain (3.1) for $n+1$. This ends our proof of (3.1) for $n+1$.

This concludes our proof of the Key Proposition.

□

Remark 3.6 Concavity of $r(\cdot)$ is only required at one particular point in the entire proof of the Key Proposition, namely, in the derivation of (3.16), where (3.15) is used.

We now derive Theorem 3.1 from the Key Proposition by means of a number of corollaries.

Corollary 3.2 *Let $n \geq 0$, $i \geq 0$ and $0 \leq k \leq N$. If it is optimal to reject an arriving job in state (i,k), then it is optimal to reject it in all states (j,l) with $j > i$ and $0 \leq l \leq N$.*

Proof. Let $n \geq 0$, $i \geq 0$ and $0 \leq k \leq N$. It suffices to show that

$$V_n(i,k) + r(0) \geq V_n(i+1,k) - c$$

implies

$$V_n(i+1,l) + r(0) \geq V_n(i+2,l) - c$$

for all $0 \leq l \leq N$. But this is immediate from (3.1).

□

Corollary 3.3 *Let $n \geq 0$, $i \geq 0$ and $1 \leq k \leq N$. If it is optimal to reject an arriving job in state (i,k), then it is optimal to reject it in all states (i,l) with $0 \leq l < k$.*

Proof. Corollary 3.3 holds by definition for $i = 0$. Let $n \geq 0$, $i \geq 1$ and $1 \leq k \leq N$. It suffices to show that

$$V_n(i,k) + r(0) \geq V_n(i+1,k) - c$$

implies

$$V_n(i,k-1) + r(0) \geq V_n(i+1,k-1) - c.$$

This is immediate from (3.2).

□

The combination of Corollaries 3.2 and 3.3 is exactly Part 1 of Theorem 3.1.

Corollary 3.4 *Let $n \geq 1$, $i \geq 1$ and $0 \leq k \leq N$. If it is optimal to abort the service of a job in state (i,k), then it is optimal to abort it in all states (j,l) with $j \geq i$ and $k \leq l \leq N$.*

Proof. Let $n \geq 1$, $i \geq 1$ and $0 \leq k \leq N$. The proof consists of two parts. Part 1: The combination of `continue` being optimal in $(i, k+1)$, i.e.,

$$V_n(i, k+1; \texttt{co}) \geq r(k+1) + V_n(i-1, 0),$$

and `continue` not being optimal in (i, k), i.e.,

$$V_n(i, k; \texttt{co}) < r(k) + V_n(i-1, 0),$$

violates Proposition 3.2.

Part 2: The combination of `abort` being optimal in (i, k), i.e.,

$$V_n(i, k) = r(k) + V_n(i-1, 0),$$

and `abort` not being optimal in $(i+1, k)$, i.e.,

$$V_n(i+1, k) > r(k) + V_n(i, 0),$$

violates (3.1).

□

Corollary 3.4 is exactly Part 2 of Theorem 3.1.

Corollary 3.5 *Let $n \geq 1$, $i \geq 1$ and $0 \leq k \leq N$. If $c > 0$ and if it is optimal to accept an arriving job in state (i, k), then it is optimal to continue service in state $(i, 0)$.*

Proof. If it is optimal to accept an arriving job in state (i, k) then we have

$$V_n(i+1, k) - c \geq V_n(i, k) + r(0).$$

But then by (3.1) also

$$V_n(i, 0) - V_n(i-1, 0) \geq V_n(i+1, k) - V_n(i, k) \geq r(0) + c > r(0),$$

which implies that in state $(i, 0)$, decision `abort` is not optimal and hence `continue` is optimal.

□

Together, Corollaries 3.4 and 3.5 constitute Part 3 of Theorem 3.1.

Summarizing, Corollaries 3.2 through 3.5 constitute Theorem 3.1.

3.3.1 Extension to convex holding costs

In our model we considered linear holding costs of $ih \geq 0$ per unit of time when there are i jobs in the system. It is straightforward to verify that all results and proofs remain valid in the more general case of non-decreasing, convex holding costs. To see this, let the holding costs be $h(i) \geq 0$ per unit of time when there are i jobs in the system, where $h(0) = 0$, and assume that $h(i)$ is convex. Note that this implies that $h(i)$ is also non-decreasing. Further, note that it suffices to consider Step 2 of the proof of the Key Proposition, since this is the only place in any of the proofs and steps of the proof of the Key Proposition where holding costs appear explicitly.

Step 2 of the proof of the Key Proposition consists of two parts. Let us first consider the first part. In the derivation concerned, the contribution of holding costs appearing on the left-hand side of the inequality now becomes $-[h(i+1) - h(i)]$, whereas the contribution on the right-hand side of the inequality becomes $-[h(i+2) - h(i+1)]$. Since $h(\cdot)$ is convex, we have $h(i+2)-h(i+1) \geq h(i+1)-h(i)$, from which the desired result is immediate.

The second part is even more easily verified. Namely, in the derivation concerned, the holding costs appearing on the left-hand side of the inequality are identical to the holding costs appearing on the right-hand side of the inequality, both contributions being equal to $-[h(i+1) - h(i)]$.

3.4 Infinite time horizon

So far we only considered a finite time horizon, i.e., a finite number of periods. For the most natural case $h > 0$, the extension of the threshold structure of the optimal strategy for the infinite horizon model is fairly standard.

First, we can argue that $h > 0$ implies that the system can be reduced to a finite state system, because jobs will not be accepted if the number of jobs in the system is large. To see this, consider a job that arrives when there are already m jobs in the system. Suppose we accept it and that it will (eventually) go into service. Recall that we treat the discount rate as the probability that the system vanishes, so if the job goes into service, the system has not yet vanished. Without loss of generality we can say that we discard jobs from the queue in the order 'latest arrivals discarded first'. Then, if our job goes into service, all jobs in front of it must have gone

into service as well. Hence it takes at least m periods before our job goes into service, and hence its total holding costs are at least mh. Its reward is at most $r(N)$. This implies that its net contribution (on top of the initial reward) is surely negative if m is larger than $(r(N) - r(0))/h$. From this we conclude that the total number of jobs in the system will not be larger than $(r(N)-r(0))/h$ and that the system is essentially a finite state system. Then the (standard) argument is as follows. For this finite state system there are only finitely many stationary strategies. For each number of periods n we get an optimal policy, f_n say, with a threshold structure. Then there must be a subsequence $\{f_{n_l}\}$ and a policy f^* with $f_{n_l} = f^*$ for all l. In the discounted case this policy is optimal, because $V_n(\cdot)$ converges (so f^* satisfies the optimality equation). In the average reward case, i.e., $\alpha = 0$, we can use the fact that for all policies the resulting Markov chain has only one recurrent class (state $(0, \cdot)$ will always be reached) and is aperiodic. Thus $V_n(\cdot) - ng^*$, with g^* the optimal average reward, converges for all initial states (such that $g^* = \limsup_{n\to\infty} n^{-1}V_n(0, \cdot)$). Thus f^* (which we know has the threshold structure) will be average reward optimal. See, e.g., Denardo [18].

More difficult are the many variants with $h = 0$. To demonstrate the complexity we will only discuss the special case $h = 0$, $c > 0$ and $\alpha = 0$. Consider the following example with $N = 1$. Let $-c + r(1) > r(0)$, so it is interesting to accept and serve jobs. Further, let $\lambda/\mu > 1$. I.e., there is not enough capacity to accept and serve all jobs. Then any stationary strategy that (on the average) accepts more than μ or less than μ jobs per time unit cannot be optimal. If it accepts less than μ, then the server idles whereas it could earn $-c + r(1)/\mu$ per time unit. If it accepts more than μ then it loses an amount of c for each job that cannot be served. Further, for any stationary strategy that accepts exactly μ, in order to be optimal, the probability that the server is idle must be 0. This would be possible only if it accepts in each state $(i, 0)$, because if the system rejects in state $(i, 0)$, all states $(j, 0)$ with $j \leq i$ would be positive recurrent and state $(0, \cdot)$ would have positive probability. But this leads to a contradiction. The strategy accepts exactly μ, but also accepts all jobs, so λ, whereas $\lambda > \mu$. Hence no optimal stationary strategy exists. We note, but do not prove, that optimal history dependent strategies do exist and can be constructed based on the idea that after each return to the idling state $(0, \cdot)$, it will take longer before the system will return to $(0, \cdot)$ again. E.g., in the first 'cycle' accept all jobs as long as $i < 10$, in the nth cycle accept all jobs as long as $i < 10^n$.

3.5 Counterexamples

It may be clear that Theorem 3.1 need not hold if the reward function $r(k)$ is *not* non-decreasing in k, i.e., strictly decreasing for at least one k. If $r(k)$ is indeed non-decreasing in k, but not concave, then, in general, Theorem 3.1 does not hold either. By means of the following two counterexamples, we subsequently show that Part 2 and both parts of Part 1 need no longer hold.

Counterexample 3.1 Consider the following instance of our model: $\mu=1$ (so $\lambda=0$ and $\alpha=0$), $h=0$, $N=7$ and $\underline{r}=(0,4,4,10,10,15,15,19)$. So there are no future arrivals and there are no holding costs for the jobs in the system. Let the remaining number of periods be 3. We are interested in the optimal policy in case the state is $(i,4), i \geq 1$, i.e., there is at least one job in the system and the job under service resides in node 4.

Noting that since $\alpha=0$ and $\lambda=0$, service phase completion events are the only events that can occur, it is straightforward to check that for $i=1,2,3$, the (unique) optimal policies and corresponding (expected) rewards $V_3(i,4)$ are as follows.

i	first decision (at $n=3$)	next decision (at $n=2$)	final decision (at $n=1$)	$V_3(i,4)$
1	`continue`	`continue`	`continue`	19
2	`abort`	`continue`	`continue`	20
3	`continue`	`abort`	`abort`	23

From the second column of the table (displaying the decisions made at $n=3$) it can be concluded that Part 2 of Theorem 3.1 does not hold.

Counterexample 3.2 Consider the same data as in Counterexample 3.1. Let $c=2$. Then $V_3(1,4,\text{arr})=19$, with unique optimal first decision `reject`. However, $V_3(2,4,\text{arr})=21$, with unique optimal first decision `accept`, violating the first part of Part 1 of Theorem 3.1, and $V_3(1,3,\text{arr})=18$, also with unique optimal first decision `accept`, violating the second part of Part 1 of Theorem 3.1. Therefore, both parts of Part 1 of Theorem 3.1 do not hold.

3.6 Convex rewards

The basic model we considered in this chapter was based on diminishing marginal returns, i.e., a non-decreasing but concave reward function. Although diminishing marginal returns are a common assumption (see Section 1.2.4), one could also imagine situations where the rewards typically exhibit a *convex* instead of concave structure. Examples are processes with a strong 'nothing ventured, nothing gained' nature, the most extreme exponent being the situation that all intermediate rewards are zero, i.e., without loss of generality, $\underline{r} = (0, 0, \ldots, 0, 1)$. In this case, for each job separately, either all N service phases are processed and the corresponding end reward of 1 is fetched, or the reward is 0, irrespective of how many phases have been completed.

It is easy to show that if the reward function is convex instead of concave, then Proposition 3.2 holds with a '$\geq$' instead of a '$\leq$' sign. It follows (cf. Corollary 3.1) that jobs in the queue are always more eligible for termination than the job in service, unless the latter has just completed its final phase. So, under the weak assumption that $\alpha r(N) + h \geq 0$, a job in service will be aborted only once it has received full service. In the infinite horizon case, this implies that the concept of termination control disappears entirely. The control problem will be a matter of admission control only. In our opinion, from an analytical point of view, this renders the 'mirror' model with convex rewards far less interesting than the original model with concave rewards. (Note that in the finite horizon case, termination control in the 'mirror' model will be limited to aborting jobs from the queue when the service of the job in service progresses slower than expected.) Therefore, we choose to skip a formal analysis of this model.

3.7 Conclusions

We have considered a single-server queueing model with Poisson arrivals and Erlang service times. For this M|E$_N$|1 queue we have dealt with two decision features. First, one has to decide upon arrival of a new job whether to accept or reject it. Second, one can decide at any moment to terminate the service of a job. The reward for a job for which k phases have been completed is $r(k)$, and we assumed that this reward function is non-decreasing and concave in the number of phases completed. We have shown that the optimal strategy for both types of decisions is characterized by threshold policies, namely,

reject if the system is considered to be full and abort if there is enough work waiting and if the job in service has already passed a sufficient number of service phases. By means of two counterexamples, we have illustrated the (general) need of the regularity conditions imposed on the reward function. In addition, we have addressed the consequences of convex instead of concave rewards.

The basic $M|E_N|1$ termination control model discussed in this chapter gives rise to many extensions to be investigated. Here, we distinguish the following two types: (1) generalization of the arrival process, and (2) generalization of the service process. The next chapter focuses exclusively on some specific extensions of the $M|E_N|1$ model. In particular, two natural extensions of the first type and one important extension of the second type will be reviewed, and generalized monotonicity and threshold results will be obtained for each of these models.

One highly relevant exponent of the second type is the *multi-server* version of our model. This model, which is analytically as well as computationally intractable, will be treated in Chapter 5. We will review two heuristics for the computation of the optimal policy for this multi-server model. One is based on an extension of the $M|E_N|1$ model which will be discussed in Chapter 4. The other is based on a modified version of the $M|E_N|1$ model in which the speed of the server depends on the number of jobs in the system.

4

Extensions of the M|E$_N$|1 model

In this chapter, we address several extensions of the basic M|E$_N$|1 model described and analysed in the previous chapter. In particular, we will consider two generalizations concerning the arrival process and one important generalization concerning the service process. For each specific model, generalized monotonicity and threshold results will be obtained by means of (mostly small-scale) adjustments to the mathematical formulation and analysis of the original M|E$_N$|1 model.

In Section 4.1, we show that the results obtained in Chapter 3 extend to the more general case of *batch* Poisson arrivals. In Section 4.2, we obtain a similar result for *Erlang* arrivals. This result can be further extended to general *phase-type* arrivals. Subsequently, in Section 4.3, we consider the most drastic and important extension to be studied in this chapter. It concerns the incorporation of a far more flexible Markov *feed-forward* routing mechanism in the M|E$_N$|1 model. The generalization of the results obtained for the original model will be established under two limiting regularity conditions, which replace Fundamental Assumption 3.1. The necessity of these regularity conditions will be illustrated by means of some counterexamples. Furthermore, as a spin-off, we capture the structure of the optimal policy for a discrete-time model with *deterministic* decision epochs and service times consisting of a sum of geometric phases. This is the subject of Section 4.4. In this context, we also describe and analyse a special model in which the reward for a job is either 0 or 1.

Remark 4.1 Wherever we use the term 'original model' in this chapter, we refer to the basic M|E$_N$|1 model studied in Chapter 3.

4.1 Extension to batch Poisson arrivals

In the original model, we assumed a standard Poisson arrival process, i.e., jobs arrive one at a time, with mean (exponential) interarrival time $1/\lambda$. However, in practice, besides independent arrivals, it is often a common phenomenon that service requests frequently arrive in batches. For example, when some large-scale calamity has happened, insurance companies can expect to receive a large number of simultaneous claims for damages.

In this section, we show that the results stated and proved in Sections 3.2 and 3.3 extend to the following $\mathrm{M}^X|\mathrm{E}_N|1$ version of the original model. Consider the original model, but let jobs arrive in batches. Let B ($\subset \mathbb{N}\backslash\{0\}$) denote the set of all possible batch sizes. Batches of size b (i.e., batches consisting of b separate jobs), where $b \in B$, arrive according to a Poisson process with rate λ_b. Upon arrival of a batch, of size j say, we may decide for each job separately whether we accept it or not. So, we may accept any number of jobs between 0 and j, and reject the remainder of the batch.

We assume that each job admitted to the system incurs its own consideration costs c, and that for each job not admitted to the system we immediately receive the initial reward $r(0)$. As a result of this assumption, jobs can effectively be accepted or rejected on a one by one basis. Since jobs are mutually indistinguishable upon arrival, the order in which the separate jobs of a batch are considered is irrelevant. Note that we can obtain the original model from the $\mathrm{M}^X|\mathrm{E}_N|1$ model by choosing $B = \{1\}$.

4.1.1 Dynamic Programming formulation

We can basically use the same state description as in the original model, i.e., (i, k) denotes the state of the system. Further, we take $\sum_{b\in B} \lambda_b + \mu + \alpha = 1$ without loss of generality. This is our uniformization. (Then, for every $b \in B$, λ_b is the probability that the next transition is an arrival of a batch of size b.) Next, for $b \geq 1$, we introduce the following notation at arrival times: $V_n(i, k, \mathrm{arr}, b)$ denotes the maximum expected n-period reward when the current state is (i, k), given that at this moment a batch of size b arrives at the system. In this notation, b is not necessarily an element of B.

We can then write down the following new DPEs at arrival times (where $n \geq 0$, $b > 1$, $i \geq 1$ and $0 \leq k \leq N$):

$$
\begin{aligned}
V_n(0,\cdot,\mathrm{arr},1) &= \mathbf{max}\{V_n(1,0)-c,\ V_n(0,\cdot)+r(0)\}\\
V_n(i,k,\mathrm{arr},1) &= \mathbf{max}\{V_n(i{+}1,k)-c,\ V_n(i,k)+r(0)\}\\
V_n(0,\cdot,\mathrm{arr},b) &= \mathbf{max}\{V_n(1,0,\mathrm{arr},b{-}1)-c,\ V_n(0,\cdot,\mathrm{arr},b{-}1)+r(0)\}\\
V_n(i,k,\mathrm{arr},b) &= \mathbf{max}\{V_n(i{+}1,k,\mathrm{arr},b{-}1)-c,\ V_n(i,k,\mathrm{arr},b{-}1)+r(0)\}
\end{aligned}
$$

Note that we have incorporated a recursion in the DPEs for $V_n(i,k,\mathrm{arr},b)$, similar to the recursion incorporated in the DPEs for $V_n(i,k)$.

Finally, we have to adjust the DPEs for $V_n(i,k;\mathtt{co})$, viz., all $\lambda V_{n-1}(i,k,\mathrm{arr})$ terms are to be replaced by $\sum_{b\in B}\lambda_b V_{n-1}(i,k,\mathrm{arr},b)$. For example, for $n\geq 1$, we obtain the following DPE in case the system is empty:

$$
V_n(0,\cdot\,;\mathtt{co}) = \sum\nolimits_{b\in B}\lambda_b\, V_{n-1}(0,\cdot,\mathrm{arr},b) + \mu V_{n-1}(0,\cdot)
$$

This completes our mathematical description of the $\mathrm{M}^X|\mathrm{E}_N|1$ model.

4.1.2 Main result and its proof

Proposition 4.1 *The threshold characterization given by Theorem 3.1 extends to the $M^X|E_N|1$ model.*

Proof. Consider the Key Proposition for the original model. For $b\geq 1$, we replace inequalities (3.3) and (3.4) by

$$
\begin{aligned}
V_n(i{+}1,l,\mathrm{arr},b)-V_n(i,l,\mathrm{arr},b) &\geq\\
V_n(i{+}2,k,\mathrm{arr},b)-V_n(i{+}1,k,\mathrm{arr},b), &\qquad (4.1)\\
V_n(i{+}1,k{+}1,\mathrm{arr},b)-V_n(i,k{+}1,\mathrm{arr},b) &\geq\\
V_n(i{+}1,k,\mathrm{arr},b)-V_n(i,k,\mathrm{arr},b). &\qquad (4.2)
\end{aligned}
$$

Thus, we obtain a new Key Proposition, consisting of (3.1) and (3.2), (4.1) and (4.2), and (3.5) and (3.6). Its proof follows the lines of the proof of the Key Proposition for the original model. Since the DPEs for $V_n(i,k)$ and $V_n(i,k;\mathtt{ab})$ remain unaltered, and since we still have Fundamental Assumption 3.1 (and thus Proposition 3.2), it suffices to verify Steps 1 and 2 of that proof.

Let us first consider Step 1. For $b = 1$, we can follow the original derivations for (3.3) and (3.4). Next, under the main induction hypothesis regarding the induction on n, assume that (4.1) and (4.2) hold for some $b \geq 1$. This is our *imbedded* induction hypothesis. It is then straightforward to show that (4.1) and (4.2) also hold for $b+1$. We distinguish the same cases and choose the same decisions in order to apply Lemma 1.1, thereby obtaining inequalities which hold on the basis of the imbedded induction hypothesis.

To illustrate this procedure, consider, e.g., (4.2), and in particular case $\mathcal{RA}$. Then (cf. (3.13)), we obtain, for $b+1$,

$$\begin{aligned} V_n(i+1,k+1,\mathrm{arr},b+1;\mathtt{rj}) - V_n(i,k+1,\mathrm{arr},b+1) & \\ = \; & V_n(i+1,k+1,\mathrm{arr},b) - V_n(i,k+1,\mathrm{arr},b) \\ \geq \; & \{\textsf{imbedded induction hypothesis; (4.1)}\} \\ & V_n(i+2,k,\mathrm{arr},b) - V_n(i+1,k,\mathrm{arr},b) \\ = \; & V_n(i+1,k,\mathrm{arr},b+1) - V_n(i,k,\mathrm{arr},b+1;\mathtt{ac}), \end{aligned}$$

to which we can apply Lemma 1.1. The derivations for the other three cases (i.e., $\mathcal{AA}$, $\mathcal{AR}$ and $\mathcal{RR}$) and the four cases corresponding to inequality (4.1) are analogous.

Next, consider Step 2. This step is readily verified. Namely, it suffices to replace all $\lambda V_n(i,k,\mathrm{arr})$ terms by the corresponding $\sum_{b\in B}\lambda_b V_n(i,k,\mathrm{arr},b)$ terms, after which the desired result follows by the induction hypothesis in combination with (4.1) in the part concerning (3.5), and with (4.2) in the part concerning (3.6).

Finally, since arriving jobs may be accepted or rejected on a one by one basis, Corollaries 3.2 through 3.5 can be obtained from the new Key Proposition by means of the exact same proofs as given for the original model.

□

4.2 Extension to Erlang arrivals

In the previous section, we extended all monotonicity and threshold results from the standard Poisson arrival process to a batch Poisson arrival process, which allows for the modelling of more general arrival streams. However, in practical situations, one particular disadvantage of the Poisson process may be its relatively high coefficient of variation of the interarrival times of

jobs (or batches of jobs). Namely, (batch) Poisson arrivals occur completely random in time. We can obtain a more balanced arrival process by assuming, for example, Erlang arrivals. These are much more equally spread out over time than Poisson arrivals. In particular, we will consider Erlang-s arrivals, which means that each interarrival time consists of s exponential phases, each having mean duration $1/\lambda$. The arrivals become more balanced as s increases. For high values of s, the interarrival times of jobs will even become near-constant. So, the degree of variability in the interarrival times can be modelled by means of the model parameter s.

In this section, we show that the results stated and proved in Sections 3.2 and 3.3 extend to the $\mathrm{E}_s|\mathrm{E}_N|1$ version of the original model, which we have loosely described above. For this purpose, we introduce additional state variable a, which denotes the number of (exponential) arrival phases already completed towards the next arrival. So, at any time, the remaining number of arrival phases until the next arrival is $s-a$. We assume that this number is available to the decision maker at any time.

The introduction of a leads to a three-dimensional state space, involving states of the form (i,k,a). However, a is influenced by neither the decision to accept or reject a new job nor the decision to continue or abort the service of the job in service, and is therefore not subject to any control (as opposed to the other two state variables). Consequently, from an analytical point of view, the state space remains effectively two-dimensional. We will show that the threshold characterization given by Part 2 of Theorem 3.1 extends for fixed values of a. By means of a counterexample, we will demonstrate that such a threshold characterization can, in general, *not* be given for fixed i and k, and variable a. In other words, as it will turn out, the optimal termination policy is monotonic in both i and k, but *not* in a.

4.2.1 Dynamic Programming formulation

First, we note that we can use the same uniformization as in the original model. Next, for $0 \leq a \leq s$, we introduce the following notation: $V_n(i,k,a)$ denotes the maximum expected n-period reward when there are i jobs in the system, the job in service resides in node k, and a arrival phases have been completed towards the next arrival. In particular, $V_n(i,k,s)$ will replace the value function $V_n(i,k,\mathrm{arr})$ used in the original model. We then obtain the following complete new set of DPEs (where $0 \leq a < s$, $i \geq 1$ and $0 \leq k \leq N$):

$V_0(0,\cdot,a) = 0$

$V_0(i,k,a) = r(k) + (i-1)r(0)$

For $n \geq 0$:

$V_n(0,\cdot,s) = \mathbf{max}\{V_n(1,0,0) - c,\ V_n(0,\cdot,0) + r(0)\}$

$V_n(i,k,s) = \mathbf{max}\{V_n(i+1,k,0) - c,\ V_n(i,k,0) + r(0)\}$

For $n \geq 1$:

$V_n(0,\cdot,a) = V_n(0,\cdot,a;\mathtt{co})$

$V_n(0,\cdot,a;\mathtt{co}) = \lambda V_{n-1}(0,\cdot,a+1) + \mu V_{n-1}(0,\cdot,a)$

$V_n(i,k,a) = \mathbf{max}\{V_n(i,k,a;\mathtt{co}),\ V_n(i,k,a;\mathtt{ab})\}$

$V_n(i,k,a;\mathtt{co}) = \lambda V_{n-1}(i,k,a+1) + \mu V_{n-1}(i,(k+1)^-,a) - ih$

$V_n(i,k,a;\mathtt{ab}) = r(k) + V_n(i-1,0,a)$

This completes our mathematical description of the $\mathrm{E}_s|\mathrm{E}_N|1$ model.

4.2.2 Main result and its proof

Proposition 4.2 *The threshold characterization given by Theorem 3.1 extends to the $E_s|E_N|1$ model, where Part 2 holds for fixed a.*

Proof. We can basically maintain the Key Proposition for the original model, where (3.1), (3.2), (3.5) and (3.6) now hold for fixed a, and where states of the form (i,k,arr) appearing in (3.3) and (3.4) are replaced by their counterparts (i,k,s). For example, inequality (3.1) now reads as follows. For all $n \geq 0$, $i \geq 0$, $0 \leq k \leq N$, $0 \leq l \leq N$ and $0 \leq a < s$,

$$V_n(i+1,l,a) - V_n(i,l,a) \geq V_n(i+2,k,a) - V_n(i+1,k,a).$$

The proof of the Key Proposition for the $\mathrm{E}_s|\mathrm{E}_N|1$ model and the subsequent proofs of Corollaries 3.2 through 3.5 are analogous to the corresponding

proofs given for the original model. The only DPEs that have changed with respect to content are those for $V_n(i,k;\mathtt{co})$. However, it is easily verified that Step 2 of the proof of the Key Proposition does not change significantly. For example, the derivation of inequality (3.5) becomes as follows. For $i \geq 0$, $0 \leq k \leq N$, $0 \leq l \leq N$ and $0 \leq a < s$,

$$\begin{aligned}
&V_{n+1}(i+1,l,a;\mathtt{co}) - V_{n+1}(i,l,a;\mathtt{co}) \\
&\quad = \lambda[V_n(i+1,l,a+1) - V_n(i,l,a+1)] \; + \\
&\qquad \mu[V_n(i+1,(l+1)^-,a) - V_n(i,(l+1)^-,a)] \; - \; h \\
&\quad \geq \{\text{induction hypothesis; (3.3) if } a+1=s\text{; (3.1)}\} \\
&\qquad \lambda[V_n(i+2,k,a+1) - V_n(i+1,k,a+1)] \; + \\
&\qquad \mu[V_n(i+2,(k+1)^-,a) - V_n(i+1,(k+1)^-,a)] \; - \; h \\
&\quad = V_{n+1}(i+2,k,a;\mathtt{co}) - V_{n+1}(i+1,k,a;\mathtt{co}).
\end{aligned}$$

It is readily seen that the derivation of inequality (3.6) is analogous.

□

4.2.3 Counterexample

By means of Theorem 4.2 we have shown that the optimal policy for the $\mathrm{E}_s|\mathrm{E}_N|1$ model is monotonic in i and k. The following counterexample shows that the optimal (termination) policy need not be monotonic in a.

Counterexample 4.1 Consider the following instance of our model: $s=4$, $\lambda = \frac{4}{7}$, $\mu = \frac{3}{7}$, $h = \frac{3}{7}$, $c = 20$, $N = 6$ and $\underline{r} = (0, 20, 36, 49, 59, 69, 75)$. Note that the reward function satisfies Fundamental Assumption 3.1. For $n = 50$ and $(i,k) = (2,3)$, the (unique) optimal termination policy turns out to be as follows, where $\Delta(a) := V_{50}(2,3,a;\mathtt{co}) - V_{50}(2,3,a;\mathtt{ab})$ for $0 \leq a < 4$, and $V_{50}(1,0,a) = V_{50}(2,3,a;\mathtt{ab}) - r(3)$ for $0 \leq a < 4$.

(i,k,a)	optimal decision	$\Delta(a)$	$V_{50}(1,0,a)$
$(2,3,0)$	`continue`	0.412	189.019
$(2,3,1)$	`continue`	0.004	189.574
$(2,3,2)$	`abort`	−0.176	189.630
$(2,3,3)$	`continue`	0.002	189.373

From these results we can conclude that the optimal termination policy is not monotonic in a in state $(2,3,a)$, and neither is $V_{50}(\cdot)$. At first sight, this may seem counterintuitive, but it can be informally explained as follows.

The reason that `continue` is optimal in states $(2,3,0)$ and $(2,3,1)$ is probably that `abort` will induce a significant probability that the system soon runs out of jobs, and an empty system does not yield any reward. In state $(2,3,2)$, this probability of starvation is apparently not large enough, since we expect a new job to arrive shortly, and thus we can very well abort the job in service and start serving the queued job. In state $(2,3,3)$, the next arrival comes even sooner, but apparently *too* soon. It would be more convenient if it would arrive a little bit later, because then, in case of admittance in both situations, the holding costs for that job would be lower. Since delay of arrivals is beyond our control, it is (in respect of holding costs) apparently likely that the next new job will be rejected, and hence, because we also want to avoid starvation, it is optimal to continue the service of the job that is currently being served.

4.2.4 Phase-type arrivals

Our analysis of the extension to Erlang arrivals, and in particular Step 2 of the proof of the Key Proposition, reveals that the generalization of the monotonicity and threshold results carries over to general phase-type arrivals, characterized by arrival phase transition probabilities $q_{aa'}$, where λq_{as} is the probability of an arrival in the next period when the system is currently in some state (i,k,a). Without loss of generality, the phase-type arrival process is described by a Markov chain on the states $0 \leq a \leq s$, with starting state 0 after each arrival, and absorbing state s corresponding to the next arrival. Then, the $\mathrm{E}_s|\mathrm{E}_N|1$ model and the more general $\mathrm{PH}|\mathrm{E}_N|1$ model differ only with respect to the DPEs for $V_n(i,k,a;\mathtt{co})$. For example, for $n \geq 1$, $i \geq 1$, $0 \leq k \leq N$ and $0 \leq a < s$, we can write down the following DPE for the $\mathrm{PH}|\mathrm{E}_N|1$ model:

$$V_n(i,k,a;\mathtt{co}) = \sum_{0 \leq a' \leq s} \lambda\, q_{aa'}\, V_{n-1}(i,k,a') + \mu V_{n-1}(i,(k+1)^-,a) - ih$$

Consequently, we only need to verify Step 2 of the proof of the Key Proposition, and it is readily seen that both derivations are analogous to

the corresponding derivations for the $E_s|E_N|1$ model. For example, consider the first derivation. Then, for $i \geq 0$, $0 \leq k \leq N$, $0 \leq l \leq N$ and $0 \leq a < s$,

$$\begin{aligned}
&V_{n+1}(i+1,l,a;\mathsf{co}) - V_{n+1}(i,l,a;\mathsf{co}) \\
&\quad = \lambda \textstyle\sum_{0\leq a'\leq s} q_{aa'}\,[V_n(i+1,l,a') - V_n(i,l,a')] \;+ \\
&\qquad \mu[V_n(i+1,(l+1)^-,a) - V_n(i,(l+1)^-,a)] \;-\; h \\
&\quad \geq \{\text{induction hypothesis; (3.3) for } a'=s;\ (3.1)\} \\
&\qquad \lambda \textstyle\sum_{0\leq a'\leq s} q_{aa'}\,[V_n(i+2,k,a') - V_n(i+1,k,a')] \;+ \\
&\qquad \mu[V_n(i+2,(k+1)^-,a) - V_n(i+1,(k+1)^-,a)] \;-\; h \\
&\quad = V_{n+1}(i+2,k,a;\mathsf{co}) - V_{n+1}(i+1,k,a;\mathsf{co}).
\end{aligned}$$

4.3 Extension to Markov feed-forward routing

Characteristic of the service process in the original model is that the outcome of each service phase is *fixed*, i.e., known in advance and equal for all jobs that complete that phase. This means that, in principle, all jobs have the same routing through the service process, and, consequently, have the same eventual outcome. This is, however, not in accordance with most practical situations, because often the outcome of a phase will be random; see also Section 1.2.1. The original model does not account for this. In this section, we study a more general routing mechanism, which allows for so-called *jumps*. In this way, multiple routings can be modelled. Under two assumptions concerning the reward function and the service phase transition probabilities, we will show that the results obtained for the original model still hold if *jumps* are allowed for. The two assumptions replace Fundamental Assumption 3.1.

4.3.1 Description of the routing mechanism

In the original model, upon completion of some phase $m < N$, the job in service enters phase $m+1$ deterministically. We generalize this deterministic routing mechanism in the following way. A job that resides in node k and that completes its current service phase, will subsequently *jump* (i.e., move on) to node j with probability p_{kj}, $k \leq j \leq N$, where $\sum_{j=k}^{N} p_{kj} = 1$. Upon such a jump, all service phases associated with the nodes in between k and j are automatically completed.

This framework can act as a reasonable representation of a class of workflow processes encountered in practice, where after each task completion, depending on the outcome of the task, the case under consideration is redirected to some downstream task. By specifying a sufficient number of service phases (i.e., tasks), one can include as many alternative routings and outcomes of jobs (i.e., cases) as one desires.

See Figure 4.1 for a graphical representation of this (Markov) routing mechanism. Note that we only allow *forward* jumps, i.e., jobs will never return to nodes they have already passed. So the routing mechanism has a *feed-forward* structure. If $p_{kk} = 0$ for all $0 \leq k < N$, then jobs will always make jumps of size at least 1. If $p_{kk} > 0$ for some $0 \leq k < N$, then upon completion of corresponding service phase $k+1$, with probability p_{kk} this service phase has to be redone. In practice, this could for example be the case when it turns out that there have been technicalities or when new external information has arrived. Both could request a repeated investigation.

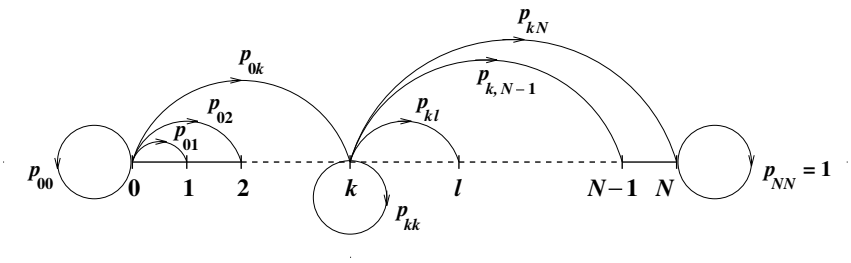

Figure 4.1: Markov feed-forward routing mechanism

4.3.2 New set of regularity assumptions

We replace Fundamental Assumption 3.1 by the following two important assumptions.

Fundamental Assumption 4.1
{Non-increasing expected direct rewards}
The expected direct reward function $s(k) := \sum_{j=k}^{N} p_{kj}[r(j) - r(k)]$ *is non-increasing in* k.

For $0 \leq k \leq N$, the expected direct reward $s(k)$ is the expected one-period increase in the overall reward for a job currently residing in node k, given

the decision to continue service. Fundamental Assumption 4.1 replaces the concavity part of Fundamental Assumption 3.1 and, in words, it reads that the more service phases have been completed for a job, the less rewarding it becomes to continue, from a one-period look-ahead point of view.

Fundamental Assumption 4.2 {No expected overtaking}
For all $0 \leq k < N$,

$$\forall_{M=k,\ldots,N}: \quad \sum_{j=k}^{M} p_{kj} \geq \sum_{j=k+1}^{M} p_{k+1,j}.$$

Fundamental Assumption 4.2 says that from a statistical point of view, a job residing in node k will not overtake a job residing in node $k+1$. This assumption is equivalent to the assumption that for all $0 \leq k < N$,

$$\forall_{M=k,\ldots,N}: \quad \sum_{j=M+1}^{N} p_{kj} \leq \sum_{j=M+1}^{N} p_{k+1,j}.$$

Remark 4.2 Given that the job in service currently resides in node k, for some $0 \leq k < N$, let discrete random variable X_k denote the node this job will reside in during the next period, provided service is not aborted. Then Fundamental Assumption 4.2 states that $X_k \leq_{\text{st}} X_{k+1}$, i.e., X_{k+1} is stochastically larger than X_k.

Remark 4.3 If jobs can only jump from node k to either $k+1$ or k itself, i.e., $p_{k,k+1} = 1 - p_{kk}$ for all $0 \leq k < N$, then Fundamental Assumption 4.2 is surely satisfied. This routing mechanism includes that of the original model.

In the remainder of this section, we will make extensive use of the following lemma.

Lemma 4.1 *Let Fundamental Assumption 4.2 be satisfied. Let $\xi(j)$ be a function which is defined on the points $j = 0, 1, \ldots, N$ and which is non-increasing [non-decreasing] in j. Then $\sum_{j=k}^{N} p_{kj}\, \xi(j)$ is non-increasing [non-decreasing] in k, $0 \leq k \leq N$.*

Proof. Cf. Remark 4.2 and use for example Proposition 9.1.2 of Ross [49].

□

In the discussion of the original model, we noted (see Remark 3.2) that under the assumption that the reward function $r(k)$ is concave, we may assume without loss of generality that $r(k)$ is also non-decreasing. In the M|E$_{N,\,jumps}$|1 model, this assumption can be obtained from Fundamental Assumptions 4.1 and 4.2, and we will write it down as a corollary below. The proof is more intricate than the short proof given for the original model. We will make use of the following lemma, which uses the notation introduced in Remark 4.2. More generally, given that the job in service currently resides in node k, let discrete random variable $X_k^{(m)}$ denote the node this job will reside in $m \geq 1$ periods from now, provided service is not aborted prematurely.

Lemma 4.2 *If*

$$\forall_{0\leq k<N}: \quad X_k \quad \leq_{st} X_{k+1},$$

then

$$\forall_{m\geq 1}\forall_{0\leq k<N}: \quad X_k^{(m)} \leq_{st} X_{k+1}^{(m)}.$$

Proof. For any $M \geq 0$, $m \geq 1$ and starting node $0 \leq k \leq N$, define $q_m(k)$ by $q_m(k) := \mathbb{P}(X_k^{(m)} > M)$. It remains to show that $q_m(k)$ is non-decreasing in k for fixed M and m. We prove this by induction on m. It is immediate from Fundamental Assumption 4.2 that $q_1(k)$ is non-increasing in k. Assume that $q_m(k)$ is non-decreasing in k for some $m \geq 1$. This will be our induction hypothesis. Then

$$\begin{array}{rcl} q_{m+1}(k) & = & \sum_{j=0}^{N} p_{kj}\, \mathbb{P}\left[X_k^{(m+1)} > M \;\middle|\; X_k = j \right] \\ & = & \sum_{j=0}^{N} p_{kj}\, q_m(j) \\ & = & \{p_{kj} = 0 \text{ for } j < k\} \\ & & \sum_{j=k}^{N} p_{kj}\, q_m(j). \end{array}$$

By the induction hypothesis, $q_m(j)$ is non-decreasing in j. Therefore, by Lemma 4.1, $\sum_{j=k}^{N} p_{kj}\, q_m(j)$ is non-decreasing in k, and hence $q_{m+1}(k)$ is non-decreasing in k.

□

Corollary 4.1 *The reward function $r(k)$ is non-decreasing in k.*

Proof. For $0 \leq k \leq N$, the reward $r(k)$ can be written as

$$r(k) \;=\; -s(k) \;+\; \textstyle\sum_{j=k}^{N} p_{kj}\, r(j),$$

or, in matrix-vector notation, as

$$\underline{r} \;=\; -\underline{s} \;+\; P\,\underline{r},$$

where P is the transition matrix. Since, without loss of generality, Fundamental Assumption 4.2 excludes p_{kk} being 1 for some $0 \leq k < N$, the jumping process for the job in service is transient to node N. Therefore, by iteration, we obtain

$$\underline{r} \;=\; -\sum_{m=0}^{\infty} P^{\,m}\underline{s} \;+\; r(N)\,\underline{e},$$

where $\underline{e} := (\underbrace{1, 1, \ldots, 1}_{N+1})$. Hence, since $(\sum_{m=0}^{\infty} P^{\,m}\underline{s})(k)$ is non-increasing in k by Fundamental Assumption 4.1 and Lemma 4.2, we obtain that $r(k)$ is non-decreasing in k.

□

4.3.3 Dynamic Programming formulation

We can adopt the DPEs for the original model, where, for all $n \geq 1$, $i \geq 1$ and $0 \leq k \leq N$, the DPEs for $V_n(i, k; \mathtt{co})$ are replaced by

$$\left|\; V_n(i,k;\mathtt{co}) = \lambda V_{n-1}(i,k,\mathrm{arr}) + \mu \textstyle\sum_{j=k}^{N} p_{kj} V_{n-1}(i,j) - ih \right.$$

4.3.4 Main result and its proof

Proposition 4.3 *Let Fundamental Assumption 3.1 be replaced by Fundamental Assumptions 4.1 and 4.2. Then the threshold characterization given by Theorem 3.1 extends to the $M|E_{N,\,jumps}|1$ model.*

Proof. The proof follows the lines of the proof for the original model. The Key Proposition and Proposition 3.2 remain unaltered. Since (only) the DPEs for $V_n(i,k,\texttt{co})$ and the regularity conditions have changed, we have to verify (and it suffices to verify) Step 2 of the proof of the Key Proposition and the proof of Proposition 3.2. We will first prove Proposition 3.2 for the M|$E_{N,\,jumps}$|1 model, and then consider Step 2 of the proof of the Key Proposition for the M|$E_{N,\,jumps}$|1 model.

For convenience, we rewrite Proposition 3.2 as follows.

Proposition 4.4 *For all $i \geq 0$ and $0 \leq k < N$,*

$$\begin{aligned} V_n(i,k+1) - V_n(i,k) &\leq r(k+1) - r(k), && (4.3)\\ V_n(i,k+1,\text{arr}) - V_n(i,k,\text{arr}) &\leq r(k+1) - r(k), && (4.4)\\ V_n(i,k+1;\texttt{co}) - V_n(i,k;\texttt{co}) &\leq r(k+1) - r(k), && (4.5) \end{aligned}$$

where (4.3) and (4.4) hold for all $n \geq 0$, and (4.5) holds for all $n \geq 1$.

Proof. The proof uses induction on n, and is organized as follows. Inequality (4.3) holds by definition for $n = 0$. Assuming that (4.3) holds for some $n \geq 0$, we prove that (4.4) holds for n (*Step I*). Using this result, we prove that (4.5) holds for $n+1$ (*Step II*). Finally, we prove that (4.3) holds for $n+1$ as well (*Step III*).

Step I. Assume that (4.3) holds for some $n \geq 0$. This will be our induction hypothesis. Let $i \geq 0$ and $0 \leq k < N$. The next decision prescribed by the (optimal) policy corresponding to $V_n(i,k+1,\text{arr})$, is either to accept or to reject the new job.

Assume that `accept` is optimal. Then

$$\begin{aligned} V_n(i,k+1,\text{arr}) - V_n(i,k,\text{arr}) &= V_n(i,k+1,\text{arr};\texttt{ac}) - V_n(i,k,\text{arr})\\ &\leq V_n(i,k+1,\text{arr};\texttt{ac}) - V_n(i,k,\text{arr};\texttt{ac})\\ &= V_n(i+1,k+1) - c - [V_n(i+1,k) - c]\\ &= V_n(i+1,k+1) - V_n(i+1,k)\\ &\leq \{\text{induction hypothesis}\}\\ &\quad r(k+1) - r(k). \end{aligned}$$

Alternatively, assume that `reject` is optimal. Then

$$\begin{aligned} V_n(i,k+1,\mathrm{arr}) - V_n(i,k,\mathrm{arr}) &= V_n(i,k+1,\mathrm{arr};\mathtt{rj}) - V_n(i,k,\mathrm{arr}) \\ &\leq V_n(i,k+1,\mathrm{arr};\mathtt{rj}) - V_n(i,k,\mathrm{arr};\mathtt{rj}) \\ &= V_n(i,k+1) + r(0) - [V_n(i,k) + r(0)] \\ &= V_n(i,k+1) - V_n(i,k) \\ &\leq \{\text{induction hypothesis}\} \\ &\quad r(k+1) - r(k). \end{aligned}$$

Step II. Let $i \geq 0$ and $0 \leq k < N$. Then

$$\begin{aligned} V_{n+1}(i,k+1;\mathtt{co}) - V_{n+1}(i,k;\mathtt{co}) &= \lambda[V_n(i,k+1,\mathrm{arr}) - V_n(i,k,\mathrm{arr})] + \\ &\quad \mu[\textstyle\sum_{j=k+1}^{N} p_{k+1,j} V_n(i,j) \; - \\ &\qquad \textstyle\sum_{j=k}^{N} p_{kj} V_n(i,j)] \\ &\leq \{\text{induction hypothesis; (4.4)}\} \\ &\quad \lambda[r(k+1) - r(k)] + \\ &\quad \mu[\textstyle\sum_{j=k+1}^{N} p_{k+1,j} V_n(i,j) \; - \\ &\qquad \textstyle\sum_{j=k}^{N} p_{kj} V_n(i,j)]. \quad (4.6) \end{aligned}$$

According to the induction hypothesis, $V_n(i,j) - r(j)$ is non-increasing in j. Therefore, by Lemma 4.1, $\sum_{j=k}^{N} p_{kj}[V_n(i,j) - r(j)]$ is non-increasing in k. Furthermore, Fundamental Assumption 4.1 states that $\sum_{j=k}^{N} p_{kj}[r(j) - r(k)]$ is non-increasing in k. Note that

$$\sum_{j=k}^{N} p_{kj}[V_n(i,j) - r(k)] = \sum_{j=k}^{N} p_{kj}[V_n(i,j) - r(j)] + \sum_{j=k}^{N} p_{kj}[r(j) - r(k)],$$

so $\sum_{j=k}^{N} p_{kj}[V_n(i,j) - r(k)]$ is also non-increasing in k.

Therefore,

$$\sum_{j=k+1}^{N} p_{k+1,j}[V_n(i,j) - r(k+1)] \leq \sum_{j=k}^{N} p_{kj}[V_n(i,j) - r(k)],$$

i.e.,

$$\sum_{j=k+1}^{N} p_{k+1,j} V_n(i,j) - \sum_{j=k}^{N} p_{kj} V_n(i,j) \leq r(k+1) - r(k). \quad (4.7)$$

Plugging (4.7) into (4.6), we obtain

$$\begin{aligned} V_{n+1}(i,k+1;\texttt{co}) - V_{n+1}(i,k;\texttt{co}) &\leq (\lambda+\mu)[r(k+1)-r(k)] \\ &\leq \{\lambda+\mu \leq 1\} \\ & \quad r(k+1)-r(k). \end{aligned}$$

Step III. Let $i \geq 0$ and $0 \leq k < N$. The next decision prescribed by the (optimal) policy corresponding to $V_{n+1}(i,k+1)$, is either to continue or to abort the job under service, where `continue` is prescribed by definition for $i = 0$.

Assume that `continue` is optimal. Then

$$\begin{aligned} V_{n+1}(i,k+1) - V_{n+1}(i,k) &= V_{n+1}(i,k+1;\texttt{co}) - V_{n+1}(i,k) \\ &\leq V_{n+1}(i,k+1;\texttt{co}) - V_{n+1}(i,k;\texttt{co}) \\ &\leq \{\text{induction hypothesis; (4.5)}\} \\ & \quad r(k+1)-r(k). \end{aligned}$$

Alternatively, assume that `abort` is optimal. Note that this implies $i > 0$. Then

$$\begin{aligned} V_{n+1}(i,k+1) - V_{n+1}(i,k) &= V_{n+1}(i,k+1;\texttt{ab}) - V_{n+1}(i,k) \\ &\leq V_{n+1}(i,k+1;\texttt{ab}) - V_{n+1}(i,k;\texttt{ab}) \\ &= r(k+1) + V_{n+1}(i-1,0;\texttt{co}) \; - \\ & \qquad\qquad [r(k) + V_{n+1}(i-1,0;\texttt{co})] \\ &= r(k+1)-r(k). \end{aligned}$$

This concludes our proof of Proposition 4.4.

□

Next, we shift our attention to Step 2 of the proof of the Key Proposition for the M|E$_{N,\,jumps}$|1 model. As in the proof of Proposition 4.4, we will make use of Lemma 4.1.

Let $i \geq 0$. Then, for $0 \leq k \leq N$ and $0 \leq l \leq N$,

$$\begin{aligned}
&V_{n+1}(i+1,l;\mathsf{co}) - V_{n+1}(i,l;\mathsf{co}) \\
&\qquad = \lambda[V_n(i+1,l,\mathrm{arr}) - V_n(i,l,\mathrm{arr})] + \\
&\qquad\qquad \mu\textstyle\sum_{j=l}^{N} p_{lj}[V_n(i+1,j) - V_n(i,j)] - h \\
&\qquad \geq \{\text{induction hypothesis; (3.3); (3.1)}\} \\
&\qquad\quad \lambda[V_n(i+2,k,\mathrm{arr}) - V_n(i+1,k,\mathrm{arr})] + \\
&\qquad\qquad \mu\textstyle\sum_{j=l}^{N} p_{lj}[V_n(i+2,N) - V_n(i+1,N)] - h \\
&\qquad = \lambda[V_n(i+2,k,\mathrm{arr}) - V_n(i+1,k,\mathrm{arr})] + \\
&\qquad\qquad \mu\textstyle\sum_{j=k}^{N} p_{kj}[V_n(i+2,N) - V_n(i+1,N)] - h \\
&\qquad \geq \{\text{induction hypothesis; (3.2)}\} \\
&\qquad\quad \lambda[V_n(i+2,k,\mathrm{arr}) - V_n(i+1,k,\mathrm{arr})] + \\
&\qquad\qquad \mu\textstyle\sum_{j=k}^{N} p_{kj}[V_n(i+2,j) - V_n(i+1,j)] - h \\
&\qquad = V_{n+1}(i+2,k;\mathsf{co}) - V_{n+1}(i+1,k;\mathsf{co})
\end{aligned}$$

and, for $0 \leq k < N$,

$$\begin{aligned}
&V_{n+1}(i+1,k+1;\mathsf{co}) - V_{n+1}(i,k+1;\mathsf{co}) \\
&\qquad = \lambda[V_n(i+1,k+1,\mathrm{arr}) - V_n(i,k+1,\mathrm{arr})] + \\
&\qquad\qquad \mu\textstyle\sum_{j=k+1}^{N} p_{k+1,j}[V_n(i+1,j) - V_n(i,j)] - h \\
&\qquad \geq \{\text{induction hypothesis; (3.4); (3.2);} \\
&\qquad\qquad \text{Lemma 4.1 with } \xi(j) = V_n(i+1,j) - V_n(i,j) \\
&\qquad\qquad\qquad \text{(non-decreasing in } j)\} \\
&\qquad\quad \lambda[V_n(i+1,k,\mathrm{arr}) - V_n(i,k,\mathrm{arr})] + \\
&\qquad\qquad \mu\textstyle\sum_{j=k}^{N} p_{kj}[V_n(i+1,j) - V_n(i,j)] - h \\
&\qquad = V_{n+1}(i+1,k;\mathsf{co}) - V_{n+1}(i,k;\mathsf{co}).
\end{aligned}$$

□

4.3.5 Phase-dependent service rates

The original model features homogeneous service rates, i.e., the expected amount of time required to complete service phase m (between 1 and N) is constant in m. From a practical point of view, this is a highly restrictive condition. However, the extension to heterogeneous, phase-dependent

service rates is readily obtained as a special case of the $M|E_{N,\,jumps}|1$ model. Let the service rate corresponding to node k (i.e., service phase $k+1$) be μ_k for $0 \leq k < N$. Then the model is captured by the $M|E_{N,\,jumps}|1$ model by taking $\mu = \max_{0 \leq k < N} \mu_k$, $p_{k,k+1} = 1 - p_{kk} = \frac{\mu_k}{\mu}$ for $0 \leq k < N$, and $p_{NN} = 1$. Note that this special case surely satisfies Fundamental Assumption 4.2.

4.3.6 Counterexamples

It may be clear that the generalized threshold characterization need not hold anymore if Fundamental Assumption 4.1 is violated. We can use the same counterexamples as given in Section 3.5 to illustrate this, since the original model is a special case of the $M|E_{N,\,jumps}|1$ model. In addition, the following two counterexamples demonstrate that violations of the other regularity condition, i.e., Fundamental Assumption 4.2, also erode away the generalized monotonicity results.

Counterexample 4.2 Consider the following set of instances of our model: $\mu = 1$ (so $\lambda = 0$ and $\alpha = 0$), $h = 0$, $N = 5$, $\underline{r} = (0, 0, 2, 2, 3, 4)$, $p_{01} = \zeta$, $p_{03} = p_{12} = p_{25} = p_{34} = p_{45} = 1 - \zeta$ and $p_{kk} = \zeta$ for $1 \leq k \leq 4$, where $\zeta \in [0, 1)$. Then $\underline{s} = (1 - \zeta)(2, 2, 2, 1, 1, 0)$, and hence Fundamental Assumption 4.1 is satisfied, whereas Fundamental Assumption 4.2 is not.

Let $\zeta \downarrow 0$. Further, let $i = 1$ and $n = 2$, i.e., there is one job in the system and we may serve this job for two periods of time at no cost. It is readily verified that $V_2(1, 0) = V_2(1, 0; \mathtt{co}) = 3$ and $V_2(1, 1) = V_2(1, 1; \mathtt{co}) = 4$, so

$$V_2(1, 1) - V_2(1, 0) = V_2(1, 1; \mathtt{co}) - V_2(1, 0; \mathtt{co}) = 1 > 0 = r(1) - r(0),$$

which contradicts both (4.3) and (4.5). In fact, the whole concept of not discarding jobs from the queue if the job in service is not aborted (as stated by Corollary 3.1) is lost. For example, take $i = 2$ and $h = 1.75$. Then, for $n = 2$, the optimal decision in state $(2, 1)$ will be to discard the queued job and to continue the service of the job currently in service, yielding a total reward of $4 - 2(1.75) = 0.5$. Aborting the job in service and serving the queued job would yield a maximum total reward of only 0.25.

Furthermore, the following counterexample illustrates that a violation of Fundamental Assumption 4.2 can also heavily affect the structure of the optimal admission policy.

Counterexample 4.3 Consider the following instance of our model: $\mu=1$ (so $\lambda = 0$ and $\alpha = 0$), $h = 0$, $c = 8$, $N = 10$, $r(0) = 0$, $r(1) = 6$, $r(k) = k$ for $2 \leq k \leq 10$, $p_{01} = \frac{9}{10}$, $p_{02} = \frac{1}{10}$, $p_{1,10} = 1$ and $p_{k,k+1} = 1$ for $2 \leq k \leq 9$. Then $s(0) = 5.6$, $s(1) = 4$ and $s(k) = 1$ for $2 \leq k \leq 9$, and hence Fundamental Assumption 4.1 is satisfied, whereas Fundamental Assumption 4.2 is not.

Let $i = 1$ and $n = 8$, and let there be an arrival event at this point in time (which will be the only arrival, since $\lambda = 0$). It is straightforward to check that

$$V_8(1, 2, \text{arr}; \texttt{ac}) = 9.3 < 10 = V_8(1, 2, \text{arr}; \texttt{rj}),$$

but

$$V_8(1, 1, \text{arr}; \texttt{ac}) = 11.8 > 10 = V_8(1, 1, \text{arr}; \texttt{rj}).$$

Thus, in state $(1, 2)$ it is optimal to reject, whereas in state $(1, 1)$ it is optimal to accept. This contradicts Corollary 3.3. Furthermore, with a little more effort, it can be verified that

$$V_8(2, 1, \text{arr}; \texttt{ac}) = 21.17 > 19.8 = V_8(2, 1, \text{arr}; \texttt{rj}),$$

so in state $(2, 1)$ it is optimal to accept, which contradicts Corollary 3.2.

4.4 Translation to deterministic decision epochs

Both the feed-forward routing mechanism of the $\text{M}|\text{E}_{N,\,jumps}|1$ model and the two regularity conditions imposed on the reward structure and the transition probabilities originate from Brouns and Van der Wal [11]. There, we studied in isolation a discrete-time single-server batch queueing model with controlled service times consisting of a sum of at most N geometric phases. We will show that this batch model, which we term the $\text{X}|\text{Geom}_N|1$ model in the remainder, is essentially a special case of the $\text{M}|\text{E}_{N,\,jumps}|1$ model. Further, we will introduce and analyse a model in which the outcome of a job is either 'success' or 'no success'. This model has its origins in the batch model. What is more, we will discuss briefly the incorporation of an arrival process in the batch model, so that we can allow for a dynamic supply of jobs, instead of just the one initial pile of work. First, however, we give a brief motivation of our discussion of deterministic decision epochs.

4.4.1 Motivation

The continuous-time models discussed and studied in this thesis exhibit exponentially distributed decision epochs. However, in practice, it may sometimes be desirable to consider *deterministic* instead of exponentially distributed decision epochs. This is, for example, of interest when there is a fixed and clear deadline after which no more jobs may be served. Continuous-time models are unsuited for this type of situation, because, as a consequence of uniformization, the total length of time is not fixed for the finite horizon problem. Moreover, in practical situations involving human decision-making and manual execution of cases, the state of the process is typically observed and decisions are typically made periodically, i.e., at fixed, equidistant points in time (e.g., every hour) rather than when an event occurs. This system of periodic inspections calls for a somewhat different model, which can, however, be obtained by means of a transformation of its discretized continuous-time counterpart. In the remainder, to illustrate the concept, we focus on the translation of the $\mathrm{M}|\mathrm{E}_{N,\,jumps}|1$ model to deterministic decision epochs.

4.4.2 The $\mathbf{X}|\mathbf{Geom}_{\boldsymbol{N}}|\mathbf{1}$ model

Consider the following version of the $\mathrm{M}|\mathrm{E}_{N,\,jumps}|1$ model with finite horizon n, and no discounting. At the beginning of the process, we are confronted with a single batch of jobs awaiting service. For the treatment of these jobs we are allowed a fixed span of time, which consists of a fixed number of periods n of equal duration, e.g., n hours, days, or weeks. There are no future arrivals, or at least no arrivals before the end of the planning horizon. At the beginning of each period, it has to be decided whether to continue the service of the job currently in service or to abort service and commence service of the next job. The rewards, costs and routing mechanism are as in the $\mathrm{M}|\mathrm{E}_{N,\,jumps}|1$ model. In particular, the probability that a job residing in node k will jump to node j in the oncoming period is p_{kj}. Hence, the time required to complete service phase $k+1$ is geometrically distributed with parameter p_{kk}. Note that if intermediate phases cannot be skipped (i.e., if the routing mechanism is as in Chapter 3), then p_{kj} can also be used to represent the probability that a job which has already completed k service phases will complete exactly $j-k$ phases in the oncoming period, i.e., before the next decision epoch.

Now, the model we have just described is exactly the $\mathrm{X}|\mathrm{Geom}_N|1$ model studied in isolation in [11]. However, the threshold characterization of the optimal termination policy derived there can also be obtained directly from the $\mathrm{M}|\mathrm{E}_{N,\,jumps}|1$ model by recognizing that the $\mathrm{X}|\mathrm{Geom}_N|1$ model is a special case of the $\mathrm{M}|\mathrm{E}_{N,\,jumps}|1$ model. It is obtained simply by taking $\lambda = 0$ and $\mu = 1$.

4.4.3 Incorporation of an arrival process

The $\mathrm{X}|\mathrm{Geom}_N|1$ model does not feature any arrivals. However, following our discussion of the $\mathrm{M}^X|\mathrm{E}_N|1$ model in Section 4.1, the incorporation of arrivals is fairly straightforward. The idea is as follows. During a period in between two equidistant decision epochs, jobs arrive according to some arbitrary but period-independent arrival process. The state of the system is observed at the end of such a period only, and only then decisions are made as regards the termination of any jobs and the admission of any newly arrived jobs. So, during a period, arrivals accumulate, essentially to form a batch of jobs seeking admittance. At the end of the period, the decision maker observes this batch of jobs and, following the same principle as in the $\mathrm{M}^X|\mathrm{E}_N|1$ model, accepts or rejects these jobs on a one by one basis. This model, which we term the $\mathrm{D}^X|\mathrm{Geom}_N|1$ model in the remainder of this section, differs from the $\mathrm{M}^X|\mathrm{E}_N|1$ model in the sense that in the latter, when the system is observed, there was either an arrival or a service phase completion, but never both, whereas in the $\mathrm{D}^X|\mathrm{Geom}_N|1$ model, multiple arrivals and multiple service phase completions may have taken place. Note that from a decision maker point of view, these events all take place simultaneously.

In our model notation, discrete random variable X denotes the number of arrivals during one period. We may write $X \in B \cup \{0\}$, where B plays the same role as in Section 4.1. Denote $B' := B \cup \{0\}$ and $\gamma_b := \mathbb{P}(X = b)$ for $b \in B'$. Note that once we have formulated the $\mathrm{D}^X|\mathrm{Geom}_N|1$ model, the $\mathrm{X}|\mathrm{Geom}_N|1$ model can be obtained from it by choosing $B' = \{0\}$.

We obtain the $\mathrm{D}^X|\mathrm{Geom}_N|1$ model from the $\mathrm{X}|\mathrm{Geom}_N|1$ model as follows. For all $n \geq 1$, $i \geq 0$ and $0 \leq k \leq N$, replace the DPE for $V_n(i, k; \mathtt{co})$ for the $\mathrm{X}|\mathrm{Geom}_N|1$ model by

$$V_n(i,k;\mathtt{co}) = \sum_{j=k}^{N} p_{kj}\big[\gamma_0 V_{n-1}(i,j) + \sum_{b\in B} \gamma_b V_{n-1}(i,j,\mathrm{arr},b)\big] - ih$$

The value function $V_n(i, j, \text{arr}, b)$ has been adopted from Section 4.1.1. Here, it serves exactly the same purpose. I.e., batches can be 'peeled off', until the last job of the batch has been considered. Subsequently, the decision whether or not to abort the service of the job currently in service is taken. This corresponds to the recursive scheme given in Section 4.1.1.

Proposition 4.5 *Let Fundamental Assumptions 4.1 and 4.2 apply. The optimal admission/termination policy for the $D^X|Geom_N|1$ model satisfies Theorem 3.1.*

Proof. The proof is virtually a combination of the proofs of Propositions 4.1 and 4.3. We adopt the Key Proposition for the $M^X|E_N|1$ model, and may copy Step 1 of its proof from that model. It remains to verify Steps 2 and 3 of the proof. We follow the lines of the proof given for the $M|E_{N,\,jumps}|1$ model. Note that with respect to Step 3, it suffices to verify Steps I and II of the proof of Proposition 4.4, where, for all $b \in B$, inequality (4.4) is replaced by

$$V_n(i, k+1, \text{arr}, b) - V_n(i, k, \text{arr}, b) \;\leq\; r(k+1) - r(k).$$

First, consider Step I. If it is optimal in state (i, k, arr, b) to accept $a \leq b$ jobs and reject the other $b - a$, then we choose to accept exactly a jobs in state $(i, k+1, \text{arr}, b)$ as well. Then the result is immediate from the induction hypothesis.

Next, consider Step II. The proof is straightforward, and uses the fact that the derivation that resulted in inequality (4.7) can also be applied if all $V_n(i, j)$ terms are replaced by $V_n(i, j, \text{arr}, b)$ for any $b \in B$.

Finally, consider Step 2 of the proof of the Key Proposition. The proof is analogous to the proof given for the $M|E_{N,\,jumps}|1$ model. It uses the induction hypothesis in combination with (3.1) and (4.1) for each $b \in B$, and subsequently the induction hypothesis in combination with (4.2) for each $b \in B$ to obtain inequality (3.5) for all $i \geq 0$, $0 \leq k \leq N$ and $0 \leq l \leq N$. Next, it uses the induction hypothesis in combination with (3.2), (4.2) for each $b \in B$ and Lemma 4.1 with $\xi(j) = V_n(i+1, j) - V_n(i, j)$ as well as $\xi(j) = V_n(i+1, j, \text{arr}, b) - V_n(i, j, \text{arr}, b)$ for each $b \in B$ separately to obtain inequality (3.6) for all $i \geq 0$ and $0 \leq k < N$.

□

4.4.4 A 'Decreasing Success Rate' model

The X|Geom$_N$|1 and D^X|Geom$_N$|1 models can be used for the following special purpose. Consider a finite time horizon. Assume jobs are served either successfully (yielding a reward of 1) or unsuccessfully (yielding a reward of 0). For any job, let discrete random variable U denote the number of periods of time required until success. Define, for all $k \geq 1$,

$$F(k) := \mathbb{P}(U \leq k), \quad p_k := \mathbb{P}(U = k),$$

and

$$q_{k-1} := \mathbb{P}(U = k \mid U > k-1) = \frac{p_k}{1 - F(k-1)} = \frac{p_k}{1 - \sum_{j=1}^{k-1} p_j},$$

where $\sum_{k=1}^{\infty} p_k = 1$. The function q_k represents the *success rate*. Assume this success rate is non-increasing in k, i.e., the longer we work on a job, the less likely it becomes and the more time it will take before we will ultimately have success with respect to this job. This 'Decreasing Success Rate' model—which we term the DSR model in the remainder of this section—can be derived from the X|Geom$_N$|1 (or D^X|Geom$_N$|1) model as follows.[5] First, instead of a maximum number of service phases N, we define a node '∞'. Let $r(k) = 0$ for all finite $k \geq 0$ and let $r(\infty) = 1$. Further, let $p_{k,\infty} = q_k$ and $p_{k,k+1} = 1 - q_k$ for all finite $k \geq 0$ and let $p_{\infty,\infty} = 1$. Note that $s(k)$ is non-increasing in k. Further, note that one may easily show that it will be optimal to abort in node ∞.

Then, for $n \geq 1$, $i \geq 1$ and $k \geq 0$, the new DPE for $V_n(i, k; \mathtt{co})$ becomes

$$\begin{aligned} V_n(i,k;\mathtt{co}) = q_k \left[1 + \gamma_0 V_{n-1}(i-1,0) + \textstyle\sum_{b\in B} \gamma_b V_{n-1}(i-1,0,\mathrm{arr},b)\right] + \\ (1-q_k)\left[\gamma_0 V_{n-1}(i,k+1) + \textstyle\sum_{b\in B} \gamma_b V_{n-1}(i,k+1,\mathrm{arr},b)\right] - ih \end{aligned}$$

In the X|Geom$_N$|1 (or D^X|Geom$_N$|1) model, we assumed a finite maximum number of service phases. This is a restrictive assumption, because it implies that, in principle, all jobs can be served successfully. It might take a very long time for a specific job to reach success, but as long as the deadline is sufficiently far away, the probability of success is strictly positive. This does typically not accord with processes such as the examination of tax returns at a taxation office, as considered in the following example.

[5] This model has also been addressed in [11], but the description there is fallacious.

Example 4.1 Consider a department of a taxation office in which a huge pile of tax returns is searched for irregularities. There will always be tax returns that contain irregularities pointing in the direction of fraud, and it is one of the duties of the taxation office to detect these. However, on the other side of the spectrum stands the respectable taxpayer, whose returns are commonly 'clean as a whistle'. No matter how much effort is put in the examination of such a return, no anomalies will be found.

If an irregularity is detected, the return is sent to another department for an in-depth investigation. The examination process consists of checks. Checks incurring a high probability of finding an irregularity are carried out first. Should these not uncover anything, then checks which are increasingly less likely to result in the detection of an irregularity can be carried out, until one is finally found, or until it is decided to give up the search and start the examination of another tax return. The time required to perform a check is deterministic and equal for all checks, and the decision whether to continue or abort the investigation of a return is made after each check. Assume each employee examining tax returns has his or her own pile of returns, which is not shared with other employees. Then the examination process at an individual employee's desk can be captured by means of our DSR model.

An important observation for our DSR model is that although it satisfies Fundamental Assumption 4.1, it only satisfies Fundamental Assumption 4.2 if the success rate is constant. Nevertheless, one might think and conjecture that the threshold characterization of the optimal admission/threshold policy as given by Theorem 3.1 remains valid. However, with respect to the optimal admission policy, this presumption can be fully discharged. The following two counterexamples demonstrate that, in general, Part 1 of Theorem 3.1 does not apply at all to the DSR model.

Counterexample 4.4 Consider the following instance of our model: $h = 0.1$, $c = 0.5975$, $q_0 = 0.9$, $q_1 = 0.3$, $q_2 = 0.25$ and $q_k = 0.11$ for $k \geq 3$. Let $n = 2$ and let there be one incidental arrival to the system, and no future arrivals. Then,

$$V_2(1, 0, \text{arr}; \texttt{ac}) = V_2(2, 0) - c = 0.9025 \ > \ 0.82 = V_2(1, 0, \text{arr}; \texttt{rj}),$$

but

$$\begin{aligned} V_2(1, 1, \text{arr}; \texttt{ac}) = V_2(2, 1) - c &= \mathbf{max}\{\texttt{co} : 0.9, \ \texttt{ab} : 0.82\} - 0.5975 \\ &= 0.3025 \ < \ 0.305 = V_2(1, 1, \text{arr}; \texttt{rj}). \end{aligned}$$

Hence, the presumption that "if it is optimal to reject an arriving job in state $(i, k+1)$, then it is optimal as well to reject it in state (i, k)" does not hold, and, consequently, neither does inequality (3.2).

Note, by the way, that it is easily verified that the reverse implication does not hold either, since, e.g.,

$$V_2(1, 3, \text{arr}; \texttt{ac}) = 0.2225 \ > \ 0.0189 = V_2(1, 3, \text{arr}; \texttt{rj}).$$

Counterexample 4.5 Consider the same instance as in the previous counterexample. Again, consider the situation of $n = 2$ and one incidental arrival to the system, without any future arrivals. We already know from Counterexample 4.4 that $V_2(1, 1, \text{arr}; \texttt{ac}) < V_2(1, 1, \text{arr}; \texttt{rj})$. However,

$$\begin{aligned} V_2(2, 1, \text{arr}; \texttt{ac}) &= V_2(3, 1) - c \\ &= V_2(2, 0) - c = 0.9025 \ > \ 0.9 = V_2(2, 1, \text{arr}; \texttt{rj}). \end{aligned}$$

Hence, the presumption that "if it is optimal to reject an arriving job in state (i, k), then it is optimal as well to reject it in state $(i + 1, k)$" does not hold, and, consequently, neither does inequality (3.1). Because of its counterintuitive nature, this is quite a remarkable observation.

Remark 4.4 The sole fact that there is no longer a finite maximum number of service phases is not responsible for the loss of structure of the optimal admission policy. Namely, although we assumed that all rewards are finite (cf. Section 3.1), we did *not* state explicitly that the maximum number of service phases should be finite. We omit the details here, but one may verify that the finite horizon results obtained for the $\text{M}|\text{E}_{N,\,jumps}|1$, $\text{X}|\text{Geom}_N|1$ and $\text{D}^X|\text{Geom}_N|1$ models extend to the situation of a countably infinite set of nodes.

Next, let us consider Part 2 of Theorem 3.1, which characterizes the optimal termination policy. Counterexample 4.5 has shown that inequality (3.1) need not hold for the DSR model. Consequently, there is little hope of being able to obtain the monotonicity of the optimal termination policy in the number of jobs via our inductive DP approach, should it even hold. The only threshold result we can show to hold, in general, is the monotonicity of the optimal termination policy in the node the job in service resides in. I.e., if it is optimal to abort in state (i, k), then it is optimal as well to abort in state $(i, k + 1)$. This result is a corollary of the following proposition; cf. Proposition 4.4 with $r(k + 1) = r(k) = 0$.

Proposition 4.6 *For all $n \geq 0$, $i \geq 0$ and $k \geq 0$,*

$$V_n(i, k+1) - V_n(i, k) \leq 0.$$

Proof. We confine ourself to a sketch of the proof. We first observe that idling, if it were allowed, will never be more advantageous than continuing. This can be shown by means of coupling and a sample path argument. The same technique can then be used to establish Proposition 4.6. Consider two n-period instances of our model, instance $\mathcal{I}_0$ starting in state (i, k) and instance $\mathcal{I}_1$ starting in state $(i, k+1)$. If it is optimal to abort in instance $\mathcal{I}_1$, then the result is immediate. As long as it is optimal to continue service in instance $\mathcal{I}_1$, let instance $\mathcal{I}_0$ copy the admission policy employed by instance $\mathcal{I}_1$. In addition, instance $\mathcal{I}_0$ continues service as long as instance $\mathcal{I}_1$ continues service, unless the job in service in instance $\mathcal{I}_0$ has already been served successfully, in which case instance $\mathcal{I}_0$ idles. This idling policy is maintained until instance $\mathcal{I}_1$ aborts service or until the job in service has been served successfully, whichever comes first.

□

4.5 Conclusions

We have derived generalized monotonicity and threshold results for several extensions of the M|E$_N$|1 model. Among these extensions were batch Poisson arrivals, as well as Erlang arrivals and the even more general case of phase-type arrivals. We showed that the optimal admission/termination policy remains monotonic in both the number of jobs in the system and the number of phases completed for the job in service, but that the optimal termination policy need not be monotonic in the arrival phase. The most important extension, however, was the incorporation of a more general feed-forward routing mechanism, a crucial part of which consisted of the formulation of the two regularity conditions this routing mechanism had to comply with in order to secure monotonicity of the value functions and hence monotonicity of the optimal policy. We concluded with a discussion of a discrete-time model with deterministic decision epochs and service times consisting of a sum of geometric phases. For this model, a characterization of the optimal policy was attained by means of combining the analysis of the M^X|E$_N$|1 model with that of the M|E$_{N,\,jumps}$|1 model.

A natural direction for further research would be to investigate even more general feed-forward routing mechanisms than that of the $M|E_{N,\,jumps}|1$ model, namely, ones that allow for so-called *tree-structured* routings and rewards. Such a routing mechanism is characterized by a *reward tree*, which shows all possible sequences of phases, termed *paths*, that can be passed through by a job entering service. See Figure 4.2 for an example of such a reward tree. Note that without loss of generality no two branches of the reward tree are interwoven.

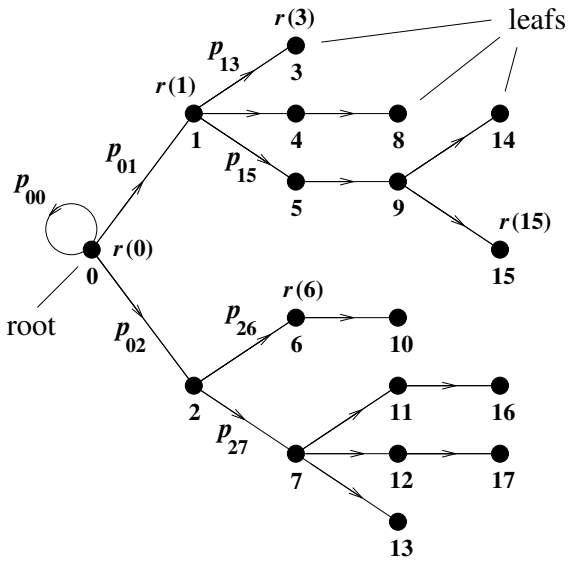

Figure 4.2: Tree-structured routing mechanism

Tree-structured routing mechanisms arise naturally in the following practical situation. Assume jobs are of different types, but the type remains unknown until the job has been fully investigated. For each job, the investigation consists of a number of phases. The distribution of the duration of a phase is independent of which type of job is served. Each time a phase is completed, we gain more insight into what kind of job we are dealing with. When the correct type has been discovered, a type-dependent reward is collected and the job is terminated. This can be modelled by assigning type-dependent rewards to the *leafs* of the tree, and a reward of 0 to all other nodes.

For general reward trees, one may attempt to formulate a set of monotonicity properties and investigate what regularity conditions on the rewards and the transition probabilities are required to assure these properties. In addition, one may investigate under what conditions a reward tree is *collapsable*. A reward tree is said to be *collapsable* if there is a one-to-one correspondence between its routing mechanism and some feed-forward routing mechanism that has the form of the routing mechanism of the $M|E_{N,\,jumps}|1$ model and that satisfies Fundamental Assumptions 4.1 and 4.2.

5

A multi-server extension of the $M|E_N|1$ model — *Computational issues and a near-optimal heuristic*

Administrative processes are seldom the responsibility of one sole employee. Far more commonly, departments and offices have several resources at their disposal and these resources share the complete workload the department or office is faced with. However, usually due to financial restrictions, it is rarely the case that the total resource capacity is sizeable enough to be capable of executing all work offered to the system. The lack of sufficient resource capacity, or alternatively stated, the superfluous supply of work, calls for intelligent decision support. In the single-server decision models we considered so far, this support consists of a control policy that prescribes at each moment in time whether to continue or abort the service of the job that is currently being served and whether to accept or reject any new jobs. Many multi-resource environments can reasonably be modelled by a set of single-server stations, where the separate servers are assumed to follow a policy which does not depend on actions taken by other servers. See, e.g., Example 4.1. The main advantage of such a system is its relatively low computational complexity, but the control will not be very advanced.

We can obtain a more sophisticated control by letting the resources act as a *pool*. This means that work is not allocated to specific resources in advance, but rather shared by the resources on a dynamic basis. If their joint capacity is insufficient to do all work offered to the system, then on-line decisions must be made as to when to accept an arriving job, when to abort jobs, and which

jobs to abort. These decisions will depend on the number of jobs already in the system and the progress of the respective servers as regards the service process. In this chapter, we will discuss this multi-server model, which is an extension of the M|E$_N$|1 model studied in Chapter 3. In particular, we will present a heuristic for the computation of a near-optimal policy for this multi-server model. The heuristic is based on a closely related model, namely, a slightly modified version of the M|E$_N$|1 model, whose optimal policy is readily computed. We will evaluate and refine the heuristic by means of a numerical study.

Section 5.1 provides a detailed model description. In Section 5.2, we delimit the action space of the multi-server model. In particular, we show that the action space is essentially the same as in the single-server model, i.e., equal to $\{0, 1, 2, \ldots, i\}$, where i is the number of jobs in the system. Next, in Section 5.3, we obtain some monotonicity results which apply specifically to the multi-server model. We show that, in general, the monotone structure of the optimal policy for the single-server model is not fully preserved if the model is extended to more than one server. This loss of structure is one of the reasons why the multi-server extension is analytically intractable.

In Section 5.4, we touch on important computational issues surrounding the multi-server model and its optimal control policy, and we advocate the need for a heuristic for this model. Section 5.5 discusses a model that is closely related to the multi-server model and which can be used as a basis for a natural heuristic. The heuristic itself is described and refined in Section 5.6. A part of the construction of the heuristic evolved from a numerical study. We present the results of this study in Section 5.7. These results indicate that the heuristic yields near-optimal performance for systems with a modest number of servers. Finally, in Section 5.8, using simulation, we assess the performance of the heuristic for larger systems, by comparing it to another natural heuristic.

5.1 Model description

We consider the following natural multi-server extension of the M|E$_N$|1 model studied in Chapter 3. Depart from the M|E$_N$|1 model, but let the system consist of s mutually indistinguishable servers in parallel, instead of only a single server. The system has a joint buffer, which is infinitely large, and the servers operate according to the FCFS discipline. Each job arriving at or residing in the system is served by at most one server. The system

features the same events, costs and rewards, and admission and termination control as the $M|E_N|1$ model, including Fundamental Assumption 3.1. So, in principle, the service requirements of a job comprise N exponential phases, but at any time it may be decided to abort the service of a job, which yields some reward $r(k)$ that depends on the number of phases k that have been completed for the job. The marginal returns are positive, but diminishing in k. See Figure 5.1 for a graphical representation of the queueing system captured by this $M|E_N|s$ model.

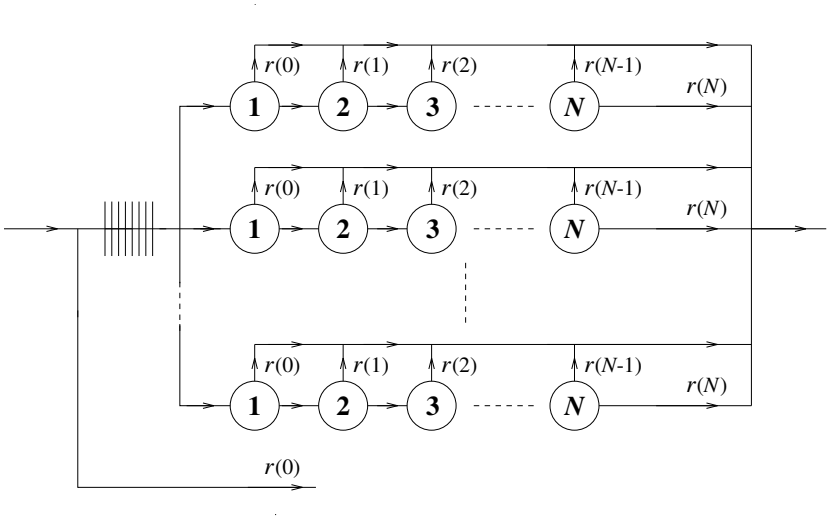

Figure 5.1: $M|E_N|s$ queue with admission and termination control

Using uniformization, we can consider that events occur at the jump times of a Poisson process with rate $\lambda + s\mu + \alpha$, where $\lambda \geq 0$, $\mu > 0$ and $\alpha \geq 0$. By scaling time, we take $\lambda + s\mu + \alpha = 1$ without loss of generality. Then, a transition concerns a service phase completion with probability $s\mu$, and a service phase completion at some specific server with probability μ.

5.1.1 State description

The state of the system can be fully described by the tuple $(i, \underline{s})$, where i is the number of jobs in the system and $\underline{s} := (s_0, \ldots, s_N)$, where s_k is the number of servers that have completed k phases for the job they are currently processing, $0 \leq k \leq N$. See also Figure 5.1. Vector $\underline{s}$ is termed

the server state vector. Since servers are indistinguishable, we do not need a full state description as regards which exact server is in which phase. Only the total number of servers that are currently in that phase is relevant. This already cuts down the total number of states considerably. If there is some upper bound on the number of jobs in the system, $i_{\max}$ say, then the total number of states drops already from $\mathcal{O}\left((N+1)^s \cdot i_{\max}\right)$ to $\mathcal{O}\left(\binom{N+s}{N} \cdot i_{\max}\right)$. If $h > 0$, then such an $i_{\max}$ is readily determined.

Note that $\sum_{k=0}^{N} s_k = \mathbf{min}\{i, s\}$. This is the number of busy servers, denoted by state variable s_B. At any time, the number of idle servers is $s_I := s - s_B$, and the number of queued jobs is $i_Q := i - s_B$.

5.1.2 Dynamic Programming formulation

We adopt the value functions $V_n(\cdot)$ and $V_n(\cdot, \text{arr})$ of the M|E$_N$|1 model and adapt them to the new state description, e.g., $V_n(i, \underline{s})$ denotes the maximum expected n-period α-discounted reward when the current state is $(i, \underline{s})$. For $0 \leq k \leq N$, define $\underline{e}_k := (\underbrace{0, 0, \ldots, 0}_{k}, \underbrace{1, 0, \ldots, 0}_{N+1-k})$. Then, in any state $(i, \underline{s})$, a decision $\pi \in \{\underline{a} \mid \underline{a} \leq \underline{s} + i_Q \cdot \underline{e}_0\}$ is to be taken as regards the termination of any jobs. In this notation, $\underline{a} := (a_0, \ldots, a_N)$, where, for $0 < k \leq N$, a_k is the number of servers that have completed k phases for the job they are currently processing and that now abort the service of that job. Note that a_0 is a special case. This is the number of servers that have not yet completed any phases for the job they are currently processing and that now abort the service of that job, *plus* the number of queued jobs being terminated. By definition, the termination of a queued job is indistinguishable from the termination of a job which is in its first service phase. Therefore, there is no reason to discriminate between these two types of jobs when deciding which jobs to abort.

Further, in any state $(i, \underline{s}, \text{arr})$, first a decision $\pi \in \{\texttt{accept}, \texttt{reject}\}$ is to be taken as regards the arrival. Subsequently, it is decided which jobs to abort, according to some decision vector $\underline{a}$, as described above.

Define $a := \sum_{k=0}^{N} a_k$ and let $\mathbf{1}[\cdot]$ be the indicator function, as defined earlier in Section 2.2.1. Then our model is defined by the following set of DPEs, where, analogous to Chapter 3, we use the notation $V_n(i, \underline{s}; \texttt{co})$ to denote $V_n(i, \underline{s})$, *given* that decision `continue` is chosen in state $(i, \underline{s})$.

For $n \geq 1$, $i \geq 0$ and all $\underline{s}$ such that $\sum_{k=0}^{N} s_k = \mathbf{min}\{i, s\}$:

$$V_0(i, \underline{s}) \;=\; \sum_{k=0}^{N} s_k r(k) \;+\; (i - s_B) r(0)$$

$$V_n(i, \underline{s}) \;=\; \max_{\underline{a} \leq \underline{s} + i_Q \underline{e}_0} [V_n(i - a,\; \underline{s} - \underline{a} + \underline{e}_0 \cdot \mathbf{min}\{a, i_Q\};\; \mathtt{co}) + \sum_{k=0}^{N} a_k r(k)]$$

where

$$V_n(i, \underline{s}; \mathtt{co}) \;=\; \lambda V_{n-1}(i, \underline{s}, \mathrm{arr}) \;+\; \sum_{k=0}^{N-1} s_k \mu V_{n-1}(i,\; \underline{s} - \underline{e}_k + \underline{e}_{k+1}) \;+\; (s - \sum_{k=0}^{N-1} s_k) \mu V_{n-1}(i, \underline{s}) \;-\; ih$$

and

$$V_{n-1}(i, \underline{s}, \mathrm{arr}) \;=\; \mathbf{max}\{V_{n-1}(i+1,\; \underline{s} + \underline{e}_0 \cdot \mathbf{1}[i < s]) - c,\; V_{n-1}(i, \underline{s}) + r(0)\}$$

The first element of the new server state vector $\underline{s} - \underline{a} + \underline{e}_0 \cdot \mathbf{min}\{a, i_Q\}$ in the DPE for $V_n(i, \underline{s})$ was obtained as follows. We may write $a_0 = a_{S,0} + a_Q$, where the S refers to jobs that were in service, and the Q refers to jobs that were in the queue. Then, immediately after the collective abort operation, in principle, $\sum_{k=1}^{N} a_k + a_{S,0}$ jobs from the queue may enter service. However, if the remaining number of jobs in the queue, which is $i_Q - a_Q$, is less than that, then only those remaining jobs can enter service. Hence, the new number of jobs in node 0 will become $s_0 - a_{S,0} + \mathbf{min}\{\sum_{k=1}^{N} a_k + a_{S,0},\; i_Q - a_Q\}$, which is readily seen to be equal to $s_0 - a_0 + \mathbf{min}\{a, i_Q\}$.

5.2 Cutting down on the action space

The state and action spaces of the M|E$_N$|s model are large. In fact, there are $(s_0 + i_Q + 1) \prod_{k=1}^{N} (s_k + 1)$ decision vectors $\underline{a}$ that satisfy $\underline{a} \leq \underline{s} + i_Q \underline{e}_0$. However, we will show that many actions are surely sub-optimal and can

therefore be omitted from the model. In particular, the following proposition states that when deciding which jobs to abort, it is optimal to start at the rear of $\underline{s}$, i.e., at those servers that have completed the most phases with respect to the job they are currently serving.

Proposition 5.1 *For any decision vector $\underline{a} \leq \underline{s} + i_Q \underline{e}_0$, there is a decision vector $\underline{a}' \leq \underline{s} + i_Q \underline{e}_0$ with $\sum_{l=0}^{N} a_l = \sum_{l=0}^{N} a'_l$ that satisfies the condition $a'_{k-1} > 0 \Rightarrow a'_k = s_k$ for $0 < k \leq N$ and that is at least as profitable as $\underline{a}$.*

From a numerical point of view, the important implication of Proposition 5.1 is that for any given state, the size of the action space can essentially be reduced from $(s_0 + i_Q + 1) \prod_{k=1}^{N} (s_k + 1)$ to just $i + 1$. For example, if $N = 6$, $s = 10$, $\underline{s} = (1, 1, 2, 3, 1, 2, 0)$ and $i_Q = 0$, then we only need to consider 11 actions instead of the original 288.

In order to prove Proposition 5.1 it suffices to establish the following result.

Proposition 5.2 *Let $i \geq 2$ and let $\underline{s}$ be such that at least one job, $\mathcal{J}_1$ say, is in some node k and at least one job, $\mathcal{J}_2$ say, is in some subsequent node $l > k$. The decision to abort the service of $\mathcal{J}_2$ and continue the service of $\mathcal{J}_1$ is at least as profitable as the decision to abort the service of $\mathcal{J}_1$ and continue the service of $\mathcal{J}_2$.*

In other words, Proposition 5.2 states that if it is optimal for a server whose job is in node k to abort service, then this decision is optimal as well for a server whose job is in a subsequent node $l > k$.

Proof. Let the current number of jobs be $i \geq 2$. For $m \geq 0$, let m^- be defined as in Chapter 3, i.e., equal to $\mathbf{min}\{m, N\}$. Consider an n-period process instance $\mathcal{I}_0$ of our model. Let $\mathcal{I}_0$ be such that at least one server is serving a job in node k and at least one server is serving a job in node $l > k$.

Next, we introduce process instances $\mathcal{I}_1$ and $\mathcal{I}_2$. Let $\mathcal{I}_1$ denote $\mathcal{I}_0$, given initial feasible action $\underline{a}' := (a_0, a_1, \ldots, a_k, \ldots, a_l, \ldots, a_N)$, where $a_k > 0$ and $a_l < s_l$. Let $\mathcal{I}_2$ denote $\mathcal{I}_0$, given initial action $\underline{a}'' := \underline{a}' - \underline{e}_k + \underline{e}_l$. Using a sample path argument, we show that $\mathcal{I}_2$ yields at least the same reward as $\mathcal{I}_1$.

First, note that the initial actions $\underline{a}'$ and $\underline{a}''$ yield direct rewards of $\sum_{j=0}^{N} a_j r(j)$ and $\sum_{j=0}^{N} a_j r(j) + r(l) - r(k)$, respectively. Furthermore, after its initial action $\underline{a}'$, instance $\mathcal{I}_1$ makes a transition to state $(i - a, \underline{s})$ for some

$\underline{s}$, whereas instance $\mathcal{I}_2$ makes a transition to state $(i-a, \underline{s}+\underline{e}_k-\underline{e}_l)$ after its initial action $\underline{a}''$. It remains to show that

$$V_n(i-a,\underline{s})+\sum_{j=0}^{N} a_j r(j) \;\le\; V_n(i-a,\underline{s}+\underline{e}_k-\underline{e}_l)+\sum_{j=0}^{N} a_j r(j)+r(l)-r(k),$$

i.e.,

$$V_n(i-a,\underline{s})-V_n(i-a,\underline{s}+\underline{e}_k-\underline{e}_l) \;\le\; r(l)-r(k). \tag{5.1}$$

Given the new states of $\mathcal{I}_1$ and $\mathcal{I}_2$, we couple all events (all arrival and service phase completion events and, finally, the event that the process ends) and all decisions at arrival and service phase completion times. To be precise, once both processes have carried through their respective initial action, instance $\mathcal{I}_2$ copies the optimal decisions taken in instance $\mathcal{I}_1$. This is feasible, because $\mathcal{I}_1$ and $\mathcal{I}_2$ feature the same number of jobs in the system and the same remaining number of periods. Since servers are indistinguishable, we may assume without loss of generality that immediately after the initial action, one particular server, denoted S, is serving a job in node k in $\mathcal{I}_2$ and a job in node l in $\mathcal{I}_1$. The distribution of the other $s-1$ servers over the $N+1$ nodes is the same for $\mathcal{I}_1$ and $\mathcal{I}_2$. Let Ω denote the set of these $s-1$ servers.

Then, for both processes, the costs of continuing service are identical, and so are the costs and rewards resulting from the admission and rejection, respectively, of new jobs. Any server from Ω that aborts service also yields the same direct reward in both processes, and this clearly also holds for the termination of any queued jobs.

Assume that server S aborts service at some point in time, possibly because time hits zero. (The alternative is that the system already vanished before then, in which case the difference in reward between the two instances is zero, which is at most $r(l)-r(k)$.) With respect to server S this leaves us with direct job rewards $r((m+l-k)^-)$ and $r(m)$ for $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively, for some $k \le m \le N$. Since $r(j)$ is concave in j,

$$r((m+l-k)^-)-r(m) \;\le\; r(l)-r(k). \tag{5.2}$$

In addition, after this abort operation, both processes become and remain identical. Together with (5.2), this assures that inequality (5.1) is satisfied.

□

Consequently, from now on we only have to consider decision vectors $\underline{a}$ that satisfy the condition $a_{k-1} > 0 \Rightarrow a_k = s_k$ for $0 < k \leq N$.

Let $(i, \underline{s}^{-a})$ denote the state immediately after the decision to abort a jobs in state $(i, \underline{s})$, where $0 \leq a \leq i$. Note that $\underline{s}^{-(a+b)} = (\underline{s}^{-a})^{-b}$ for $0 \leq a + b \leq i$. Using Proposition 5.1, one may verify that

$$\underline{s}^{-a} \;=\; \underline{s} - \sum_{k=0}^{N} \underline{e}_k \cdot \mathbf{min} \left\{ (\underline{s} + i_Q \underline{e}_0)_k \,, \left(a - \sum_{j=k+1}^{N} s_j \right)^+ \right\} + \underline{e}_0 \cdot \mathbf{min}\{a, i_Q\},$$

where

$$x^+ := \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{else.} \end{cases}$$

Similarly, we introduce the notation $r^a(\underline{s})$, which denotes the total direct reward corresponding to decision a in state $(i, \underline{s})$, where $0 \leq a \leq i$. Note that $r^{a+b}(\underline{s}) = r^a(\underline{s}) + r^b(\underline{s}^{-a})$ for $0 \leq a + b \leq i$. One may verify that

$$r^a(\underline{s}) \;=\; \sum_{k=0}^{N} \mathbf{min} \left\{ (\underline{s} + i_Q \underline{e}_0)_k \,, \left(a - \sum_{j=k+1}^{N} s_j \right)^+ \right\} \cdot r(k).$$

Now, we can replace the original DPE for $V_n(i, \underline{s})$ by

$$\left| \; V_n(i, \underline{s}) \;=\; \max_{a \leq i} \; V_n(i - a, \; \underline{s}^{-a}; \; \mathtt{co}) \;+\; r^a(\underline{s}) \right.$$

5.3 Properties of the optimal policy

In this section, under two natural regularity conditions, we derive two basic properties of the optimal admission/termination policy for the multi-server model. Further, we touch upon the complexity of the model by showing that, in general, the model does not feature monotonicity in the server state vector $\underline{s}$. Finally, we discuss briefly the phenomenon that it can be optimal to abort more than one job at a time in the infinite horizon case.

5.3.1 Non-termination for $i \leq s$

We show that under the following condition on the costs and rewards, it will surely be optimal not to abort any jobs as long as the queue is empty and none of the jobs in the system has received full service. This applies to both the finite horizon problem and the infinite horizon problem with either the average reward criterion or the criterion of total discounted reward.

Assumption 5.1 Let

$$\mu[r(N) - r(N-1)] \;\geq\; h + \alpha r(N).$$

Proposition 5.3 *If $n \geq 1$, $1 \leq i \leq s$ and $\underline{s}$ is such that $s_N = 0$, then it is optimal not to abort any jobs in state $(i, \underline{s})$.*

Proof. Let $n \geq 1$, $1 \leq i \leq s$ and let $\underline{s}$ be such that $s_N = 0$. We show that serving some job for one period and then aborting it gives a non-negative additional reward, as compared to aborting the job right now. Consider some job in service, $\mathcal{J}$ say, and suppose it currently resides in node k. Termination of the service of job $\mathcal{J}$ yields a direct reward of $r(k)$. Further, in the oncoming period, the holding costs incurred by the system will be jh, for some $j < i$. Alternatively, if the service of job $\mathcal{J}$ is continued, then the holding costs will be $(j+1)h$ in the oncoming period. After this period, we abort the service of job $\mathcal{J}$ after all. With probability μ, job $\mathcal{J}$ has completed its current service phase, and now resides in node $k+1$. With probability α, the process has vanished. With probability $1-\mu-\alpha$, job $\mathcal{J}$ remains in node k. This means that the expected reward for job $\mathcal{J}$ is $\mu r(k+1)+(1-\mu-\alpha)r(k)$. Hence, the expected difference in reward between the decision to serve job $\mathcal{J}$ for exactly one more period and the decision to abort it right away is equal to

$$\mu r(k+1) + (1-\mu-\alpha)r(k) - (j+1)h \;-\; [r(k) - jh],$$

which is at least zero, because of Assumption 5.1 and the assumption that $r(\cdot)$ is non-decreasing and concave.

□

Remark 5.1 If Assumption 5.1 is not satisfied, then we may remove a sufficient number of service phases from the initial workload of a job, from back to front, starting with phase N, until the condition is met. These are service phases that will never be reached by a job. Hence, the condition can be assumed to hold without loss of generality.

Remark 5.2 If there is no discounting, i.e., $\alpha = 0$, then Assumption 5.1 and Fundamental Assumption 3.1 imply that for all $0 \leq k < l \leq N$,

$$r(l) - r(k) \ \geq \ (l-k)\mu^{-1}h. \tag{5.3}$$

5.3.2 Admission for $i < s$

We show that under the following condition on the costs and rewards, it will surely be optimal in the infinite horizon problem with average reward criterion to accept arriving jobs as long as there is at least one idle server.

Assumption 5.2 Let

$$r(N) - r(0) \ \geq \ N\mu^{-1}h + c.$$

Proposition 5.4 *It is optimal in the infinite horizon problem with average reward criterion to accept a new job as long as $i < s$.*

Proof. Without loss of generality, denote the servers by $1, 2, \ldots, s$ and always send an accepted job to the idle server with the lowest index. Now, suppose there is an arrival to the system at some point T_0 in time and suppose there are $s-1$ servers busy at time T_0. Let strategy π reject the new job and let strategy π' accept it. We couple all events and decisions with respect to servers 1 to $s-1$, where π' copies the actions taken by π. As far as server s is concerned, let π' continue service until either (i) π aborts a job (which arrived and was directed to server s at some time $T_1 > T_0$) at server s, or (ii) all N service phases have been completed. Let $\mathcal{I}(\pi)$ and $\mathcal{I}(\pi')$ denote the processes governed by π and π', respectively.

If π never directs a job to server s in the future (hence, T_1 does not exist), then the expected reward earned by π' with respect to server s before $\mathcal{I}(\pi)$ and $\mathcal{I}(\pi')$ become identical is $r(N) - N\mu^{-1}h - c$. For π, there is only the initial reward $r(0)$ for rejecting at time T_0. It is then immediate from Assumption 5.2 that π' is at least as profitable as π.

Alternatively, assume that T_1 exists. With respect to the execution of π', and for $t \geq T_1$, we pretend the state (i.e., the number of completed service phases) of server s at time t to be the one at time $t - (T_1 - T_0)$. Next, with respect to server s, we couple the state of the service process in $\mathcal{I}(\pi)$ at time t to the state of the service process in $\mathcal{I}(\pi')$ at time $t - (T_1 - T_0)$. So π' can copy π and take exactly the same actions, apart from the initial action

at time T_1, which is to reject the new job. The moment π aborts the job at server s, at time T_2 say, strategy π' does the same. If we discard for the time being the initial rewards $r(0)$ for rejecting jobs, then, as far as server s is concerned, (i) the reward earned by π during the time interval $[T_1, T_2]$ equals the reward earned by π' during $[T_0, T_0+T_2-T_1]$, (ii) the reward earned by π during the time interval $[T_0, T_1)$ is zero, and (iii) the expected reward earned by π' during $(T_0+T_2-T_1, T_2]$ is at least zero by (5.3). Hence, noting that π and π' reject the same number of jobs during $[T_0, T_2]$, the expected reward earned by π' in the time interval $[T_0, T_2]$ is at least as high as the expected reward earned by π during that interval. Since the two processes become identical at time T_2, it follows that π' is at least as profitable as π from T_0 onward.

□

5.3.3 Monotonicity and loss of monotonicity for $i \geq s$

We show that, in general, the monotone structure of the optimal policy for the single-server model does not extend in full to the multi-server model. For this purpose, we first give two preliminary definitions.

First, for each state $(i, \underline{s})$ with $i \geq s$, there is a unique position k, $0 \leq k \leq N$, such that $s_k > 0$ and $s_l = 0$ for $k < l \leq N$. Denote this unique position by $\ell(\underline{s})$, where the symbol ℓ stands for 'last'.

Second, for two server state vectors $\underline{s}$ and $\underline{t}$, we say that $\underline{t} \succeq \underline{s}$ if

$$\underline{t} \;=\; \sum_{k=0}^{N-1} \left(\underline{s} - \sigma_k \underline{e}_k + \sigma_k \underline{e}_{k+1}\right)$$

for some $\sigma_0, \ldots, \sigma_{N-1} \geq 0$ such that $t_k \geq 0$ for all $0 \leq k < N$. For example, $(2,1,1,0) \preceq (0,3,1,0) \preceq (0,1,2,1)$ in some model with $s = 4$ and $N = 3$.

For the single-server model, we obtained the following result: if it is optimal to abort the service of a job in state (i,k), then it is optimal as well to abort service in all states (i,l) with $l \geq k$. One could conjecture that in the multi-server model the following generalization holds: for $i \geq s$, if it is optimal to abort at least one job in state $(i+1, \underline{s})$, then it is optimal as well to abort at least one job in all states $(i+1, \underline{t})$ with $\underline{t} \succeq \underline{s}$. However, the following two counterexamples show that in general this is not true in neither the finite horizon case nor the infinite horizon case with average reward criterion.

Counterexample 5.1 Consider the following two-server instance of our model: $\mu = \frac{1}{2}$ (so $\lambda = \alpha = 0$), $h = 5$, $N = 4$ and $r(k) = 25k$ for $0 \leq k \leq 4$.

Let $\underline{s} := (2, 0, 0, 0, 0)$ and $\underline{t} := (0, 0, 0, 2, 0)$. Note that $\underline{t} \succeq \underline{s}$.

Then,

$$\begin{aligned} V_3(3, \underline{s}) &= \max_{a \leq 3} V_3(3 - a,\ \underline{s}^{-a};\ \mathtt{co}) + r^a(\underline{s}) \\ &= \mathbf{max}\{0 : 40,\ 1 : 45,\ 2 : 22\tfrac{1}{2},\ 3 : 0\} \ = \ 45, \end{aligned}$$

so $a = 1$ is optimal, i.e., it is optimal to abort exactly 1 job in state $(3, \underline{s})$ if the remaining number of periods is 3.

However,

$$\begin{aligned} V_3(3, \underline{t}) &= \max_{a \leq 3} V_3(3 - a,\ \underline{t}^{-a};\ \mathtt{co}) + r^a(\underline{t}) \\ &= \mathbf{max}\{0 : 186\tfrac{1}{4},\ 1 : 185\tfrac{5}{8},\ 2 : 172\tfrac{1}{2},\ 3 : 150\} \ = \ 186\tfrac{1}{4}, \end{aligned}$$

so $a = 0$ is optimal, i.e., it is optimal not to abort any jobs in state $(3, \underline{t})$ if the remaining number of periods is 3.

The reasoning behind this phenomenon lies in the trade-off between holding costs on the one hand and the desire to prevent undesirable idle time on the other. Apparently, in state $(i, \underline{t})$, the reduction in holding costs resulting from the termination of one of the two jobs in service is outweighed by the probability of having to witness idle time at one of the servers. In state $(i, \underline{s})$, this is apparently the other way around.

Counterexample 5.2 Consider the following two-server instance of our model: $\lambda = \frac{1}{21}$, $\mu = \frac{10}{21}$, $h = \frac{40}{21}$, $c = 0$, $N = 4$ and $r(k) = 25k$ for $0 \leq k \leq 4$.

Let $\underline{s} = (1, 0, 1, 0, 0)$ and $\underline{t} = (0, 1, 0, 1, 0)$, so $\underline{t} \succeq \underline{s}$. Consider the infinite horizon average reward criterion. Computations reveal that both $(10, \underline{s})$ and $(10, \underline{t})$ are recurrent states and that it is optimal to abort in state $(10, \underline{s})$, but to continue service in state $(10, \underline{t})$.

The monotonicity result for the single-server model leans on the fact that in the single-server model an abort operation in state (i, k) and an abort operation in state (i, l) lead to the same state, namely $(i - 1, 0)$. In general, this property is lost in the multi-server model. E.g., if we consider $\underline{s}$ and $\underline{t}$ with $\underline{t} \succeq \underline{s}$ and $\underline{t} \neq \underline{s}$, then in general, $(i - 1, \underline{s}^{-1})$ does not equal $(i - 1, \underline{t}^{-1})$.

There is exactly one case in which these two states are indeed identical, namely if $\ell(\underline{s}) < N$ and $\underline{t} = \underline{s} - \underline{e}_{\ell(\underline{s})} + \underline{e}_{\ell(\underline{s})+1}$. The following proposition states that in this specific case we have the same monotonicity as in the single-server model.

Proposition 5.5 *Let $i \geq s$, $\ell(\underline{s}) < N$ and $\underline{t} = \underline{s} - \underline{e}_{\ell(\underline{s})} + \underline{e}_{\ell(\underline{s})+1}$, so $(i-1, \underline{s}^{-1}) = (i-1, \underline{t}^{-1})$. If it is optimal to abort at least one job in state $(i+1, \underline{s})$, then it is optimal as well to abort at least one job in state $(i+1, \underline{t})$.*

Proof. Suppose it is optimal to abort at least one job in $(i+1, \underline{s})$ and suppose it is strictly optimal not to abort any jobs in $(i+1, \underline{t})$. Then $V_n(i, \underline{s}^{-1}) + r^1(\underline{s}) = V_n(i+1, \underline{s})$ and $V_n(i, \underline{t}^{-1}) + r^1(\underline{t}) < V_n(i+1, \underline{t})$. Together with $(i, \underline{s}^{-1}) = (i, \underline{t}^{-1})$, $r^1(\underline{s}) = r(\ell(\underline{s}))$ and $r^1(\underline{t}) = r(\ell(\underline{s})+1)$, this yields $V_n(i, \underline{t}) - V_n(i, \underline{s}) > r(\ell(\underline{s})+1) - r(\ell(\underline{s}))$, contradicting (5.1).

□

5.3.4 Multiple aborts in the infinite horizon case

One might surmise that in the infinite horizon case, it will be optimal to abort at most one job at a time in recurrent states. However, the following counterexample invalidates this assertion.

Counterexample 5.3 Consider the following three-server instance of our model: $\lambda = \frac{1}{3}$, $\mu = \frac{2}{9}$, $h = \frac{4}{9}$, $c = 10$, $N = 4$ and $\underline{r} = (0, 25, 45, 60, 72)$.

Let $\underline{s} = (1, 0, 0, 2, 0)$. In state $(4, \underline{s}, \text{arr})$, which is recurrent and which can be reached via, e.g., state $(3, \underline{s})$ and two consecutive arrivals before a service phase completion, it is optimal to first accept the new job, which causes the system to enter state $(5, \underline{s})$, and then to abort *two* jobs.

Counterexample 5.3 observes the system at an arrival time. Just before the corresponding arrival, there are 4 jobs in the system and this number drops to 3 immediately after the abort operation. Hence, the 'net decrease' is 1 and one might conjecture that, in general, the net decrease will never be larger than 1. However, a net decrease of more than one job can also be accomplished. This is illustrated by the following counterexample.

Counterexample 5.4 Consider the instance of Counterexample 5.3, but let there be 6 servers instead of 3. Adjusting the uniformization accordingly, we obtain $\lambda = \frac{1}{5}$ and $\mu = \frac{2}{15}$. Further, let $h = \frac{4}{15}$.

Let $\underline{s} = (3, 0, 0, 3, 0)$. In state $(9, \underline{s}, \text{arr})$, which is recurrent and which can be reached via, e.g., state $(6, \underline{s})$ and four consecutive arrivals before a service phase completion, it is optimal to first accept the new job, which causes the system to enter state $(10, \underline{s})$, and then to abort *three* jobs. Note that the new state will be $(7, \underline{s}^{-3})$, where $\underline{s}^{-3} = (6, 0, 0, 0, 0)$.

Counterexamples 5.3 and 5.4 have demonstrated that it can be optimal to trade a number of jobs which have some joint remaining workload for a fresh job which has a higher (remaining) workload. In retrospect, this is a natural observation.

The above observation applies specifically to arrival times of jobs. It does not say anything about service phase completion times. We make the following conjecture.

Conjecture 5.1 *In the infinite horizon case, it will be optimal in recurrent states to abort at most one job at service phase completion times.*

5.4 Optimal control versus heuristics

In this section, we indicate and illustrate the need for a good heuristic for multi-server workload models with controlled service times. Subsequently, we describe our heuristic and study its performance.

In theory, since we have derived a complete set of DPEs for our multi-server model, we can compute the optimal control policy for any instance. However, besides analytic intractability, the model suffers severely from computational intractability. This is not caused by the action space, which has been cut down in Section 5.2, but by the huge state space. The optimal control policy for the multi-server model is state-dependent. As argued in Section 5.1.1, we can assume that the number of states is of the order $\binom{N+s}{N} \cdot i_{\max}$, where $i_{\max}$ is some upper bound on the number of jobs in the system. This means that the number of states in an instance with $N = 8$ service phases, $s = 15$ servers and at most $i_{\max} = 20$ jobs in the system will be of the order 10^7. For $N = 15$, $s = 20$ and $i_{\max} = 30$, this will be a staggering 10^{11}.

Consequently, the amount of physical space required to store a complete multi-server instance grows extremely rapidly as either N or s increases. For example, consider an instance with $N = 6$, $s = 6$ and $i_{\max} = 12$. We needed 20MB of disk space to store all model data. The program we use to

do all the numerical work cannot handle an input file of such size.[6] Even if we were to use a more efficient method of encoding and decoding, then the storage limit would also be reached very soon. For example, using our encoding scheme, an instance with $N = 6$, $s = 10$ and $i_{\max} = 20$, which is still relatively small, would already require over 300MB of model data. This phenomenon—memory exhaustion due to high dimensionality of the state space—is known as the curse of dimensionality; see Bellman [7].

In addition, time will become a constraint. Our optimization program processes input files of up to about 10MB and for such file sizes it already takes several minutes to read the model data and compute the optimal policy. Computation times will be much longer if we use a program that does not first read all model data, but that generates specific data whenever it is needed.

Moreover, assuming the optimal policy can actually be computed, another problem arises. The output of the optimization program will consist of an exhaustive list of states and accompanying actions, which will be extremely hard to read. This is even worsened by the fact that, in general, the optimal policy does not feature a threshold structure in the server state vector, as illustrated in Section 5.3.3.

Finally, the optimal policy assumes that one has full knowledge of the state of the system, e.g., each server knows exactly which service phases other servers are in. In practice, this will very often not be the case. Even supervisors that manage the entire process may not have the disposal of a complete state description.

We conclude that the detailed optimal policy for the multi-server system will be impractical and undesirable. Practice demands practical decision support routines, i.e., sets of operational decision rules which are of an intuitive nature and which can be computed and implemented easily. We will describe a heuristic which provides a set of simple control rules which are adequate to obtain near-optimal performance. This set of control rules emerges from the optimal control policy for a single-server model closely related to the multi-server model. The control rules only demand that a server is aware of its own state and the size of the queue. A server need not be aware of or inquire for the state of other servers. The closely related model the heuristic is based on is in fact a slightly modified version of the

[6] We used `MDP`, developed at Eindhoven University of Technology.

M|E$_N$|1 model studied in Chapter 3. We discuss this model in the next section, as a preliminary to the description of the heuristic.

Subsequently, we will show by means of numerical results that for fairly small multi-server systems this simple control mechanism yields a near-optimal average reward per period, making the complicated state-dependent optimal control policy redundant.

Remark 5.3 We specifically aim at infinite horizon problems with average reward criterion, and at instances that satisfy Fundamental Assumption 3.1 and Assumptions 5.1 and 5.2, so in the remainder of this chapter, either these conditions are assumed to hold, or we make sure that they are met.

5.5 The M|E$_N^{\mu_i}$|1 model

Consider the M|E$_N$|1 model studied extensively in Chapter 3, but let the service rate be variable instead of constant. In particular, we consider a workload-dependent service rate of $\mu_i := \mathbf{min}\{i, s\}\mu$, where i is the number of jobs in the system and s is some fixed positive integer. We term this model the M|E$_N^{\mu_i}$|1 model. Note that the M|E$_N$|1 can be obtained from the M|E$_N^{\mu_i}$|1 model by choosing $s = 1$.

Conversely, the M|E$_N^{\mu_i}$|1 model can be obtained from the M|E$_N$|1 model by rewriting the DPEs for $V_n(i, k; \mathsf{co})$ as follows, where $n \geq 1$, $i \geq 0$ and $0 \leq k \leq N$. Note that the uniformization is as in the M|E$_N$|s model, i.e., $\lambda + s\mu + \alpha = 1$, where s is a new model parameter.

$$\begin{aligned} V_n(i,k;\mathsf{co}) \;=\; \lambda V_{n-1}(i,k,\mathrm{arr}) \;+\; \mu_i V_{n-1}(i,(k+1)^-) \;+ \\ (s-\mu_i)V_{n-1}(i,k) \;-\; ih \end{aligned}$$

In Chapter 4, we considered several extensions of the M|E$_N$|1 model, for which we derived generalized monotonicity and threshold results. For the M|E$_N^{\mu_i}$|1 model, however, it is impossible to establish such a generalization. The following two counterexamples demonstrate that both inequality (3.1) and inequality (3.2) need not hold, hence destroying the Key Proposition.

Counterexample 5.5 Consider the following instance of the M|E$_N^{\mu_i}$|1 model: $\mu_i = \mathbf{min}\{\frac{i}{3}, 1\}$ (so $\lambda = \alpha = 0$), $h = 0$, $N = 3$ and $\underline{r} = (0, 30, 57, 81)$.

Then,

$$V_1(2,2) - V_1(1,2) \;=\; \tfrac{2}{3}\cdot 81 + \tfrac{1}{3}\cdot 57 \;-\; (\tfrac{1}{3}\cdot 81 + \tfrac{2}{3}\cdot 57) \;=\; 8,$$

but

$$V_1(3,0) - V_1(2,0) \;=\; 30 \;-\; (\tfrac{2}{3}\cdot 30 + \tfrac{1}{3}\cdot\ 0) \;=\; 10,$$

which contradicts inequality (3.1).

Counterexample 5.6 Consider the same instance as in the previous counterexample, but let $N = 2$ and $\underline{r} = (0, 30, 57)$.

Then,

$$V_1(2,1) - V_1(1,1) \;=\; \tfrac{2}{3}\cdot 57 + \tfrac{1}{3}\cdot 30 \;-\; (\tfrac{1}{3}\cdot 57 + \tfrac{2}{3}\cdot 30) \;=\; 9,$$

but

$$V_1(2,0) - V_1(1,0) \;=\; \tfrac{2}{3}\cdot 30 + \tfrac{1}{3}\cdot\ 0 \;-\; (\tfrac{1}{3}\cdot 30 + \tfrac{2}{3}\cdot\ 0) \;=\; 10,$$

which contradicts inequality (3.2).

The fact that inequalities (3.1) and (3.2) have become invalid does not imply that the threshold characterization of the optimal admission/termination policy is lost as well. However, the following counterexample shows that, in general, the optimal admission policy is no longer monotone in the number of service phases completed for the job in service.

Counterexample 5.7 Consider the following instance of the $\mathrm{M}|\mathrm{E}_N^{\mu_i}|1$ model: $\mu_i = \mathbf{min}\{\frac{i}{2}, 1\}$, $h = 5$, $c = 4$, $N = 5$ and $\underline{r} = (0, 20, 35, 50, 65, 76)$.

Let there be a one-time arrival to the system, one period before the end of the process. Then,

$$V_1(1,0,\mathrm{arr};\mathtt{ac}) \;=\; \mathbf{max}\{10,5\} \;-\; 4 \;=\; 6 \;>\; 5 \;=\; V_1(1,0,\mathrm{arr};\mathtt{rj}),$$

$$V_1(1,3,\mathrm{arr};\mathtt{ac}) \;=\; \mathbf{max}\{55,55\} \;-\; 4 \;=\; 51 \;<\; 52\tfrac{1}{2} \;=\; V_1(1,3,\mathrm{arr};\mathtt{rj}),$$

$$V_1(1,4,\mathrm{arr};\mathtt{ac}) \;=\; \mathbf{max}\{66,70\} \;-\; 4 \;=\; 66 \;>\; 65\tfrac{1}{2} \;=\; V_1(1,4,\mathrm{arr};\mathtt{rj}),$$

so the optimal admission policy is not monotone in k.

The monotonicity of the optimal admission policy in the number of jobs in the system will probably still apply. Namely, the relative increase in the service rate when an extra job joins the system is higher when there are less jobs in the system, which adds even more to the value of an additional job when there are less jobs present.

Conjecture 5.2 *In the finite horizon case as well as the infinite horizon case, the optimal admission policy for the* $M|E_N^{\mu_i}|1$ *model is monotone in the number of jobs in the system.*

Further, with respect to the optimal termination policy, we can argue that Proposition 3.2 still holds. The proof given for the $M|E_N|1$ model remains valid for the $M|E_N^{\mu_i}|1$ model. Consequently, the optimal termination policy for the $M|E_N^{\mu_i}|1$ model is monotone in k. We conjecture that the optimal termination policy will be monotone in i as well, although we lack a cogent and insightful theoretical explanation for this assertion.

Conjecture 5.3 *In the finite horizon case as well as the infinite horizon case, the optimal termination policy for the* $M|E_N^{\mu_i}|1$ *model is monotone in the number of jobs in the system.*

Regarding the average reward criterion, the two conjectures are supported by our numerical study (the results of which are presented in Section 5.7). Without a single exception, all instances of the $M|E_N^{\mu_i}|1$ model we considered as part of this numerical study featured optimal admission and termination policies that were of a threshold type.

5.6 Description of the heuristic

In this section, we describe our heuristic for the multi-server model. This heuristic provides a rule of thumb which approximates the optimal infinite horizon control policy for the multi-server model with respect to the average reward yielded by the system per unit of time.

5.6.1 Basic form of the heuristic

We take the variable service rate model described and studied in Section 5.5 as point of departure. Our main idea is the following. If the service times of jobs were merely exponential instead of Erlang, and uncontrollable instead of

subject to control, then the systems described by the $\mathrm{M}|\mathrm{E}_N|s$ and $\mathrm{M}|\mathrm{E}_N^{\mu_i}|1$ models would coincide if the latter had a variable service rate of $\mathbf{min}\{i, s\}\mu$, where i is the number of jobs in the single-server system and s is the number of servers in the multi-server system, each server having service rate μ. In our case of Erlang distributed service times and admission as well as termination control, we can still use the $\mathrm{M}|\mathrm{E}_N^{\mu_i}|1$ model as an approximation for the $\mathrm{M}|\mathrm{E}_N|s$ model.

We base our heuristic on this idea. The optimal policy for the $\mathrm{M}|\mathrm{E}_N^{\mu_i}|1$ model is readily computed. For example, a fairly large single-server instance with $N = 10$, $s = 10$ and $i_{\max} = 20$ required only about 0.2MB of disk space and virtually no computing time. The corresponding multi-server instance would be unmanageable; cf. the discussion in Section 5.4. Once the optimal single-server policy has been determined, it can be used as a reference for the multi-server system, in effect for each of the servers separately.

Let $i^*_{\max}$ denote the maximum number of jobs admitted to the single-server system at any point in time, as indicated by the optimal policy for the $\mathrm{M}|\mathrm{E}_N^{\mu_i}|1$ model. Then, with respect to the multi-server system, for each $1 \leq i \leq i^*_{\max}$, this policy prescribes up to which service phase the decision will be to continue service and from which phase on the decision will be to abort. Note that it was shown in Section 5.5 that the optimal termination policy for the $\mathrm{M}|\mathrm{E}_N^{\mu_i}|1$ model is guaranteed to be monotone in k.

In that section, we also showed that the optimal admission policy for the $\mathrm{M}|\mathrm{E}_N^{\mu_i}|1$ model is not necessarily monotone in k. However, we can ensure a threshold form as follows: if it is optimal to accept a new job in some state (i, k), then we prescribe to accept in all states (i, l) with $l > k$ as well, regardless of whether this was also prescribed by the optimal single-server policy itself. We then obtain an admission policy for the multi-server system that prescribes for each $0 \leq i \leq i^*_{\max}$, and given that there is an arrival, up to which service phase the decision will be to reject the new job and from which phase on the decision will be to accept.

The scheme we have just described acts as a reference for each of the servers in the multi-server system. Once a particular server has completed a service phase or notices an arrival, it observes the number of service phases it has completed for the job in service, and decides whether to accept or reject the new job, if applicable, and whether to continue or abort service of the job in service. To be clear, servers are not aware of the total amount of work in the system, let alone the distribution of the workload over the various servers, but they do know the total number of jobs in the system.

In case of an arrival at the system, some servers may express the desire to accept the new job, whereas others may reject it. This calls for a higher-level decision, taken by some process supervisor, to determine whether the job is actually admitted to the system or not. Two natural higher-level admission policies are the following:

1. Admission policy Minor: accept if and only if at least one server wants to accept.
2. Admission policy Major: accept if and only if the majority of servers wants to accept; the new job will also be accepted if the ballot results in a tie.

Hence, the heuristic constitutes in fact two admission policies, one for each separate server and one for the process supervisors. We will include both higher-level admission policies in our numerical study, and compare their performance.

5.6.2 Refining the heuristic

In this section, we discuss a couple of natural adjustments we made to our heuristic to enhance its performance.

5.6.2.1 Non-termination for $i \leq s$ and admission for $i < s$

We know from Proposition 5.3 and Remark 5.3 that it will be optimal in the multi-server system not to abort any jobs as long as $i \leq s$ and no server has completed all N service phases with respect to the job it is currently serving. However, the optimal termination policy for the $\mathrm{M}|\mathrm{E}_N^{\mu_i}|1$ model may very well prescribe to abort the job in service in certain states (i, k) with $i \leq s$ and $k < N$.[7] We correct this in our heuristic for the multi-server system simply by instructing each server to continue service as long as the queue is empty and node N has not yet been reached by the job being served by that server. In the instances we considered in our numerical study, this modification of the heuristic improved the performance of the heuristic significantly.

Similarly, we know from Proposition 5.4 and Remark 5.3 that it will be optimal in the multi-server system to accept any new jobs as long as $i < s$. Hence, we adjust the heuristic such that new jobs are automatically admitted

[7] This is actually a common phenomenon if the arrival rate is high. As an illustration, we will highlight one occurrence in Section 5.7. See Example 5.2.

to the multi-server system as long as there is at least one idle server. We note, however, that we did not encounter any instances for which the optimal admission policy for the $M|E_N^{\mu_i}|1$ did not already accept any new jobs as long as $i < s$, hence the following conjecture.

Conjecture 5.4 *In the infinite horizon case with average reward criterion, it is optimal in the $M|E_N^{\mu_i}|1$ model to accept a new job if $i < s$.*

Remark 5.4 The prescribed and hence guaranteed admission for $i < s$ and non-termination for $i \leq s$ imply that the assumption that servers know the number of jobs in the system may be replaced by the assumption that servers know the size of the queue and, if the queue is empty, know whether there is at least one idle server.

5.6.2.2 Preventing multiple service aborts

Upon arrival of a job which will be accepted by the higher-level admission policy, the decision prescribed by our heuristic as regards the termination of any jobs may cause more than one server to abort service. This might, however, be very unfavourable for the system as a whole. Namely, it could drain the system far too heavily, e.g., from $i \geq s$ jobs in the system to some undesired level far below s. See Example 5.1. We correct this in our heuristic by prescribing that irrespective of how many servers request an abort, only one may actually abort service. This will be the one that has completed the most service phases with respect to the job it is serving. Again, in our instances, the performance of the heuristic improved significantly by this adjustment. Note that we have obtained the following form of higher-level termination control:

- make full use of capacity: if $i \leq s$ and $\underline{s}$ is such that $s_N = 0$, then no server may abort service;
- avoid unintended clearance: at any time, at most one server may abort service.

Example 5.1 Consider some recurrent state $(8, (1, 1, 4, 0))$ in a six-server instance of our multi-server model. Assume that the optimal single-server policy accepts in each node if there are 8 jobs in the system and aborts service in node 3 if there are 9 jobs in the system. Then an ignorant multi-server implementation would let all 4 servers in node 3 abort service, inducing a

transition to state $(5, (4, 1, 0, 0))$. Clearly, it will be better to let only 1 of the 4 servers abort service, resulting in a transition to state $(8, (2, 1, 3, 0))$.

Remark 5.5 In Conjecture 5.1, we asserted that the optimal multi-server policy, which has complete information on the state of the system, will abort at most one job at service phase completion times. But we have also demonstrated that it may very well perform multiple service aborts at arrival times; cf. Counterexamples 5.3 and 5.4. Our heuristic, which always aborts at most one job at a time, is set against this policy.

5.6.3 Final form of the heuristic

We term the heuristic we have just described and refined the VSR ('variable service rate') heuristic. Its final form is summarized below.

The VSR *heuristic*

1. *Let be given an instance of the* $M|E_N|s$ *model.*
2. *Consider the corresponding instance of the* $M|E_N^{\mu_i}|1$ *model.*
3. *Compute its optimal admission/termination control policy.*
4. *If not yet prescribed as such, fix the admission policy such that decision* `accept` *is taken at arrival times as long as* $i < s$*. For* $i \leq s$*, if prescribed otherwise, fix the termination policy such that decision* `continue` *is taken as long as node* N *has not yet been reached.*
5. *Moreover, if not yet prescribed as such, fix the admission policy for each* $s \leq i \leq i^*_{\max}$ *such that if decision* `accept` *is taken in state* (i, k)*, then decision* `accept` *is also taken in all states* (i, l) *with* $k < l \leq N$*.*
6. *Translate this single-server policy to control rules for the multi-server system:*
   i. *In principle, each server in the multi-server system follows the optimal policy obtained in Steps 3 to 5.*
   ii. *Apply higher-level admission control: at arrival times, the decision to accept or reject depends on the number of servers that express their desire to accept. Under admission policy* Minor*, a new job is admitted if at least one server wants to accept. Under*

admission policy Major*, a new job is admitted if at least half of all servers want to accept.*

iii. Apply higher-level termination control: if, after the admission of a new job, more than one server wants to abort, then only the server that has completed the most service phases with respect to the job it is serving is granted permission to do so.

5.7 Numerical results

We have evaluated the VSR heuristic for a large collection of test instances of the multi-server model. In this section, we present the results we obtained for three representative sample sets of instances. These results are characteristic of the performance of the heuristic in case of a modest number of servers.

The remainder of the section is organized as follows. Section 5.7.1 contains some preliminaries. Sections 5.7.2 through 5.7.4 list the results for various instances of the model. The conclusions that can be drawn from the collected results are compiled in Section 5.7.5.

5.7.1 Some general remarks

We will consider consecutively (i) a 3-server system with at most 4 service phases per job, (ii) a 6-server system with at most 4 service phases per job, and (iii) a 4-server system with at most 6 service phases per job. For each system, we will consider 3 different cost structures. Further, for each system and cost structure, we will consider a range of values of the *gross workload*.

Following Section 1.2.2, the gross workload ρ is defined as

$$\rho := \lambda \frac{N}{s\mu}.$$

It is the utilization of the system if each job was to receive full service. Interesting values of ρ are those for which we cannot do *all* work, but still a sizeable fraction, e.g., values of ρ that lie somewhere between 1 and 2. The service rate of the servers is considered fixed and equal to 1, i.e., $\mu_{\text{init}} = 1$, where μ_{init} denotes the service rate *before* uniformization. Hence, different values of ρ are obtained solely by varying the arrival rate of jobs.

For $\rho \to \infty$, where N and s are fixed, it will be optimal for each server to accept a new job if the job it is serving has already passed at least $k^{\text{opt}} \in \mathcal{K}$ service phases, where

$$\mathcal{K} := \arg\max_k \frac{r(k) - r(0) - c}{k}.$$

The job in service is then terminated immediately, and replaced by the new job. Note that this policy can be carried out for sure if higher-level admission policy Minor is employed. In case of higher-level admission policy Major, the new job will only be admitted to the system if at least $\lceil \frac{s}{2} \rceil - 1$ of the other servers accept the new job as well. So, for very high values of ρ, one may expect that under higher-level admission policy Major, servers will often be a nuisance to each other, which will probably reduce the performance of the heuristic drastically. As we will see later on, this is indeed the case.

Denote by k^* the smallest element of $\mathcal{K}$ and define

$$\rho_{k^*} := \lambda \frac{k^*}{s\mu}.$$

This would be the workload of the system if *all* jobs were to receive exactly k^* service phases' worth of attention. In our decision model, the average number of service phases that jobs admitted to the system will pass through will gradually drop from N to k^* as the ratio $\frac{\lambda}{s\mu}$ increases. For larger values of this ratio, ρ_{k^*} will give a more meaningful representation of the workload of the system than ρ. For each system and cost structure considered, we will include the value of ρ for which $\rho_{k^*} = 1$, as well as at least two values around that particular value.

Furthermore, we would like to note that the refinement of the heuristic as discussed in Section 5.6.2 resulted in an improvement of up to 3.5% in absolute value and up to 95% in relative value with respect to the average reward per unit of time, as compared to the situation in which the measures described there are not applied.

Finally, a sensitivity analysis indicated that small changes to the policy (as computed by the heuristic) can influence the performance considerably. In our test instances, artificial disturbances led to deviations of up to about 8% in absolute value.

5.7.2 A 3-server system with at most 4 service phases per job

We consider a system with $s = 3$ servers, $N = 4$ service phases and reward function $\underline{r} = (0, 25, 45, 60, 72)$. We consider three sets of instances:

Set 1 $h_{\text{init}} = 2$ and $c = 10$.

Set 2 $h_{\text{init}} = 1$ and $c = 20$.

Set 3 $h_{\text{init}} = 5$ and $c = 0$.

Here, h_{init} are the holding costs *before* uniformization. Note that these are independent of the arrival rate of jobs. The uniformized holding costs h are equal to $h_{\text{init}}(\lambda_{\text{init}}+s)^{-1}$, where λ_{init} is the arrival rate before uniformization.

Within each set, we consider various values of λ_{init}, and hence ρ, and evaluate the heuristic under both admission policy Minor and admission policy Major. Note that Minor and Major will be identical and hence yield the same results for any instance in Set 3, since $c = 0$ implies that jobs will never be rejected; cf. Remark 3.4.

5.7.2.1 Results for Set 1

The results for Set 1 are listed in Table 5.1, where, as well as in the remainder of this chapter, g^* denotes the optimal average reward per unit of time for the multi-server system and g_{Minor} (g_{Major}, respectively) denotes the average reward per unit of time as obtained by the VSR heuristic under admission policy Minor (Major, respectively). The corresponding relative errors are denoted by Δ_{Minor} and Δ_{Major}, e.g.,

$$\Delta_{\mathsf{Minor}} := \frac{g_{\mathsf{Minor}} - g^*}{g^*} \cdot 100\%.$$

The absolute and relative accuracy in all calculations performed in our numerical study was 10^{-4}.

Additionally, to give an impression of maximum queue sizes, $i_{\max}^{\mathsf{Minor}}$ denotes the maximum number of jobs admitted to the multi-server system under the VSR heuristic and admission policy Minor. This number is readily obtained from the optimal single-server policy computed in Step 3 of the heuristic.

Further, note that for $\rho \to \infty$, it will be optimal for each server to accept a new job as soon as the job currently in service has passed 2 service phases,

because $\max_k (r(k) - 10)/k = 17.5$, which is attained for $k^* = 2$. We have

$$g^* \to \left(\frac{r(k^*) - r(0) - c}{k^*} - h_{\text{init}} \right) s \quad \text{as} \quad \rho \to \infty,$$

which we can use to verify our implementation of the multi-server model. We should have $g^* \to (17.5 - 2) \cdot 3 = 46.5$ as $\rho \to \infty$. In addition, note that $\rho = 2.000$ corresponds to $\rho_{k^*} = 1.000$.

ρ	g^*	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	Δ_{Minor}	g_{Major}	Δ_{Major}
0.333	13.44	20	13.44	−0.0%	13.44	−0.0%
0.667	25.88	13	25.88	−0.0%	25.88	−0.0%
1.000	34.06	9	34.04	−0.1%	34.04	−0.1%
1.333	38.22	7	38.14	−0.2%	38.14	−0.2%
1.667	40.54	6	40.49	−0.1%	40.49	−0.1%
2.000	41.97	6	41.65	−0.8%	41.75	−0.5%
4.000	44.66	4	44.55	−0.2%	44.55	−0.2%
100.000	46.49	3	46.49	−0.0%	43.52	−6.4%

Table 5.1: Results for Set 1

5.7.2.2 Results for Set 2

The results for Set 2 are listed in Table 5.2. For $\rho \to \infty$, it will be optimal for each server to accept a new job as soon as the job currently in service has passed 3 service phases, because $\max_k (r(k) - 20)/k = 13\frac{1}{3}$, which is attained for $k^* = 3$. Hence, we know that $g^* \to (13\frac{1}{3} - 1) \cdot 3 = 37$ as $\rho \to \infty$. Further, note that $\rho = 1.333$ corresponds to $\rho_{k^*} = 1.000$. Values of ρ below 1 have been omitted from the table, because in those cases the relative errors were negligible, as was the case for Set 1.

ρ	g^*	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	Δ_{Minor}	g_{Major}	Δ_{Major}
1.000	30.82	9	30.81	−0.0%	30.81	−0.0%
1.333	33.28	7	33.26	−0.1%	33.25	−0.1%
1.667	34.34	6	34.32	−0.1%	34.31	−0.1%
2.000	34.92	5	34.81	−0.3%	34.81	−0.3%
4.000	36.07	5	35.95	−0.3%	35.95	−0.3%
100.000	36.98	3	36.98	−0.0%	36.23	−2.0%

Table 5.2: Results for Set 2

5.7.2.3 Results for Set 3

The results for Set 3 are listed in Table 5.3. We only have results for Minor on display, since $c = 0$ implies that the policies corresponding to Minor and Major are identical, as already noted. From $c = 0$ and the concavity of the reward function, we can derive immediately that for $\rho \to \infty$, it will be optimal for each server to accept a new job as soon as the job currently in service has passed the first service phase. Hence, $g^* \to (25 - 5) \cdot 3 = 60$ as $\rho \to \infty$. Further, note that $\rho = 4.000$ corresponds to $\rho_{k^*} = 1.000$.

ρ	g^*	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	Δ_{Minor}
1.000	32.49	8	32.42	−0.2%
1.333	38.41	7	38.31	−0.3%
1.667	42.55	6	42.53	−0.0%
2.000	45.59	6	45.35	−0.5%
4.000	53.41	4	53.41	−0.0%
6.000	55.56	4	55.56	−0.0%
100.000	59.83	3	59.83	−0.0%

Table 5.3: Results for Set 3

5.7.3 A 6-server system with at most 4 service phases per job

We consider the same system as in Section 5.7.2, but let $s = 6$ instead of 3, so the system has twice as many servers at its disposal. Again, we consider three sets of instances and various values of ρ within each set. In particular, **Set 4** features the same cost structure as Set 1, **Set 5** features the same cost structure as Set 2, and **Set 6** features the same cost structure as Set 3.

5.7.3.1 Results for Set 4

The results for Set 4 are listed in Table 5.4. Note that $g^* \to 93$ as $\rho \to \infty$.

ρ	g^*	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	Δ_{Minor}	g_{Major}	Δ_{Major}
0.333	26.99	39	26.99	−0.0%	26.99	−0.0%
0.667	53.24	26	53.24	−0.0%	53.24	−0.0%
1.000	72.05	16	71.98	−0.1%	71.98	−0.1%
1.333	80.82	12	80.57	−0.3%	80.57	−0.3%
1.667	85.30	11	85.07	−0.3%	85.10	−0.2%
2.000	87.67	10	86.97	−0.8%	87.08	−0.7%
4.000	91.13	8	90.04	−1.2%	90.44	−0.8%
100.000	92.98	6	92.98	−0.0%	86.89	−6.5%

Table 5.4: Results for Set 4

5.7.3.2 Results for Set 5

The results for Set 5 are listed in Table 5.5. Note that $g^* \to 74$ as $\rho \to \infty$.

ρ	g^*	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	Δ_{Minor}	g_{Major}	Δ_{Major}
1.000	64.68	15	64.64	−0.1%	64.64	−0.1%
1.333	69.37	11	69.29	−0.1%	69.30	−0.1%
1.667	71.03	9	70.91	−0.2%	70.91	−0.2%
2.000	71.81	9	71.55	−0.4%	71.71	−0.1%
4.000	73.09	7	72.97	−0.2%	72.97	−0.2%
100.000	73.98	6	73.98	−0.0%	72.38	−2.2%

Table 5.5: Results for Set 5

5.7.3.3 Results for Set 6

The results for Set 6 are listed in Table 5.6. Note that $g^* \to 120$ as $\rho \to \infty$.

ρ	g^*	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	Δ_{Minor}
1.000	69.26	16	69.06	−0.3%
1.333	82.04	14	81.60	−0.5%
1.667	90.64	12	90.43	−0.2%
2.000	96.68	11	95.71	−1.0%
4.000	111.36	8	111.13	−0.2%
6.000	114.75	7	114.75	−0.0%
100.000	119.79	6	119.79	−0.0%

Table 5.6: Results for Set 6

Example 5.2 To illustrate the concept of non-termination for $i \leq s$, which we have discussed in Section 5.6.2, consider the instance where $\rho = 4.000$. Clearly, for $i \leq 6$, the multi-server optimal termination policy only aborts the service of a job if it has completed all 4 service phases. However, for $i \leq 6$, the optimal termination policy for the corresponding $\mathrm{M}|\mathrm{E}_N^{\mu_i}|1$ model is as follows: for $i = 1, 2$, abort if at least 4 phases have been completed; for $i = 3, 4$, abort if at least 3 phases have been completed; for $i = 5, 6$, abort if at least 2 phases have been completed.

5.7.4 A 4-server system with at most 6 service phases per job

We consider a system with $s = 4$ servers, $N = 6$ service phases and reward function $\underline{r} = (0, 20, 36, 49, 59, 69, 75)$. As before, we consider three sets of instances and various values of ρ within each set. **Set 7** features the same cost structure as Set 2, **Set 8** features the same cost structure as Set 1, and **Set 9** features the same cost structure as Set 3.

5.7.4.1 Results for Set 7

The results for Set 7 are listed in Table 5.7. For $\rho \to \infty$, it will be optimal for each server to accept a new job as soon as the job currently in service has passed 5 service phases, because $\max_k (r(k)-20)/k = 9.8$, which is attained for $k^* = 5$. Hence, we know that $g^* \to (9.8-1)\cdot 4 = 35.2$ as $\rho \to \infty$. Further, note that $\rho = 1.200$ corresponds to $\rho_{k^*} = 1.000$.

ρ	g^*	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	Δ_{Minor}	g_{Major}	Δ_{Major}
0.333	10.87	29	10.87	−0.0%	10.87	−0.0%
0.667	21.31	18	21.31	−0.0%	21.31	−0.0%
1.000	28.70	11	28.66	−0.1%	28.66	−0.1%
1.200	30.82	9	30.75	−0.2%	30.74	−0.3%
1.333	31.66	8	31.57	−0.3%	31.59	−0.2%
1.667	32.83	7	32.64	−0.6%	32.71	−0.4%
2.000	33.45	6	33.17	−0.8%	33.17	−0.8%
4.000	34.57	5	34.09	−1.4%	34.09	−1.4%
100.000	35.12	4	35.12	−0.0%	33.43	−4.8%

Table 5.7: Results for Set 7

5.7.4.2 Results for Set 8

The results for Set 8 are listed in Table 5.8. Note that $g^* \to (13-2)\cdot 4 = 44$ as $\rho \to \infty$, because $\max_k (r(k)-10)/k = 13$, which is attained for $k \in \{2,3\}$. Hence, $k^* = 2$ and therefore $\rho = 3.000$ corresponds to $\rho_{k^*} = 1.000$.

ρ	g^*	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	Δ_{Minor}	g_{Major}	Δ_{Major}
1.000	30.81	11	30.77	−0.1%	30.77	−0.1%
1.333	35.50	8	35.45	−0.1%	35.45	−0.1%
1.667	38.28	7	37.85	−1.1%	37.84	−1.1%
2.000	39.97	7	39.73	−0.6%	39.76	−0.5%
3.000	42.16	6	41.51	−1.5%	41.65	−1.2%
4.000	42.95	5	42.16	−1.8%	42.23	−1.7%
100.000	43.99	4	43.99	−0.0%	42.79	−2.7%

Table 5.8: Results for Set 8

5.7.4.3 Results for Set 9

The results for Set 9 are listed in Table 5.9. Note that $g^* \to (20-5)\cdot 4 = 60$ as $\rho \to \infty$ and that $\rho = 6.000$ corresponds to $\rho_{k^*} = 1.000$, since $k^* = 1$, which follows immediately from $c = 0$ and the concavity of the reward function.

ρ	g^*	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	Δ_{Minor}
1.000	26.83	10	26.80	−0.1%
1.333	32.70	9	32.61	−0.3%
1.667	37.14	8	36.79	−0.9%
2.000	40.54	8	40.32	−0.5%
4.000	50.69	6	50.64	−0.1%
6.000	54.16	5	54.16	−0.0%
8.000	55.69	5	55.45	−0.4%
100.000	59.75	4	59.75	−0.0%

Table 5.9: Results for Set 9

5.7.5 Conclusions

The results presented in Tables 5.1 through 5.9 indicate that for meaningful values of ρ, the VSR heuristic delivers a maximum relative error of up to about 2%. In most cases, the relative error stays well below 1%. A recurrent phenomenon in our numerical study is that the errors generally tend to be higher for higher values of ρ. Alas, we do not have a clear understanding of the cause of this recurrent pattern.

We further observe that the simple admission policy Minor and the more complicated policy Major perform nearly the same, except for extremely large values of ρ, in which case the performance of Major is relatively poor. We already predicted this phenomenon in Section 5.7.1. Based on this, we could give an all-round preference to Minor. An additional option would be to calculate both g_{Minor} and g_{Major} and to use the policy that yields the highest average reward. We note, however, that it will not be possible to compute these values explicitly for large instances, i.e., instances with a large number of servers or a high maximum number of service phases. They have to be obtained by means of simulation, which means that it could take some time before accurate values are obtained.

The conclusions drawn in this section apply to the complete collection of sets of instances we included in our numerical study to evaluate the performance of the VSR heuristic. We mention that the results we obtained for sets other than those considered in Sections 5.7.2 through 5.7.4 are in accordance with these conclusions. These sets also featured systems with up to 6 servers. This means that our conclusions are valid for instances with a fairly modest number of servers. In principle, we have no knowledge of the performance for systems for which the number of servers plus the maximum number of service phases is large. In the next section, we will compare the performance of the VSR heuristic to that of another natural heuristic. In this comparison, it will be possible to consider large instances. As will be illustrated by means of some numerical results, the VSR heuristic outperforms the other heuristic.

5.8 Comparison of two heuristics via simulation

Besides modifying the service process of the $M|E_N|1$ model to obtain a heuristic for the computation of the optimal policy for the multi-server model, as we did in our construction of the VSR heuristic, we could have chosen to alter the arrival process of the $M|E_N|1$ model to obtain a heuristic. A model that comes to mind naturally in this respect is the $E_s|E_N|1$ model of Section 4.2.

5.8.1 Obtaining a heuristic from the $E_s|E_N|1$ model

The main idea is that if we consider some arbitrary server in the multi-server system, then, on average, one out of every s accepted jobs is routed to that particular server. As an approximation, we can model the interarrival time of jobs at each separate server by an Erlang(λ)-s distribution. Subsequently, to obtain a single-server heuristic, we can consider each separate server in isolation. This gives s identical $E_s|E_N|1$ systems. Further, we can allow for the rejection of arrival phases, symbolizing the rejection of a new job in the multi-server system. The optimal policy for this single-server system with extended admission control, which is readily computed, can then be used as a reference for each of the servers in the multi-server system separately.

Here, a natural correspondence is to let i jobs and a completed arrival phases in the single-server system (cf. Section 4.2) correspond to $s \cdot i + a$ jobs in the multi-server system. In the single-server system, we charge consideration costs c/s for the acceptance of an arrival phase and associate a reward

of $r(0)/s$ with the rejection of an arrival phase. This is irrespective of whether an accepted arrival phase actually leads to an extra job in the system, which is the case if the number of arrival phases left has reached zero. The multi-server system faces holding costs of h_{init} per job per unit of time. In the corresponding single-server model, we charge total holding costs of $(i + a/s) \cdot h_{\text{init}}$ per unit of time.

A natural translation from the optimal policy for the single-server system to actual control rules for the multi-server system is then as follows. We let each of the servers in the multi-server system follow the optimal single-server policy. In particular, the prescription by the optimal single-server policy to abort in state (i, k, a) translates to the decision by an individual server in the multi-server system to abort if there are $si+a$ jobs in the system and the server has completed k service phases with respect to the job it is serving. Similarly, the prescription by the optimal single-server policy to reject an arrival phase in state (i, k, a) translates to the decision by an individual server in the multi-server system to reject a new job if there are already $si + a$ jobs in the system and the server has completed k service phases with respect to the job it is serving.

Then, by ensuring a threshold form in the same way as we did in the construction of the VSR heuristic, and by applying higher-level admission and termination control in the same way as we did in the construction of the VSR heuristic, we obtain a natural and practical heuristic based on the $\text{E}_s|\text{E}_N|1$ model with extended admission control. We term this heuristic the ERL ('Erlang') heuristic. Its final form is summarized below.

The ERL *heuristic*

1. *Let be given an instance of the $M|E_N|s$ model.*
2. *Consider the corresponding instance of the $E_s|E_N|1$ model featuring extended admission control. Accepting an arrival phase costs c/s and rejecting it yields $r(0)/s$. The holding costs are equal to $(i + a/s)h_{init}$ per unit of time.*
3. *Compute its optimal admission/termination control policy.*
4. *Fix this policy as follows: make sure that decision* `accept` *is taken as long as $i = 0$ and that decision* `continue` *is taken in state $(i, a) = (1, 0)$ as long as node N has not yet been reached.*

5. *Perform Step 5 of the* VSR *heuristic. This entails ensuring that the admission policy is monotone in k for fixed (i, a).*
6. *Translate this single-server policy to control rules for the multi-server system:*
   i. *In principle, each server in the multi-server system follows the optimal policy obtained in Steps 3 to 5, where the situation of i jobs and a completed arrival phases in the single-server system corresponds to $si + a$ jobs in the multi-server system.*
   ii. *Apply higher-level admission control, as described for the* VSR *heuristic.*
   iii. *Apply higher-level termination control, as described for the* VSR *heuristic.*

5.8.2 Numerical results

In case of a fairly modest number of servers, a separate numerical study (the results of which we omit here) pointed out that the VSR heuristic yields better results than the ERL heuristic, although the difference in performance is not substantial. More importantly, we would like to be able to say something about the strength of the VSR heuristic in case of larger systems. Unfortunately, we cannot compare g_{Minor} or g_{Major} to g^*, since the latter cannot be computed, but we can show numerically that the superiority of the VSR heuristic over the ERL heuristic grows as the system becomes larger, hence justifying the selection of the VSR heuristic to be used as 'the' heuristic for the multi-server model. It remains the question whether the average reward yielded by the VSR heuristic lies very close to the optimal average reward. We do know g^* as $\rho \to \infty$, however, and our results show that for sizeable values of ρ, the average reward yielded by the heuristic lies close to the g^* corresponding to $\rho \to \infty$, from which we can conclude that the performance is at least good.

The average reward associated with a policy corresponding to either the VSR heuristic or the ERL heuristic can be computed by means of simulation. Once the heuristic has established a single-server policy, we can simulate the multi-server system with given decisions, taken from the single-server policy. We have tested the performance of both heuristics for a selection of larger instances of the multi-server model. Below, we show the results we obtained

for two instances whose results are illustrative of the performance of the heuristics, and of how their realization and performance compare. In both cases, we considered 5 test values of the gross workload: $\rho = 1$, $\rho^* = 1$ and 3 values around and in between these values. We only show the results for admission policy Minor. Admission policy Major yielded similar results.

5.8.2.1 Results for an $M|E_{10}|10$ system

We consider a ten-server system with $h_{\text{init}} = 1.5$, $c = 30$, $N = 10$ and reward function $\underline{r} = (0, 24, 45, 60, 71, 79, 86, 91, 95, 98, 100)$. It is straightforward to check that $k^* = 4$, so we know that $g^* \to (10.25 - 1.5) \cdot 10 = 87.5$ as $\rho \to \infty$ and that $\rho_{k^*} = 1$ corresponds to $\rho = 2.5$. Based on this, we select the set $R := \{1, 1.5, 2, 2.5, 3\}$ of values of ρ. This constitutes **Set 10** of instances.

The results for Set 10 are listed in Table 5.10. For each instance, we conducted 10 simulation runs, each considering 10^6 units of time (hence considering approximately 10^6 service phase completions and 10^6 to $3 \cdot 10^6$ arrivals per run, depending on which $\rho \in R$ is being considered). For each instance and either the VSR or ERL heuristic, this yielded average rewards per unit of time of $g_1, \ldots, g_{10}$. The value g_{Minor} listed in the table is defined as the average of these 10 values, with σ its standard deviation. Further, Δ is the relative gain if we were to use the VSR heuristic instead of the ERL heuristic. Note that g^* remains unknown for $\rho \in R$. We only have 87.5 as an upper bound.

	VSR			ERL			
ρ	$i_{\max}^{\text{Minor}}$	g_{Minor}	σ	$i_{\max}^{\text{Minor}}$	g_{Minor}	σ	Δ
1.000	33	52.89	0.04	41	47.80	0.02	11%
1.500	23	69.95	0.04	31	57.31	0.02	22%
2.000	17	78.42	0.03	26	65.50	0.03	20%
2.500	15	82.42	0.04	23	69.15	0.03	19%
3.000	13	84.02	0.03	20	70.29	0.04	20%

Table 5.10: Results for Set 10

5.8.2.2 Results for an M|$\mathbf{E_{16}}$|8 system

We consider an eight-server system with $h_{\text{init}} = 1$, $c = 100$, $N = 16$ and reward function $\underline{r} = (0, 30, 56, 80, 102, 122, 138, 151, 162, 172, 182, 191, 200, 207, 212, 216, 220)$. It is straightforward to check that $k^* = 12$, so we know that $g^* \to (8\frac{1}{3} - 1) \cdot 8 = 58\frac{2}{3}$ as $\rho \to \infty$ and that $\rho_{k^*} = 1$ corresponds to $\rho = 1\frac{1}{3}$. Based on this, we select the set $R := \{1, 1.333, 1.667, 2, 3\}$ of values of ρ. This constitutes **Set 11** of instances. The results for this set are listed in Table 5.11.

	VSR			ERL			
ρ	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	σ	$i_{\max}^{\mathsf{Minor}}$	g_{Minor}	σ	Δ
1.000	19	48.07	0.05	26	46.23	0.03	4%
1.333	13	54.09	0.04	21	48.52	0.02	11%
1.667	11	55.95	0.03	18	49.63	0.02	13%
2.000	10	56.65	0.02	16	49.90	0.02	14%
3.000	10	57.40	0.02	16	50.62	0.02	13%

Table 5.11: Results for Set 11

5.8.3 Conclusions

The overall picture is that the ERL heuristic is clearly outperformed by the VSR heuristic. A trend is that the difference in performance becomes more significant for larger s. An explanation for this is that the approximation of the arrival process used by the ERL heuristic and the subsequent translation from i jobs in the single-server policy to si jobs in the multi-server policy are less accurate than anticipated. Consequently, the deviation increases with s. Furthermore, Tables 5.10 and 5.11 indicate that the ERL heuristic takes more insurance against idle time than the VSR heuristic. Apparently, it takes too much insurance, which only causes extra holding costs, resulting in a lower average reward than could have been obtained by maintaining a smaller safety stock. Finally, we note that for sizeable ρ, the average reward obtained by the VSR heuristic lies close to the upper bound corresponding to $\rho \to \infty$, which cannot even be reached by the optimal policy for those values of ρ. Hence, although we do not know the exact difference between g_{Minor} and g^*, we do know empirically that the relative error is fairly small.

5.9 Conclusions

We have discussed a multi-server extension of the $M|E_N|1$ model. The focus was on numerical aspects of this $M|E_N|s$ model and its optimal control policy. We have advocated the need for a heuristic for this model, and have presented a heuristic which is based on the optimal policy for a slightly modified version of the $M|E_N|1$ model. The heuristic was refined and evaluated by means of a numerical study. The numerical results showed that our heuristic performs very well for multi-server systems with a fairly small number of servers. For larger multi-server systems, we used simulation to compare our heuristic to another natural heuristic. We showed empirically that the first delivers superior performance.

There remain many intriguing open problems to be investigated. For example, one may attempt to prove the following threshold result for the multi-server system, which we pose as a conjecture.

Conjecture 5.5 *Let $n \geq 1$ and $(i, \underline{s})$ such that $i > s$. If it is optimal to abort at least one job in state $(i, \underline{s})$, then it is optimal as well to abort at least one job in state $(i+1, \underline{s})$.*

Another challenge would be to prove or possibly disprove Conjectures 5.2 and 5.3, made with respect to the $M|E_N^{\mu_i}|1$ model.

Further, from a practical point of view, a natural and highly useful direction for further research would be to investigate the performance of heuristics for multi-server systems with more general routing mechanisms, such as the feed-forward routing mechanism of the $M|E_{N,\,jumps}|1$ model, or the general tree-structured routing mechanism discussed at the end of Chapter 4.

6

A two-class preemptive priority queue with admission and termination control

In the preceding chapters, we considered several on-line decision models with both admission and termination control. A common feature of these models was that there was only one class of jobs. This means that, upon arrival to the system, jobs were assumed to be mutually indistinguishable. This represents the situation that the actual content of a job is unknown beforehand, and will only become clear during the service process. However, if the nature of the process is such that new jobs can very well be classified into distinct classes of jobs, based on job characteristics that can already be recognized before any capacity engagement, then one-class models cannot be used to accurately represent the system, and multi-class models are required.

There is an extensive literature on multi-class queueing models, either with or without decision features. In the latter case, the focus has mainly been on admission and scheduling control. For example, a classic *static* scheduling result, established by various authors, is the $c\mu$ rule; see, e.g., Kakalik [30]. In case of a single-server queue with Poisson arrivals, non-preemptive service discipline and independent service times, where class-i jobs have service rate μ_i and holding costs c_i per unit of time spent in the system, the $c\mu$ rule minimizes the expected holding costs per unit of time over all jobs in the system. At any time, the $c\mu$ rule gives priority to a class-i job if i maximizes $c_i\mu_i$ among all classes of which there are currently jobs present. It is an example of an *index* rule (where $c_i\mu_i$ is the index). Harrison [25] showed that in the case of job rewards, the objective of maximizing the expected discounted reward over an infinite horizon is optimized by an index rule too.

A specific example of a *dynamic* scheduling control model is the two-class preemptive priority queueing model of Groenevelt et al. [22], which we already mentioned in Section 1.5.3. In this chapter, which is based on Brouns and Van der Wal [12], we will consider a similar model, but with a different cost structure and additional control features. In particular, we will study a two-class $M^{\lambda_1,\lambda_2}|M^{\mu}|1$ preemptive priority queue with admission as well as termination control. The two classes are assumed to represent more or less valuable jobs, where the distinction can already be made upon arrival to the system. The admission and termination control features are the same as in Chapter 3. This means that one has the option to accept or reject new type-1 or type-2 jobs, and, at any time, one has the option to remove any number of type-1 or type-2 jobs from the system. We will show that there exist optimal threshold policies for these two types of decisions. Formally, there is also a third decision feature: the service order. However, we will show presently that this third type is not a real issue—the conditions will be such that the system is essentially a priority queue in which type-2 jobs have priority over type-1 jobs.

The remainder of this chapter is organized as follows. In Section 6.1, we describe the model in detail. We also reduce the model by recognizing that type-2 jobs are preferable to type-1 jobs and should be given priority over type-1 jobs. Section 6.2 gives an overview of the main results for the reduced model, and the line of proof. The proof itself is given in Section 6.3. Finally, in Section 6.4, under a certain restriction on the admission control structure, we extend our results to the multi-server version of our model.

6.1 Model description

The basic model we study is a two-class single-server queueing system with infinite buffer capacity. Type-i jobs, $i = 1, 2$, arrive at the station according to a Poisson process with arrival rate $\lambda_i \geq 0$. The workload of a job is exponential with mean service time $1/\mu$, independent of which of the two classes the job belongs to. The service discipline is unrestricted. Queued jobs may be rearranged at any time and at any time the service of a job may be interrupted—and resumed later, if so desired—in order to commence the service of another job. The system is controlled in three ways: one has to decide to accept or reject new arrivals, one has to decide to remove jobs from the system or to maintain them, and one has to decide what job to serve.

We assume that the decision maker knows the number of type-1 jobs and the number of type-2 jobs in the system. The structure of the system is that of a (semi-)Markovian decision process. It can be described as follows.

States: The state of the system is described by the tuple (x, y), where x is the number of type-1 jobs in the system and y is the number of type-2 jobs in the system. A two-dimensional state space suffices, because service may be interrupted at any time and because service times are exponential. The question what job is served is part of the decision, not of the state. We also use the intermediate states $(x, y, \text{arr}/1)$ and $(x, y, \text{arr}/2)$ immediately after the arrival of a type-1 or type-2 job, respectively.

Events: We distinguish two possible events: (i) the arrival of a new job and (ii) a service completion.

Decisions: If the event is an arrival, then first it has to be decided whether to accept (decision `accept`) or reject (decision `reject`) the newly arrived job. If it concerns a type-1 job, then this changes the state from $(x, y, \text{arr}/1)$ to $(x + 1, y)$ or (x, y), respectively. If, alternatively, it concerns a type-2 job, then this changes the state from $(x, y, \text{arr}/2)$ to $(x, y + 1)$ or (x, y), respectively. Next, it is decided either to maintain all jobs in the system (decision `continue`) or to remove one or more jobs from the system (decision `abort`), and it is decided what job to serve. If this is not the job already in service, then the job in service is either removed from the system or put back in the queue. The service of a job that has been placed back in the queue can be resumed later—it need not be started all over again.

If the event is a service completion, then only the continue/abort decision and the decision what job to serve have to be taken.

Costs and rewards: The reward for a type-i job, $i = 1, 2$, is r_i, where it is assumed that $r_2 > r_1 > 0$. This reward is earned upon service completion. Jobs that do not complete their service, e.g., because they are rejected upon arrival or removed while awaiting service, receive a reward of zero. Removing jobs from the system is free of charge.

Apart from these rewards there are holding costs for the jobs residing in the system, either awaiting service or being served. We assume these costs are linear in the number of jobs and class-independent, namely, $mh \geq 0$ per unit of time when there are $m = x + y$ jobs present. In addition, each time a job is admitted to the system, class-independent consideration costs $c \geq 0$ are incurred. Rejecting jobs is free of charge. We assume that $r_1 > c + h/\mu$, otherwise it will not be interesting to serve any type-1 jobs.

Finally, there are no switching costs, i.e., no costs are incurred if we start serving a type-2 job if the previous job in service was a type-1 job and vice versa.

Discounting: We discount at a rate $\alpha \geq 0$, i.e., rewards and costs at time t are to be multiplied by $\exp(-\alpha t)$. As in Chapter 3, the discount rate α is treated as the rate by which the process vanishes; cf. Section 3.1.

Criterion: The objective is to maximize the expected (discounted) reward over an n-period time horizon. We allow $\lambda_1+\lambda_2>\mu$, as well as $\lambda_1=\lambda_2=0$. In the latter case, there are two batches, one consisting of type-1 jobs awaiting service and one consisting of type-2 jobs awaiting service, without any future arrivals.

Uniformization: The system evolves at arrival times, at service completion times, and eventually when the process vanishes. Applying uniformization, we can consider that transitions occur at the jump times of a Poisson process with rate $\lambda_1 + \lambda_2 + \mu + \alpha > 0$. By scaling time, we take $\lambda_1 + \lambda_2 + \mu + \alpha = 1$ without loss of generality. Then, with probability $\lambda_i \geq 0$, $i = 1, 2$, a transition concerns the arrival of a type-i job, with probability $\mu > 0$ it concerns a service completion and with probability $\alpha \geq 0$ the process vanishes. A service completion is either a real service completion or an artificial service completion when the server idles. In the latter case the state of the system stays $(0, 0)$ and the `continue` decision is taken by definition.

6.1.1 Dynamic Programming formulation

We now summarize and complete the model in terms of a mathematical formulation. We first introduce some general value functions and then promptly reduce the model by showing that it is optimal to always give type-2 jobs priority over type-1 jobs. We then give the DPEs for the reduced model. After that, we successively state and prove our main theorem.

Recapitulating, x and y denote the number of type-1 and type-2 jobs in the system, respectively, and (x, y) is the state of the system for $x, y \geq 0$. We will use the following notation:

- $V_n(x, y)$ denotes the maximum expected n-period α-discounted reward when the current state, *just before* the next decision to continue or abort, is (x, y). State (x, y) may be the result of an arrival immediately after the accept/reject decision.

- $V_n(x, y; \pi)$ denotes the maximum expected n-period α-discounted reward when the current state, just before the next decision to continue or abort, is (x, y), and given that decision π is chosen in that state, where $\pi \in \{\texttt{continue}, \texttt{abort}\}$ if either $x = 0$ or $y = 0$ and $\pi \in \{\texttt{continue}/1, \texttt{continue}/2, \texttt{abort}/1, \texttt{abort}/2\}$ if $x, y > 0$. Here, $\texttt{continue}/i$ means we take the `continue` decision—i.e., it is decided to maintain all jobs currently present—and commence the service of a type-i job, $i = 1, 2$, and $\texttt{abort}/i$ means we remove a type-i job from the system, $i = 1, 2$, after which we make a transition to state $(x-1, y)$ if $i = 1$ and a transition to state $(x, y-1)$ if $i = 2$. Let π^* denote the optimal decision, so $V_n(x, y) = V_n(x, y; \pi^*)$. Note, again, that in the notation π^* the dependence on x, y and n is suppressed, and that we use commas in our notation to separate state characteristics and a semi-colon to separate the decision from the state.
- $V_n(x, y, \mathrm{arr}/i)$ denotes the maximum expected n-period reward when the current state is (x, y), given that at this very point in time an arrival event occurs, concerning a type-i job, $i = 1, 2$.
- $V_n(x, y, \mathrm{arr}/i; \pi)$ denotes the maximum expected n-period reward when the current state is (x, y), given that there is an arrival of a type-i job, $i = 1, 2$, at this point in time and given that decision π is chosen in that state, where $\pi \in \{\texttt{accept}, \texttt{reject}\}$. Again, π^* denotes the optimal decision, so $V_n(x, y, \mathrm{arr}/i) = V_n(x, y, \mathrm{arr}/i; \pi^*)$.
- Finally, when time hits zero, all jobs currently in the system yield a reward of zero for not having completed service. So, $V_0(x, y) = 0$ for all x, y.

Proposition 6.1 *For all $n \geq 0$ and $x, y \geq 0$,*

$$0 \leq V_n(x, y+1) - V_n(x+1, y) \leq r_2 - r_1.$$

Proof. We first consider the left-hand inequality. Consider two n-period process instances of our model, instance $\mathcal{I}_1$ starting in $(x+1, y)$ and instance $\mathcal{I}_2$ starting in $(x, y+1)$. We couple all jobs, all events and all decisions. Instance $\mathcal{I}_1$ follows the optimal policy and instance $\mathcal{I}_2$ copies all actions taken in $\mathcal{I}_1$. In particular, we let the additional type-2 job in $\mathcal{I}_2$ go through exactly the same as the additional type-1 job in $\mathcal{I}_1$. I.e., if $\mathcal{I}_1$ aborts its additional type-1 job, then $\mathcal{I}_2$ aborts its additional type-2 job, and if $\mathcal{I}_1$ takes the additional type-1 job into service, then $\mathcal{I}_2$ takes the additional type-2 job into service.

As long as the additional job does not complete its service, the rewards and costs are the same for both instances. So, if the additional job never completes its service, then the difference in reward between the two instances is zero. If, alternatively, the additional job completes its service at some point in time, generating a reward of r_1 in $\mathcal{I}_1$ and a reward of r_2 in $\mathcal{I}_2$, then $\mathcal{I}_1$ and $\mathcal{I}_2$ become identical immediately after this service completion, so that the difference in reward between the two instances is $r_2 - r_1 > 0$.

The reasoning is almost the same for the right-hand inequality. Again, let instance $\mathcal{I}_1$ start in $(x+1, y)$ and let instance $\mathcal{I}_2$ start in $(x, y+1)$. But now let $\mathcal{I}_2$ follow the optimal policy and let $\mathcal{I}_1$ copy all actions taken in $\mathcal{I}_2$.

□

Corollary 6.1 *Type-2 jobs are preferred to type-1 jobs in the sense that for all $n \geq 0$ and $x, y > 0$, decision* `abort`*/1 in state (x, y) is at least as good as decision* `abort`*/2.*

Proof. Immediate, by noting that

$$V_n(x, y; \texttt{abort}/1) = V_n(x-1, y) \;\geq\; V_n(x, y-1) = V_n(x, y; \texttt{abort}/2).$$

□

Corollary 6.1 makes the use of the notation `abort`/1 and `abort`/2 redundant. We only need the notation `abort`, where it is determined by the state (x, y) what type of job will be removed: a type-1 job if $x > 0$ and a type-2 job if $x = 0$.

Corollary 6.2 *Type-2 jobs are preferred to type-1 jobs in the sense that for all $n \geq 1$ and $x, y > 0$, decision* `continue`*/2 in state (x, y) is at least as good as decision* `continue`*/1.*

Proof. Immediate, from

$$\begin{aligned} V_n(x, y; \texttt{continue}/2) - V_n(x, y; \texttt{continue}/1) \;&= \\ &\mu[V_{n-1}(x, y-1) + r_2] - \mu[V_{n-1}(x-1, y) + r_1] \end{aligned}$$

and the right-hand inequality of Proposition 6.1.

□

Corollary 6.2 makes the use of the notation `continue`/1 and `continue`/2 redundant. We only need the notation `continue`, where it is determined by the state (x, y) what type of job will be served: a type-2 job if $y > 0$ and a type-1 job if $y = 0$.

Then our (reduced) model is defined by the following set of DPEs. Again, to save space, we will usually write `ab` for `abort` and `co` for `continue` in formal expressions, and also `ac` for `accept` and `rj` for `reject`.

$$V_0(x, y) = 0 \qquad x, y \geq 0$$

For $n \geq 0$:

$$V_n(x, y, \mathrm{arr}/1) = \mathbf{max}\{V_n(x+1, y) - c,\ V_n(x, y)\} \qquad x, y \geq 0$$

$$V_n(x, y, \mathrm{arr}/2) = \mathbf{max}\{V_n(x, y+1) - c,\ V_n(x, y)\} \qquad x, y \geq 0$$

and for $n \geq 1$:

$$V_n(0, 0) = V_n(0, 0; \mathtt{co})$$

$$V_n(0, 0; \mathtt{co}) = \textstyle\sum_{i=1}^{2} \lambda_i V_{n-1}(0, 0, \mathrm{arr}/i) + \mu V_{n-1}(0, 0)$$

$$V_n(x, y) = \mathbf{max}\{V_n(x, y; \mathtt{co}),\ V_n(x, y; \mathtt{ab})\} \qquad x + y > 0$$

$$V_n(x, y; \mathtt{co}) = \textstyle\sum_{i=1}^{2} \lambda_i V_{n-1}(x, y, \mathrm{arr}/i) + \mu[V_{n-1}(x, y-1) + r_2] - (x+y)h \qquad x \geq 0, y > 0$$

$$V_n(x, 0; \mathtt{co}) = \textstyle\sum_{i=1}^{2} \lambda_i V_{n-1}(x, 0, \mathrm{arr}/i) + \mu[V_{n-1}(x-1, 0) + r_1] - xh \qquad x > 0$$

$$V_n(x, y; \mathtt{ab}) = V_n(x-1, y) \qquad x > 0, y \geq 0$$

$$V_n(0, y; \mathtt{ab}) = V_n(0, y-1) \qquad y > 0$$

6.2 Overview of the results

We will prove the following theorem.

Theorem 6.1
{Characterization of the optimal admission/termination policy}
For any remaining number of periods n, the optimal admission/termination policy can be characterized as follows:

1. i. *If it is optimal to reject an arriving* ***type-1*** *job in state (x, y), then it is optimal as well to reject it in all states $(x, y + j)$ with $j > 0$ and in all states $(x + j, y - j)$ with $0 < j \leq y$, and thus in all states $(x + j, y)$ with $j > 0$.*

   ii. *If it is optimal to reject an arriving* ***type-2*** *job in state (x, y), then it is optimal as well to reject it in all states $(x + j, y)$ with $j > 0$ and in all states $(x - j, y + j)$ with $0 < j \leq x$, and thus in all states $(x, y + j)$ with $j > 0$.*

2. *If it is optimal to abort in state (x, y), then it is optimal as well to abort in all states $(x, y + j)$ with $j > 0$, in all states $(x + j, y - j)$ with $0 < j \leq y$, and thus in all states $(x + j, y)$ with $j > 0$.*

3. *If it is optimal to reject an arriving type-2 job in state (x, y), then it is optimal as well to reject an arriving type-1 job in state (x, y).*

We refer to Figure 6.1 for a graphical representation of the structure of a typical admission/termination policy. In the optimal termination policy, the hollow dots represent states in which we continue and the solid dots represent states in which we abort. The polyline marks the termination region. In the optimal admission policy, the hollow dots represent states in which we accept any new job, the half-filled dots represent states in which we only accept a new job if it is a type-2 job and the solid dots represent states in which we reject any new job.

Remark 6.1 It is easily seen that Part 3 of Theorem 6.1 is immediate from Proposition 6.1 and the DPEs for $V_n(x, y, \text{arr}/1)$ and $V_n(x, y, \text{arr}/2)$.

Remark 6.2 If $c = 0$, then it is always optimal to accept an arriving job, since it may be aborted at the same moment in time at no additional cost.

Example 6.1 Consider the following instance of our model: $\mu = 0.45$, $\lambda_1 = 0.35$, $\lambda_2 = 0.2$, $\alpha = 0$, $h = 0.5$, $c = 5$, $r_1 = 10$ and $r_2 = 25$. Let $n = 5$. The optimal admission policy—should there be an arrival at this point in time—and the optimal termination policy are given by Figure 6.1.

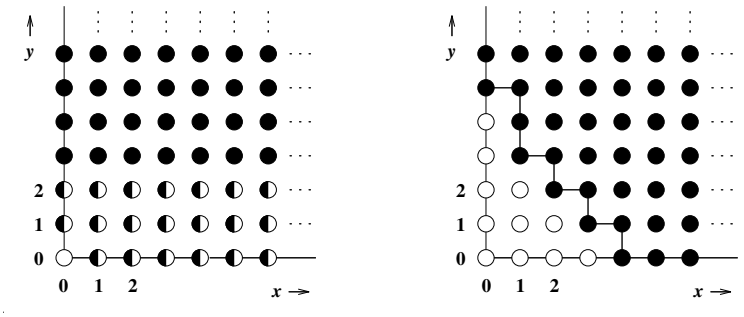

Figure 6.1: Optimal admission (*left-hand side*) and termination (*right-hand side*) policies for Example 6.1

6.2.1 The line of proof

In order to establish Parts 1 and 2 of Theorem 6.1, we will prove the following monotonicity results, which will be interpreted directly below.

Proposition 6.2 {Key Proposition}
For all $n \geq 0$ *and* $x, y \geq 0$,

$$\begin{aligned}
V_n(x+1,y) - V_n(x,y) &\geq V_n(x+2,y) - V_n(x+1,y), && (6.1)\\
V_n(x,y+1) - V_n(x,y) &\geq V_n(x,y+2) - V_n(x,y+1), && (6.2)\\
V_n(x+1,y) - V_n(x,y) &\geq V_n(x+1,y+1) - V_n(x,y+1), && (6.3)\\
V_n(x+1,y+1) - V_n(x,y+1) &\geq V_n(x+2,y) - V_n(x+1,y), && (6.4)\\
V_n(x+1,y+1) - V_n(x+1,y) &\geq V_n(x,y+2) - V_n(x,y+1), && (6.5)\\
V_n(0,y+1) - V_n(0,y) &\geq V_n(x+1,y+1) - V_n(x,y+1). && (6.6)
\end{aligned}$$

In addition, inequalities (6.1) to (6.6) hold:

- *at arrival times of type-1 jobs; the inequalities are then referred to as* $(6.1^{\mathrm{arr}/1})$ *to* $(6.6^{\mathrm{arr}/1})$,
- *at arrival times of type-2 jobs; the inequalities are then referred to as* $(6.1^{\mathrm{arr}/2})$ *to* $(6.6^{\mathrm{arr}/2})$,
- *given that we take the (not necessarily optimal) decision* `continue` *in each of the states that appear in the respective inequality; the inequalities are then referred to as* $(6.1^{\mathtt{co}})$ *to* $(6.6^{\mathtt{co}})$.

Inequality (6.1) states that $V_n(x, y)$ is concave in x, i.e., the value of an additional type-1 job is non-increasing in x for fixed y. Inequality (6.2) states that $V_n(x, y)$ is concave in y as well. Inequality (6.3) states that $V_n(x, y)$ is *submodular*, i.e., the value of an additional type-1 job is non-increasing in y for fixed x and the value of an additional type-2 job is non-increasing in x for fixed y; cf. (1.3). Inequality (6.4) states that $V_n(x, y)$ is *subconcave* in x, i.e., the value of an additional type-1 job is non-increasing in x for a fixed total number of jobs in the system; cf. (1.7). Inequality (6.5) states that $V_n(x, y)$ is subconcave in y as well. Inequality (6.6) is an auxiliary inequality, which is used in our proofs of the other inequalities for certain boundary states.

Remark 6.3 Adding (6.4) to (6.3), we obtain (6.1). Adding (6.5) to (6.3), we obtain (6.2). Furthermore, for $x, y \geq 0$, (6.6) can be obtained through

$$\begin{array}{lcll} V_n(0, y+1) - V_n(0, y) & \geq & \{\text{Proposition 6.1}\} & V_n(1, y) - V_n(0, y) \\ & \geq & \{(6.1)\ x \text{ times}\} & V_n(x+1, y) - V_n(x, y) \\ & \geq & \{(6.3)\} & V_n(x+1, y+1) - V_n(x, y+1). \end{array}$$

Therefore, it suffices to prove the set of inequalities (6.3) to (6.5). However, it will be convenient in our proofs to make use of (6.1), (6.2) and (6.6) as well. One may easily verify that these implications also apply at arrival times and given that we take the `continue` decision in each state appearing in the respective inequality.

One might conjecture the reverse of (6.4), i.e., that the value of an additional type-1 job is non-decreasing in x for a fixed total number of jobs in the system. This would imply that (6.4) holds by equality for all $x, y \geq 0$. However, the conjecture translates to the assertion that for all $x, y \geq 0$,

$$V_n(x+2, y) - V_n(x+1, y) \geq V_n(x+1, y+1) - V_n(x, y+1), \qquad (6.7)$$

which does not hold in general. For example, in the instance considered in Example 6.1, $V_5(2, 1) - V_5(1, 1)$ is 1.0863 ± 0.0002 (the accuracy in the calculations is 0.0002), whereas $V_5(1, 2) - V_5(0, 2)$ is 1.3216 ± 0.0002, so (6.7) is surely violated.

However, if we let $n \to \infty$ in the instance considered in Example 6.1, then for each pair $(x + 1, y)$ with $0 \leq x \leq 5$ and $0 \leq y \leq 14$ (following the optimal policy, the number of type-1 jobs in the system will never exceed 5 and the maximum number of type-2 jobs in the system will never exceed 14) the left-hand side and right-hand side of (6.7) converge to the same value. We have numerically analysed a variety of other instances and have always found the same result. This leads to the following conjecture.

Conjecture 6.1 *Let the total number of jobs in the system be fixed and at least 1. Then for $n \to \infty$ the value of an additional type-1 job is constant in x.*

The conjecture implies that for $n \to \infty$ and $x \geq 1$, the decision to abort or not to abort is determined solely by the total number of jobs in the system, i.e., $x + y$, and not by x and y individually.

6.3 Proof of the Key Proposition

The proof of the Key Proposition is analogous to the proof of the Key Proposition for the M|E$_N$|1 model of Chapter 3. It uses induction on the remaining number of periods and runs as follows.

Step 0: Observe that (6.3), (6.4) and (6.5) hold for $n = 0$.

Step 1: Assuming (6.3) to (6.5) to hold for some $n \geq 0$, prove $(6.3^{\,\mathrm{arr}/1})$ to $(6.5^{\,\mathrm{arr}/1})$ for n, as well as $(6.3^{\,\mathrm{arr}/2})$ to $(6.5^{\,\mathrm{arr}/2})$ for n.

Step 2: Using this result, prove that $(6.3^{\,\mathtt{co}})$ to $(6.5^{\,\mathtt{co}})$ hold for $n + 1$.

Step 3: Finally, prove that (6.3) to (6.5) also hold for $n + 1$.

In the proof we will make use of Lemma 1.1.

Proof of the Key Proposition.

Step 0. Inequalities (6.3) to (6.5) hold by definition for $n = 0$.

Induction hypothesis. Assume that for some $n \geq 0$, inequalities (6.3) to (6.5) hold for all $x, y \geq 0$. This will be our induction hypothesis.

Step 1. Under the induction hypothesis, we will show that $(6.3^{\,\mathrm{arr}/1})$ to $(6.5^{\,\mathrm{arr}/1})$ and $(6.3^{\,\mathrm{arr}/2})$ to $(6.5^{\,\mathrm{arr}/2})$ hold for n.

Let $x, y \geq 0$. Let us first consider $(6.3^{\,\mathrm{arr}/1})$, and thus the arrival of a type-1 job.

The next decision, d_1 say, prescribed by the (optimal) policy corresponding to $V_n(x, y, \mathrm{arr}/1)$, is either to accept or to reject the new (type-1) job. Clearly, this also holds for the next decision, d_2 say, prescribed by the (optimal) policy corresponding to $V_n(x + 1, y + 1, \mathrm{arr}/1)$. There are at most

four joint cases (d_1, d_2). These cases can be presented as follows, where $\mathcal{A}$ indicates that `accept` is optimal and $\mathcal{R}$ indicates that `accept` is not optimal:

$$\begin{array}{lll}
\mathcal{AA} : V_n(x{+}1,y) - c \geq V_n(x,y) & \wedge & V_n(x{+}2,y{+}1) - c \geq V_n(x{+}1,y{+}1), \\
\mathcal{AR} : V_n(x{+}1,y) - c \geq V_n(x,y) & \wedge & V_n(x{+}2,y{+}1) - c < V_n(x{+}1,y{+}1), \\
\mathcal{RA} : V_n(x{+}1,y) - c < V_n(x,y) & \wedge & V_n(x{+}2,y{+}1) - c \geq V_n(x{+}1,y{+}1), \\
\mathcal{RR} : V_n(x{+}1,y) - c < V_n(x,y) & \wedge & V_n(x{+}2,y{+}1) - c < V_n(x{+}1,y{+}1).
\end{array}$$

We will show that inequality $(6.3^{\,\mathrm{arr}/1})$ holds for each case separately, irrespective of the question whether that case can actually occur. This is done by choosing an appropriate decision that is to be taken in the state corresponding to the leftmost term of inequality $(6.3^{\,\mathrm{arr}/1})$, i.e., state $(x+1, y, \mathrm{arr}/1)$, and an appropriate decision that is to be taken in the state corresponding to the rightmost term of the inequality, i.e., state $(x, y+1, \mathrm{arr}/1)$, such that we obtain an inequality that holds under the induction hypothesis, and by subsequently applying Lemma 1.1.

E.g., under $\mathcal{AA}$,

$$\begin{array}{rcl}
V_n(x+1,y,\mathrm{arr}/1;\mathtt{ac}) - V_n(x,y,\mathrm{arr}/1) & & \\
\quad = & V_n(x+2,y) - c - [V_n(x+1,y) - c] & \\
\quad = & V_n(x+2,y) - V_n(x+1,y) & \\
\quad \geq & \{\textsf{induction hypothesis; } (6.3)\} & \\
 & V_n(x+2,y+1) - V_n(x+1,y+1) & \\
\quad = & V_n(x+1,y+1,\mathrm{arr}/1) - V_n(x,y+1,\mathrm{arr}/1;\mathtt{ac}), &
\end{array}$$

to which we apply Lemma 1.1 to obtain the desired result for this case.

It is easy to see that the reasoning for case $\mathcal{AA}$ is similar for $(6.4^{\,\mathrm{arr}/1})$ and $(6.5^{\,\mathrm{arr}/1})$, and $(6.3^{\,\mathrm{arr}/2})$ to $(6.5^{\,\mathrm{arr}/2})$. For each inequality $(6.j^{\,\mathrm{arr}/i})$, $j = 3, 4, 5$, $i = 1, 2$, case $\mathcal{AA}$ can be dealt with by choosing `accept` in the other two states as well and by then using inequality $(6.j)$, which holds under the induction hypothesis. Similarly, case $\mathcal{RR}$ can always be dealt with by choosing `reject` in the other two states as well.

The remaining cases $\mathcal{AR}$ and $\mathcal{RA}$ are somewhat more complicated. We have conveniently summarized the analysis of these two cases in Table 6.1. For each inequality we give two decisions that can be inserted such that an inequality is obtained that holds, either under the induction hypothesis or because its left-hand side is identical to its right-hand side. In each case, Lemma 1.1 can then be applied to obtain the desired result.

inequality		case		result by
$(6.3^{\mathrm{arr}/1})$	rj	$\mathcal{AR}$	ac	lhs=rhs=c
	rj	$\mathcal{RA}$	ac	induction hypothesis; (6.3); (6.1)
$(6.4^{\mathrm{arr}/1})$	rj	$\mathcal{AR}$	ac	lhs=rhs=c
	rj	$\mathcal{RA}$	ac	induction hypothesis; (6.4); (6.1)
$(6.5^{\mathrm{arr}/1})$	rj	$\mathcal{AR}$	ac	induction hypothesis; (6.4); (6.5)
	rj	$\mathcal{RA}$	ac	induction hypothesis; (6.2)
$(6.3^{\mathrm{arr}/2})$	ac	$\mathcal{AR}$	rj	lhs=rhs
	rj	$\mathcal{RA}$	ac	induction hypothesis; (6.3) twice
$(6.4^{\mathrm{arr}/2})$	rj	$\mathcal{AR}$	ac	induction hypothesis; (6.5); (6.4)
	rj	$\mathcal{RA}$	ac	induction hypothesis; (6.1)
$(6.5^{\mathrm{arr}/2})$	rj	$\mathcal{AR}$	ac	lhs=rhs=c
	rj	$\mathcal{RA}$	ac	induction hypothesis; (6.5); (6.2)

Table 6.1: Analysis of cases $\mathcal{AR}$ and $\mathcal{RA}$

Step 2. Assuming (6.3) to (6.5), $(6.3^{\mathrm{arr}/1})$ to $(6.5^{\mathrm{arr}/1})$ and $(6.3^{\mathrm{arr}/2})$ to $(6.5^{\mathrm{arr}/2})$ for n, we show that (6.3^{co}) to (6.5^{co}) hold for $n+1$. We will use the following lemma.

Lemma 6.1 *For all $n \geq 0$ and $x, y \geq 0$,*

$$V_n(x+1, y) - V_n(x, y) \leq r_1.$$

Proof. The proof uses coupling and a sample path argument; cf. the proof of Proposition 6.1. Let instance $\mathcal{I}_1$ start in $(x+1, y)$ and instance $\mathcal{I}_0$ in (x, y). Couple all events and all decisions. Instance $\mathcal{I}_1$ follows the optimal policy and instance $\mathcal{I}_0$ copies all actions taken in $\mathcal{I}_1$, until either $\mathcal{I}_1$ aborts its additional type-1 job or the additional type-1 job in $\mathcal{I}_1$ completes its service. This occurs at time T, say. Until then, the difference in reward between $\mathcal{I}_1$ and $\mathcal{I}_0$ is at most zero. At time T, either $\mathcal{I}_1$ aborts its additional type-1 job, generating a reward of zero, or this additional type-1 job completes its service, generating a reward of r_1. Immediately afterwards, $\mathcal{I}_1$ and $\mathcal{I}_0$ become identical. So the difference in reward between the two instances is at most r_1.

□

Now consider (6.3^{co}). We distinguish the following three cases, which cover all possible states (x, y): (I) $x \geq 0$, $y > 0$, (II) $x > 0$, $y = 0$ and (III) $x = 0$, $y = 0$.

Case (I) gives

$$\begin{aligned}
&V_{n+1}(x+1,y;\mathsf{co}) - V_{n+1}(x,y;\mathsf{co}) \\
&\quad = \sum_{i=1}^{2} \lambda_i [V_n(x+1,y,\mathrm{arr}/i) - V_n(x,y,\mathrm{arr}/i)] + \\
&\qquad\qquad \mu[V_n(x+1,y-1) - V_n(x,y-1)] - h \\
&\quad \geq \{\text{induction hypothesis; } (6.3^{\mathrm{arr}/1});\ (6.3^{\mathrm{arr}/2});\ (6.3)\} \\
&\qquad \sum_{i=1}^{2} \lambda_i [V_n(x+1,y+1,\mathrm{arr}/i) - V_n(x,y+1,\mathrm{arr}/i)] + \\
&\qquad\qquad \mu[V_n(x+1,y) - V_n(x,y)] - h \\
&\quad = V_{n+1}(x+1,y+1;\mathsf{co}) - V_{n+1}(x,y+1;\mathsf{co}).
\end{aligned}$$

Case (II) gives

$$\begin{aligned}
&V_{n+1}(x+1,0;\mathsf{co}) - V_{n+1}(x,0;\mathsf{co}) \\
&\quad = \sum_{i=1}^{2} \lambda_i [V_n(x+1,0,\mathrm{arr}/i) - V_n(x,0,\mathrm{arr}/i)] + \\
&\qquad\qquad \mu[V_n(x,0) - V_n(x-1,0)] - h \\
&\quad \geq \{\text{induction hypothesis; } (6.3^{\mathrm{arr}/1});\ (6.3^{\mathrm{arr}/2});\ (6.1)\} \\
&\qquad \sum_{i=1}^{2} \lambda_i [V_n(x+1,1,\mathrm{arr}/i) - V_n(x,1,\mathrm{arr}/i)] + \\
&\qquad\qquad \mu[V_n(x+1,0) - V_n(x,0)] - h \\
&\quad = V_{n+1}(x+1,1;\mathsf{co}) - V_{n+1}(x,1;\mathsf{co}).
\end{aligned}$$

Case (III) gives

$$\begin{aligned}
&V_{n+1}(1,0;\mathsf{co}) - V_{n+1}(0,0;\mathsf{co}) \\
&\quad = \sum_{i=1}^{2} \lambda_i [V_n(1,0,\mathrm{arr}/i) - V_n(0,0,\mathrm{arr}/i)] + \mu r_1 - h \\
&\quad \geq \{\text{induction hypothesis; } (6.3^{\mathrm{arr}/1});\ (6.3^{\mathrm{arr}/2});\ \text{Lemma 6.1}\} \\
&\qquad \sum_{i=1}^{2} \lambda_i [V_n(1,1,\mathrm{arr}/i) - V_n(0,1,\mathrm{arr}/i)] + \mu[V_n(1,0) - V_n(0,0)] - h \\
&\quad = V_{n+1}(1,1;\mathsf{co}) - V_{n+1}(0,1;\mathsf{co}).
\end{aligned}$$

Next, consider (6.4^{co}), and distinguish the following two cases, covering all possible states (x,y): (I) $x \geq 0$, $y > 0$ and (II) $x \geq 0$, $y = 0$.

For case (I), the derivation is analogous to the one for (6.3^{co}) for case (I), using the induction hypothesis and inequalities $(6.4^{\mathrm{arr}/1})$, $(6.4^{\mathrm{arr}/2})$ and (6.4).

For case (II), we have

$$\begin{aligned}
&V_{n+1}(x+1,1;\mathsf{co}) - V_{n+1}(x,1;\mathsf{co}) \\
&\quad = \sum_{i=1}^{2} \lambda_i [V_n(x+1,1,\mathrm{arr}/i) - V_n(x,1,\mathrm{arr}/i)] + \\
&\qquad\qquad \mu[V_n(x+1,0) - V_n(x,0)] - h \\
&\quad \geq \{\text{induction hypothesis; } (6.4^{\,\mathrm{arr}/1});\ (6.4^{\,\mathrm{arr}/2})\} \\
&\qquad \sum_{i=1}^{2} \lambda_i [V_n(x+2,0,\mathrm{arr}/i) - V_n(x+1,0,\mathrm{arr}/i)] + \\
&\qquad\qquad \mu[V_n(x+1,0) - V_n(x,0)] - h \\
&\quad = V_{n+1}(x+2,0;\mathsf{co}) - V_{n+1}(x+1,0;\mathsf{co}).
\end{aligned}$$

Finally, consider $(6.5^{\,\mathsf{co}})$, and distinguish the same two cases as considered for $(6.4^{\,\mathsf{co}})$.

For case (I), the derivation is analogous to the one for $(6.3^{\,\mathsf{co}})$ for case (I), using the induction hypothesis and inequalities $(6.5^{\,\mathrm{arr}/1})$, $(6.5^{\,\mathrm{arr}/2})$ and (6.5).

For case (II), we have

$$\begin{aligned}
&V_{n+1}(x+1,1;\mathsf{co}) - V_{n+1}(x+1,0;\mathsf{co}) \\
&\quad = \sum_{i=1}^{2} \lambda_i [V_n(x+1,1,\mathrm{arr}/i) - V_n(x+1,0,\mathrm{arr}/i)] + \\
&\qquad\qquad \mu[V_n(x+1,0) - V_n(x,0) + r_2 - r_1] - h \\
&\quad \geq \{\text{induction hypothesis; } (6.5^{\,\mathrm{arr}/1});\ (6.5^{\,\mathrm{arr}/2});\ \text{Proposition 6.1}\} \\
&\qquad \sum_{i=1}^{2} \lambda_i [V_n(x,2,\mathrm{arr}/i) - V_n(x,1,\mathrm{arr}/i)] + \\
&\qquad\qquad \mu[V_n(x,1) - V_n(x,0)] - h \\
&\quad = V_{n+1}(x,2;\mathsf{co}) - V_{n+1}(x,1;\mathsf{co}).
\end{aligned}$$

Step 3. Assuming (6.3) to (6.5), $(6.3^{\,\mathrm{arr}/1})$ to $(6.5^{\,\mathrm{arr}/1})$ and $(6.3^{\,\mathrm{arr}/2})$ to $(6.5^{\,\mathrm{arr}/2})$ for n, and $(6.3^{\,\mathsf{co}})$ to $(6.5^{\,\mathsf{co}})$ for $n+1$, we show that (6.3) to (6.5) hold for $n+1$.

The line of reasoning resembles the one we followed in Step 1 of our proof. For any of the three inequalities (6.3), (6.4) and (6.5) for $n+1$, we distinguish all possible combinations of optimal decisions in two of the four states, namely, the states s_2 and s_3 in Lemma 1.1.

The results are summarized in Table 6.2, where for each relevant situation appropriate arguments are given, including the choices for the decisions in states s_1 and s_4; cf. Lemma 1.1. If no decision is shown, then we take

the optimal decision in that state. The abbreviation 'ih' used in the table indicates that the induction hypothesis is used there. In addition, to save more space, we will write $(6.j\,^{\mathsf{co}})^{\,m}$ to indicate that inequality $(6.j\,^{\mathsf{co}})$ is used m times, where $1 \leq j \leq 6$ and m is some non-negative integer.

In the notation, $\mathcal{C}$ indicates that `continue` is strictly optimal, $\mathcal{A}$ that `abort` is optimal and X that it is of no interest whether `continue` or `abort` is optimal. Recall that $\mathcal{A}$ is not an option in state $(0,0)$. Furthermore, if it is optimal to first abort j jobs and then to continue, then this is denoted by $\mathcal{A}^j$ and we write $\mathsf{ab}^{\,j}$ for the not necessarily optimal copied decision. So, $\mathcal{C} \equiv \mathcal{A}^0$ and $\mathcal{A}$ equals $\mathcal{A}^j$ for some $j > 0$.

As an example of the reasoning and the notation used in Table 6.2, we consider the second case for inequality (6.3), i.e., $\mathcal{A}^k\mathcal{C}$ for $0 \leq k \leq x$. Then

$$\begin{aligned}
&V_{n+1}(x+1,y;\mathsf{ab}^{\,k}) - V_{n+1}(x,y)\\
&\qquad = \quad V_{n+1}(x+1-k,y;\mathsf{co}) - V_{n+1}(x-k,y;\mathsf{co})\\
&\qquad \geq \quad \{\mathsf{ih};\ \ (6.1\,^{\mathsf{co}})^{\,k}\}\\
&\qquad\quad\ \ V_{n+1}(x+1,y;\mathsf{co}) - V_{n+1}(x,y;\mathsf{co})\\
&\qquad \geq \quad \{\mathsf{ih};\ \ (6.3\,^{\mathsf{co}})\}\\
&\qquad\quad\ \ V_{n+1}(x+1,y+1;\mathsf{co}) - V_{n+1}(x,y+1;\mathsf{co})\\
&\qquad = \quad V_{n+1}(x+1,y+1) - V_{n+1}(x,y+1;\mathsf{co}),
\end{aligned}$$

to which we apply Lemma 1.1 to obtain the desired result for this case.

For all $0 \leq j \leq k \leq x$, $y \geq 0$, $0 < l \leq y$ (provided $y > 0$), $0 < m \leq y+1$, $0 < p \leq y+2$, and $k < q \leq x+1$:

inequality		case		result by
(6.3)	ab	$X\mathcal{A}$		lhs=rhs=0
	$\mathtt{ab}^{\,k}$	$\mathcal{A}^k\mathcal{C}$	co	ih; $(6.1^{\,\mathsf{co}})^{\,k}$; $(6.3^{\,\mathsf{co}})$
	$\mathtt{ab}^{\,x+l}$	$\mathcal{A}^{x+l}\mathcal{C}$	co	ih; $(6.2^{\,\mathsf{co}})^{\,l}$; $(6.6^{\,\mathsf{co}})$
(6.4)	ab	$X\mathcal{A}$		lhs=rhs=0
	$\mathtt{ab}^{\,k}$	$\mathcal{A}^k\mathcal{C}$	co	ih; $(6.1^{\,\mathsf{co}})^{\,k}$; $(6.4^{\,\mathsf{co}})$
	$\mathtt{ab}^{\,x+m}$	$\mathcal{A}^{x+m}\mathcal{C}$	co	ih; $(6.2^{\,\mathsf{co}})^{\,m-1}$; $(6.6^{\,\mathsf{co}})$; $(6.4^{\,\mathsf{co}})$
(6.5)	ab	$X\mathcal{A}^{x+p}$	$\mathtt{ab}^{\,x+(p-1)}$	Proposition 6.1; rhs=0
	$\mathtt{ab}^{\,k}$	$\mathcal{A}^j\mathcal{A}^k$	$\mathtt{ab}^{\,j}$	ih; $(6.4^{\,\mathsf{co}})^{\,k-j}$; $(6.5^{\,\mathsf{co}})$
	$\mathtt{ab}^{\,q}$	$\mathcal{A}^q\mathcal{A}^k$	$\mathtt{ab}^{\,k}$	ih; $(6.3^{\,\mathsf{co}})^{\,q-k}$; $(6.5^{\,\mathsf{co}})$
	$\mathtt{ab}^{\,x+1+l}$	$\mathcal{A}^{x+1+l}\mathcal{A}^k$	$\mathtt{ab}^{\,k}$	ih; $(6.2^{\,\mathsf{co}})^{\,l+1}$; $(6.3^{\,\mathsf{co}})^{\,x-k}$

Table 6.2: Analysis of inequalities (6.3), (6.4) and (6.5) for $n+1$

With Table 6.2 we conclude our proof of the Key Proposition.

□

We now derive Parts 1 and 2 of Theorem 6.1 from the Key Proposition by means of three corollaries. Note that Corollaries 6.3 and 6.4 correspond to the first and second part of Part 1 of Theorem 6.1, respectively, and that Corollary 6.5 corresponds to Part 2 of Theorem 6.1.

Corollary 6.3 *Let $n \geq 0$ and $x, y \geq 0$. If it is optimal to reject an arriving type-1 job in state (x, y), then it is optimal to reject it in all states $(x, y+j)$ with $j > 0$ and in all states $(x+j, y-j)$ with $0 < j \leq y$.*

Proof. Let $n \geq 0$ and $x \geq 0$. It suffices to show that

$$V_n(x,y) \geq V_n(x+1,y) - c \;\Rightarrow\; V_n(x,y+1) \geq V_n(x+1,y+1) - c, \tag{6.8}$$

$$V_n(x,y) \geq V_n(x+1,y) - c \;\Rightarrow\; V_n(x+1,y-1) \geq V_n(x+2,y-1) - c, \tag{6.9}$$

where (6.8) holds for $y \geq 0$ and (6.9) for $y > 0$. One can easily verify that implications (6.8) and (6.9) are immediate from inequalities (6.3) and (6.4), respectively.

□

Corollary 6.4 *Let $n \geq 0$ and $x, y \geq 0$. If it is optimal to reject an arriving type-2 job in state (x, y), then it is optimal to reject it in all states $(x+j, y)$ with $j > 0$ and in all states $(x-j, y+j)$ with $0 < j \leq x$.*

Proof. Let $n \geq 0$ and $y \geq 0$. It suffices to show that

$$V_n(x,y) \geq V_n(x,y+1) - c \;\Rightarrow\; V_n(x+1,y) \geq V_n(x+1,y+1) - c, \tag{6.10}$$
$$V_n(x,y) \geq V_n(x,y+1) - c \;\Rightarrow\; V_n(x-1,y+1) \geq V_n(x-1,y+2) - c, \tag{6.11}$$

where (6.10) holds for $x \geq 0$ and (6.11) for $x > 0$. One can easily verify that implications (6.10) and (6.11) are immediate from inequalities (6.3) and (6.5), respectively.

□

Corollary 6.5 *Let $n \geq 0$. If it is optimal to abort in state (x, y), then it is optimal to abort in all states $(x, y+j)$ with $j > 0$ and in all states $(x+j, y-j)$ with $0 < j \leq y$.*

Proof. Let $n \geq 0$. It suffices to show that

$$V_n(x-1,y) = V_n(x,y) \;\Rightarrow\; V_n(x-1,y+1) \not< V_n(x,y+1), \tag{6.12}$$
$$V_n(0,y-1) = V_n(0,y) \;\Rightarrow\; V_n(0,y) \not< V_n(0,y+1), \tag{6.13}$$
$$V_n(x-1,y) = V_n(x,y) \;\Rightarrow\; V_n(x,y-1) \not< V_n(x+1,y-1), \tag{6.14}$$
$$V_n(0,y-1) = V_n(0,y) \;\Rightarrow\; V_n(0,y-1) \not< V_n(1,y-1), \tag{6.15}$$

where (6.12) holds for $x > 0$, (6.13) for $y > 0$, (6.14) for $x, y > 0$, and (6.15) for $y > 0$. One can easily verify that implications (6.12), (6.13), (6.14) and (6.15) are immediate from inequalities (6.3), (6.2) and (6.4) and Proposition 6.1, respectively.

□

6.3.1 Extension to heterogeneous consideration costs

In our basic model we considered class-independent consideration costs $c \geq 0$. It is readily verified that Proposition 6.1 as well as the Key Proposition and its proof stay intact if we consider class-dependent consideration costs $c_1, c_2 \geq 0$. As a result, Parts 1 and 2 of Theorem 6.1 remain valid if the consideration costs are class-dependent. If $c_1 \geq c_2$, i.e., if the consideration

costs are at least as high for type-1 jobs as for type-2 jobs, then part 3 of Theorem 6.1 remains valid as well, by Proposition 6.1 and the new DPEs for $V_n(x, y, \text{arr}/1)$ and $V_n(x, y, \text{arr}/2)$.

6.4 General multi-server model

In our basic model we considered a single-server queue. The extension of our monotonicity and threshold results for the single-server model to the general multi-server case is not straightforward. If we follow the same approach as for the single-server model, then it turns out that (6.7) is required in order to establish (6.5) for all x, y for the multi-server model. But in Example 6.1 we have seen that (6.7) need not hold. Under the restrictive assumption that $c_2 = 0$ or that type-2 jobs may not be rejected upon arrival, it can be shown that (6.7) holds for the single-server model, so that the value of an additional type-1 job only depends on the total number of jobs in the system, and not on the number of type-1 jobs and the number of type-2 jobs individually. In this case our results can be extended to the multi-server model, which we term the $\text{M}^{\lambda_1,\lambda_2}|\text{M}^{\mu}|s$ model in the remainder of this section.

We start our analysis by recognizing that the extension of Proposition 6.1 to the multi-server case is straightforward and hence Corollaries 6.1 and 6.2 hold for the $\text{M}^{\lambda_1,\lambda_2}|\text{M}^{\mu}|s$ model as well. As a result, we can use the same state space as in the single-server case to fully describe the state of the system. Then, in effect, in some state (x, y), a total of $\mathbf{min}\{y, s\}$ servers will be serving a type-2 job, $\mathbf{min}\{x, \mathbf{max}\{0, s - y\}\}$ servers will be serving a type-1 job and $\mathbf{max}\{0, s - (x + y)\}$ servers will be idle. Note that these three numbers add up to s.

In view of what follows, we also note now that the extension of Lemma 6.1 to the multi-server case is analogous to the extension of Proposition 6.1.

6.4.1 Dynamic Programming formulation

We first note that the uniformization is such that $\lambda_1 + \lambda_2 + s\mu + \alpha = 1$. Now, extending the $\text{M}^{\lambda_1,\lambda_2}|\text{M}^{\mu}|1$ model to the $\text{M}^{\lambda_1,\lambda_2}|\text{M}^{\mu}|s$ model, the only DPEs that change are the ones for $V_n(x, y; \texttt{co})$. For $n \geq 1$, the original DPEs for $V_n(x, y; \texttt{co})$ are replaced by

$$
\begin{aligned}
V_n(x,y;\mathtt{co}) &= \textstyle\sum_{i=1}^{2} \lambda_i V_{n-1}(x,y,\mathrm{arr}/i) + \\
&\quad s\mu[V_{n-1}(x,y-1)+r_2] - (x+y)h \qquad x \geq 0,\ y \geq s \\
V_n(x,y;\mathtt{co}) &= \textstyle\sum_{i=1}^{2} \lambda_i V_{n-1}(x,y,\mathrm{arr}/i) + \\
&\quad y\mu[V_{n-1}(x,y-1)+r_2] + \\
&\quad (s-y)\mu[V_{n-1}(x-1,y)+r_1] - (x+y)h \quad 0 \leq y < s,\ x \geq s-y \\
V_n(x,y;\mathtt{co}) &= \textstyle\sum_{i=1}^{2} \lambda_i V_{n-1}(x,y,\mathrm{arr}/i) + \\
&\quad y\mu[V_{n-1}(x,y-1)+r_2] + \\
&\quad x\mu[V_{n-1}(x-1,y)+r_1] + \\
&\quad [s-(x+y)]\mu V_{n-1}(x,y) - (x+y)h \qquad x,y \geq 0,\ x+y < s
\end{aligned}
$$

6.4.2 Main result and its proof

We first make the following restrictive model assumption.

Assumption 6.1 Type-2 jobs do not incur any consideration costs (i.e., $c_2 = 0$) or may not be rejected upon arrival.

The reason for this assumption is contained in the proof of our main result, which is stated in Proposition 6.3 below. In that proof, we will need (6.7). But in Section 6.2.1 we have seen that (6.7) need not hold. In the proof of Proposition 6.3, we will show that (6.7) does hold under Assumption 6.1, which paves the way for the main result.

Proposition 6.3 *Under Assumption 6.1, the threshold characterization given by Parts 1a and 2 of Theorem 6.1 extends to the* $M^{\lambda_1,\lambda_2}|M^{\mu}|s$ *model.*

Proof. We can maintain the Key Proposition of the single-server model, and follow the lines of the proof given in Section 6.3. Note that it suffices to verify Step 2 of the proof, since only the DPEs for $V_n(x,y;\mathtt{co})$ have changed. In our proof of each of the three inequalities ($6.3^{\mathtt{co}}$), ($6.4^{\mathtt{co}}$) and ($6.5^{\mathtt{co}}$) for $n+1$, we will distinguish between several cases, where case (I) universally corresponds to $y \geq s$.

It is straightforward to check that for each of the inequalities $(6.3^{\,\mathsf{co}})$, $(6.4^{\,\mathsf{co}})$ and $(6.5^{\,\mathsf{co}})$, the derivation corresponding to case (I) is analogous to the derivation corresponding to case (I) in the single-server model. I.e., in the derivation of $(6.j^{\,\mathsf{co}})$ for $n+1$, $j=3,4,5$, we only need to apply $(6.j^{\,\mathrm{arr}/1})$, $(6.j^{\,\mathrm{arr}/2})$ and $(6.j)$, which hold on the basis of the induction hypothesis.

We will now check the three inequalities one by one for the remaining case of $y<s$. Let us first consider $(6.3^{\,\mathsf{co}})$. We distinguish the following three cases: (II) $0\le y<s$, $x\ge s-y$, (III) $x,y\ge 0$, $x+y<s-1$, and (IV) $x,y\ge 0$, $x+y=s-1$. For convenience, we will use the following notation: $\phi(x,y):=\sum_{i=1}^{2}\lambda_i[V_n(x+1,y,\mathrm{arr}/i)-V_n(x,y,\mathrm{arr}/i)]$. Furthermore, we will make use of the same symbols as in Chapter 2 (such as ① and ❶) to make compound arguments more comprehensible; see Section 2.3.2.

Case (II) gives

$$
\begin{aligned}
&V_{n+1}(x+1,y;\mathsf{co})-V_{n+1}(x,y;\mathsf{co})\\
&\quad=\ \phi(x,y)\,①\ +\ y\mu[V_n(x+1,y-1)-V_n(x,y-1)]\,②\ +\\
&\qquad\quad [s-(y+1)]\mu[V_n(x,y)-V_n(x-1,y)]\,③\ +\\
&\qquad\qquad \mu[V_n(x,y)-V_n(x-1,y)]\,④\ -\ h\\
&\quad\ge\ \{\mathsf{ih};\ ①\ge ❶\ \mathsf{by}\ (6.3^{\,\mathrm{arr}/1}),\ (6.3^{\,\mathrm{arr}/2});\\
&\qquad\quad ②\ge ❷,\ ③\ge ❸\ \mathsf{by}\ (6.3);\ ④\ge ❹\ \mathsf{by}\ (6.1)\}\\
&\quad=\ \phi(x,y+1)\,❶\ +\ y\mu[V_n(x+1,y)-V_n(x,y)]\,❷\ +\\
&\qquad\quad [s-(y+1)]\mu[V_n(x,y+1)-V_n(x-1,y+1)]\,❸\ +\\
&\qquad\qquad \mu[V_n(x+1,y)-V_n(x,y)]\,❹\ -\ h\\
&\quad=\ V_{n+1}(x+1,y+1;\mathsf{co})-V_{n+1}(x,y+1;\mathsf{co}).
\end{aligned}
$$

Case (III) gives

$$
\begin{aligned}
&V_{n+1}(x+1,y;\mathsf{co}) - V_{n+1}(x,y;\mathsf{co}) \\
&\quad = \quad \phi(x,y)\,① \;+\; y\mu[V_n(x+1,y-1) - V_n(x,y-1)]\,② + \\
&\qquad\quad x\mu[V_n(x,y) - V_n(x-1,y)]\,③ \;+\; \mu[V_n(x,y)+r_1]\,④ + \\
&\qquad\qquad [s-(x+y+2)]\mu[V_n(x+1,y) - V_n(x,y)]\,⑤ + \\
&\qquad\qquad\quad \mu[V_n(x+1,y) - V_n(x,y)]\,⑥ \;-\; \mu V_n(x,y)\,⑦ \;-\; h \\
&\quad \geq \quad \{\mathsf{ih};\ ① \geq ❶ \text{ by } (6.3^{\,\mathrm{arr}/1}),\ (6.3^{\,\mathrm{arr}/2});\\
&\qquad\quad ② + ⑥ \geq ❷,\ ③ \geq ❸,\ ⑤ \geq ❺ \text{ by } (6.3);\ ④ + ⑦ = ❹ + ❻\} \\
&\quad = \quad \phi(x,y+1)\,❶ \;+\; (y+1)\mu[V_n(x+1,y) - V_n(x,y)]\,❷ + \\
&\qquad\quad x\mu[V_n(x,y+1) - V_n(x-1,y+1)]\,❸ + \mu[V_n(x,y+1)+r_1]\,❹ + \\
&\qquad\qquad [s-(x+y+2)]\mu[V_n(x+1,y+1) - V_n(x,y+1)]\,❺ - \\
&\qquad\qquad\quad \mu V_n(x,y+1)\,❻ \;-\; h \\
&\quad = \quad V_{n+1}(x+1,y+1;\mathsf{co}) - V_{n+1}(x,y+1;\mathsf{co}).
\end{aligned}
$$

Case (IV) gives

$$
\begin{aligned}
&V_{n+1}(x+1,y;\mathsf{co}) - V_{n+1}(x,y;\mathsf{co}) \\
&\quad = \quad \phi(x,y)\,① \;+\; y\mu[V_n(x+1,y-1) - V_n(x,y-1)]\,② + \\
&\qquad\quad x\mu[V_n(x,y) - V_n(x-1,y)]\,③ \;+\; \mu[V_n(x,y)+r_1]\,④ - \\
&\qquad\qquad \mu V_n(x,y)\,⑤ \;-\; h \\
&\quad \geq \quad \{\mathsf{ih};\ ① \geq ❶ \text{ by } (6.3^{\,\mathrm{arr}/1}),\ (6.3^{\,\mathrm{arr}/2});\\
&\qquad\quad ② \geq ❷,\ ③ \geq ❸ \text{ by } (6.3);\ ④ + ⑤ \geq ❹ \text{ by Lemma 6.1}\} \\
&\quad = \quad \phi(x,y+1)\,❶ \;+\; y\mu[V_n(x+1,y) - V_n(x,y)]\,❷ + \\
&\qquad\quad x\mu[V_n(x,y+1) - V_n(x-1,y+1)]\,❸ + \\
&\qquad\qquad \mu[V_n(x+1,y) - V_n(x,y)]\,❹ \;-\; h \\
&\quad = \quad V_{n+1}(x+1,y+1;\mathsf{co}) - V_{n+1}(x,y+1;\mathsf{co}).
\end{aligned}
$$

Next, consider $(6.4^{\,\mathsf{co}})$. We distinguish the following two cases: (II) $0 \leq y < s$, $x \geq s-(y+1)$, and (III) $x,y \geq 0$, $x+y < s-1$.

Case (II) gives

$$\begin{aligned}
&V_{n+1}(x+1,y+1;\mathsf{co}) - V_{n+1}(x,y+1;\mathsf{co}) \\
&\quad = \phi(x,y+1)\,① \;+\; (y+1)\mu[V_n(x+1,y) - V_n(x,y)]\,② + \\
&\qquad [s-(y+1)]\mu[V_n(x,y+1) - V_n(x-1,y+1)]\,③ \;-\; h \\
&\quad \geq \{\mathsf{ih};\; ① \geq ❶ \text{ by } (6.4^{\,\mathrm{arr}/1}),\ (6.4^{\,\mathrm{arr}/2});\; ② \geq ❷ + ❹,\ ③ \geq ❸ \text{ by } (6.4)\} \\
&\quad = \phi(x+1,y)\,❶ \;+\; y\mu[V_n(x+2,y-1) - V_n(x+1,y-1)]\,❷ + \\
&\qquad [s-(y+1)]\mu[V_n(x+1,y) - V_n(x,y)]\,❸ + \\
&\qquad\quad \mu[V_n(x+1,y) - V_n(x,y)]\,❹ \;-\; h \\
&\quad = V_{n+1}(x+2,y;\mathsf{co}) - V_{n+1}(x+1,y;\mathsf{co}).
\end{aligned}$$

Case (III) gives

$$\begin{aligned}
&V_{n+1}(x+1,y+1;\mathsf{co}) - V_{n+1}(x,y+1;\mathsf{co}) \\
&\quad = \phi(x,y+1)\,① \;+\; (y+1)\mu[V_n(x+1,y) - V_n(x,y)]\,② + \\
&\qquad x\mu[V_n(x,y+1) - V_n(x-1,y+1)]\,③ + \mu[V_n(x,y+1) + r_1]\,④ + \\
&\qquad\quad [s-(x+y+2)]\mu[V_n(x+1,y+1) - V_n(x,y+1)]\,⑤ - \\
&\qquad\qquad \mu V_n(x,y+1)\,⑥ \;-\; h \\
&\quad \geq \{\mathsf{ih};\; ① \geq ❶ \text{ by } (6.4^{\,\mathrm{arr}/1}),\ (6.4^{\,\mathrm{arr}/2}); \\
&\qquad ② \geq ❷ + ❹,\ ③ \geq ❸,\ ⑤ \geq ❺ \text{ by } (6.4);\; ④ + ⑥ = ❻ + ❼\} \\
&\quad = \phi(x+1,y)\,❶ \;+\; y\mu[V_n(x+2,y-1) - V_n(x+1,y-1)]\,❷ + \\
&\qquad x\mu[V_n(x+1,y) - V_n(x,y)]\,❸ \;+\; \mu[V_n(x+1,y) - V_n(x,y)]\,❹ + \\
&\qquad\quad [s-(x+y+2)]\mu[V_n(x+2,y) - V_n(x+1,y)]\,❺ + \\
&\qquad\qquad \mu[V_n(x+1,y) + r_1]\,❻ \;-\; \mu V_n(x+1,y)\,❼ \;-\; h \\
&\quad = V_{n+1}(x+2,y;\mathsf{co}) - V_{n+1}(x+1,y;\mathsf{co}).
\end{aligned}$$

Finally, consider $(6.5^{\,\mathsf{co}})$. For this inequality, we distinguish the following three cases: (II) $x \geq 0$, $y = s-1$, (III) $0 \leq y < s-1$, $x \geq s-(y+1)$, and (IV) $x, y \geq 0$, $x + y < s-1$. For convenience, we will use the following notation: $\varphi(x,y) := \sum_{i=1}^{2} \lambda_i [V_n(x,y+1,\mathrm{arr}/i) - V_n(x,y,\mathrm{arr}/i)]$.

Case (II) gives

$$\begin{aligned}
&V_{n+1}(x+1,y+1;\mathsf{co}) - V_{n+1}(x+1,y;\mathsf{co})\\
&\quad= \varphi(x+1,s-1)\,① + (s-1)\mu[V_n(x+1,s-1) - V_n(x+1,s-2)]\,② +\\
&\qquad\quad \mu[V_n(x+1,s-1)+r_2]\,③ \;-\; \mu[V_n(x,s-1)+r_1]\,④ \;-\; h\\
&\quad\geq \{\mathsf{ih};\ ① \geq ❶ \text{ by } (6.5^{\,\mathrm{arr}/1}),\ (6.5^{\,\mathrm{arr}/2});\\
&\qquad\quad ② \geq ❷ \text{ by } (6.5);\ ③ + ④ \geq ❸ \text{ by Proposition 6.1}\}\\
&\quad= \varphi(x,s)\,❶ \;+\; (s-1)\mu[V_n(x,s) - V_n(x,s-1)]\,❷ +\\
&\qquad\quad \mu[V_n(x,s) - V_n(x,s-1)]\,❸ \;-\; h\\
&\quad= V_{n+1}(x,y+2;\mathsf{co}) - V_{n+1}(x,y+1;\mathsf{co}).
\end{aligned}$$

Case (III) gives

$$\begin{aligned}
&V_{n+1}(x+1,y+1;\mathsf{co}) - V_{n+1}(x+1,y;\mathsf{co})\\
&\quad= \varphi(x+1,y)\,① \;+\; y\mu[V_n(x+1,y) - V_n(x+1,y-1)]\,② +\\
&\qquad\quad [s-(y+2)]\mu[V_n(x,y+1) - V_n(x,y)]\,③ +\\
&\qquad\qquad \mu[V_n(x+1,y)+r_2]\,④ \;-\; \mu[V_n(x,y)+r_1]\,⑤ +\\
&\qquad\qquad\quad \mu[V_n(x,y+1) - V_n(x,y)]\,⑥ \;-\; h\\
&\quad\geq \{\mathsf{ih};\ ① \geq ❶ \text{ by } (6.5^{\,\mathrm{arr}/1}),\ (6.5^{\,\mathrm{arr}/2});\\
&\qquad\quad ② + ⑥ \geq ❷,\ ③ \geq ❸ \text{ by } (6.5);\ ④ + ⑤ \geq ❹ + ❺ \text{ by assuming } (6.7)\}\\
&\quad= \varphi(x,y+1)\,❶ \;+\; (y+1)\mu[V_n(x,y+1) - V_n(x,y)]\,❷ +\\
&\qquad\quad [s-(y+2)]\mu[V_n(x-1,y+2) - V_n(x-1,y+1)]\,❸ +\\
&\qquad\qquad \mu[V_n(x,y+1)+r_2]\,❹ \;-\; \mu[V_n(x-1,y+1)+r_1]\,❺ \;-\; h\\
&\quad= V_{n+1}(x,y+2;\mathsf{co}) - V_{n+1}(x,y+1;\mathsf{co}).
\end{aligned}$$

Case (IV) gives

$$
\begin{aligned}
&V_{n+1}(x+1,y+1;\mathsf{co}) - V_{n+1}(x+1,y;\mathsf{co}) \\
&\quad = \quad \varphi(x+1,y)\,① \;+\; y\mu[V_n(x+1,y) - V_n(x+1,y-1)]\,② + \\
&\qquad\quad x\mu[V_n(x,y+1) - V_n(x,y)]\,③ + \\
&\qquad\qquad [s-(x+y+2)]\mu[V_n(x+1,y+1) - V_n(x+1,y)]\,④ + \\
&\qquad\qquad\quad \mu[V_n(x+1,y) + r_2]\,⑤ \;-\; \mu V_n(x+1,y)\,⑥ + \\
&\qquad\qquad\qquad \mu[V_n(x,y+1) - V_n(x,y)]\,⑦ \;-\; h \\
&\quad \geq \quad \{\mathsf{ih};\;\; ① \geq ❶ \text{ by } (6.5^{\,\mathrm{arr}/1}),\ (6.5^{\,\mathrm{arr}/2}); \\
&\qquad\quad ② + ⑦ \geq ❷,\ ③ \geq ❸,\ ④ \geq ❹ \text{ by } (6.5);\;\; ⑤ + ⑥ = ❺ + ❻\} \\
&\quad = \quad \varphi(x,y+1)\,❶ \;+\; (y+1)\mu[V_n(x,y+1) - V_n(x,y)]\,❷ + \\
&\qquad\quad x\mu[V_n(x-1,y+2) - V_n(x-1,y+1)]\,❸ + \\
&\qquad\qquad [s-(x+y+2)]\mu[V_n(x,y+2) - V_n(x,y+1)]\,❹ + \\
&\qquad\qquad\quad \mu[V_n(x,y+1) + r_2]\,❺ \;-\; \mu V_n(x,y+1)\,❻ \;-\; h \\
&\quad = \quad V_{n+1}(x,y+2;\mathsf{co}) - V_{n+1}(x,y+1;\mathsf{co}).
\end{aligned}
$$

In the second last derivation, we assumed (6.7). Therefore, it remains to prove that (6.7) holds under Assumption 6.1. We first add (6.7) to the Key Proposition and define $(6.7^{\,\mathrm{arr}/1})$, $(6.7^{\,\mathrm{arr}/2})$ and $(6.7^{\,\mathsf{co}})$, analogous to the definition of $(6.j^{\,\mathrm{arr}/1})$, $(6.j^{\,\mathrm{arr}/2})$ and $(6.j^{\,\mathsf{co}})$ for $1 \leq j \leq 6$. Next, we observe that inequality (6.7) holds by definition for $n = 0$ (*Step 0*), and we add it to the induction hypothesis. It now suffices to establish $(6.7^{\,\mathrm{arr}/1})$ and $(6.7^{\,\mathrm{arr}/2})$ for n (*Step 1*), $(6.7^{\,\mathsf{co}})$ for $n+1$ (*Step 2*), and finally (6.7) for $n+1$ (*Step 3*).

Step 1 is analogous to Step 1 for $(6.4^{\,\mathrm{arr}/1})$ and $(6.4^{\,\mathrm{arr}/2})$. See Table 6.3 for details. Note that because of Assumption 6.1, type-2 jobs are never rejected and hence cases $\mathcal{RA}$, $\mathcal{AR}$ and $\mathcal{RR}$ do not exist for $(6.7^{\,\mathrm{arr}/2})$.

inequality		case		result by
$(6.7^{\,\mathrm{arr}/1})$	ac	$\mathcal{AA}$	ac	induction hypothesis; (6.7)
	rj	$\mathcal{AR}$	ac	lhs=rhs=c
	rj	$\mathcal{RA}$	ac	induction hypothesis; (6.3)
	rj	$\mathcal{RR}$	rj	induction hypothesis; (6.7)
$(6.7^{\,\mathrm{arr}/2})$	ac	$\mathcal{AA}$	ac	induction hypothesis; (6.7)

Table 6.3: Case distinction for inequalities $(6.7^{\,\mathrm{arr}/1})$ and $(6.7^{\,\mathrm{arr}/2})$

Step 2 is analogous to Step 2 for (6.4^{co}). In fact, distinguishing the same three cases, the derivations are identical to the derivations for (6.4^{co}), except that the inequality signs are read the other way around, i.e., all '$\geq$'-signs are replaced by '$\leq$'-signs. These hold on the basis of the induction hypothesis and inequalities ($6.7^{\mathrm{arr}/1}$), ($6.7^{\mathrm{arr}/2}$) and (6.7).

Finally, Step 3 is analogous to Step 3 for (6.4). See Table 6.4 for details.

For all $x, y \geq 0$, $0 \leq k \leq x+1$, and $0 < m \leq y$ (provided $y > 0$):

inequality		case		result by
	ab	$X\mathcal{A}$		lhs=rhs=0
(6.7)	$\mathsf{ab}^{\,k}$	$\mathcal{A}^k\mathcal{C}$	co	ih; (6.1^{co}) k times; (6.7^{co})
	$\mathsf{ab}^{\,x+1+m}$	$\mathcal{A}^{x+1+m}\mathcal{C}$	co	ih; (6.6^{co}); (6.3^{co}) m times

Table 6.4: Analysis of inequality (6.7) for $n+1$

Table 6.4 concludes our proof of (6.7), and hence our proof of Proposition 6.3 is complete.

□

Remark 6.4 In the derivation corresponding to case (III) for (6.5^{co}), we used (6.7). We did not see any possibility to establish (6.5^{co}) without the use of (6.7). This explains the introduction of Assumption 6.1. However, it remains unclear whether Assumption 6.1 is actually a necessary condition for the extension of the threshold characterization to the multi-server model, as given by Proposition 6.3.

6.5 Conclusions

We have considered a $\mathrm{M}^{\lambda_1,\lambda_2}|\mathrm{M}^{\mu}|1$ preemptive priority queue. For this queue we have dealt with two additional decision features. First, one has to decide upon arrival of a job to accept or reject the new job. Second, at any point in time, one may decide to remove any number of jobs from the system. We have shown that the optimal strategy for both types of decisions is characterized by threshold policies. Under the assumption that type-2 jobs will always be admitted to the system, we have extended our results to the general multi-server case.

Two other extensions of our basic two-class model that come to mind naturally are (1) the extension to more than 2 classes of jobs, and (2) the extension to more general service time distributions, e.g., Erlang distributed service times. However, both extensions will require a higher-dimensional state space, and additional monotonicity properties have to be formulated and established to obtain a monotonic characterization of the optimal policy via inductive DP. This is also the case if we want to consider a non-preemptive instead of preemptive resume service discipline, although the analysis of that particular model is probably somewhat easier, because the third dimension features only two states: '1' if the server is serving a type-1 job, and '2' if the server is serving a type-2 job.

These models are left for future research. To give an impression of what monotonicity properties would be required, we outline below some results one would aim for in the context of the multi-class extension of the $M^{\lambda_1,\lambda_2}|M^{\mu}|1$ model, which we term the $M^{\lambda_j}|M^{\mu}|1$ model.

Let $J \geq 2$ denote the number of classes of jobs. Type-i jobs, $1 \leq i \leq J$, arrive at the station according to a Poisson process with arrival rate $\lambda_i \geq 0$. The workload of a job is exponential with class-independent mean service time $1/\mu$. The reward for a type-i job is r_i, where $r_J > r_{J-1} > \cdots > r_2 > r_1 > 0$. The holding costs are class-independent and equal to $mh \geq 0$ per unit of time when there are m jobs present, and the consideration costs are $c_i \geq 0$ per type-i job admitted to the system.

The state of the system is described by the vector $\underline{z} = (z_1, \ldots, z_J)$, where $z_i \geq 0$ (with $1 \leq i \leq J$) denotes the number of type-i jobs in the system. Analogous to the two-class model, we define the value function $V_n(\underline{z})$. If the current state is $\underline{z}$ and a type-i job is aborted, then there is an instant transition to state $\underline{z} - \underline{e}_i$.

Then, in generalization of Theorem 6.1, one may for example wish to show that if it is optimal to abort in some state $\underline{z}$, then it is optimal as well to abort in all states $\underline{z} + \underline{e}_i$ with $1 \leq i \leq J$ and in all states $\underline{z} + \underline{e}_i - \underline{e}_j$ with $1 \leq i < j \leq J$. This monotonic characterization of the optimal termination policy would be a corollary of the following set of inequalities:

$$V_n(\underline{z} + \underline{e}_i) - V_n(\underline{z}) \geq V_n(\underline{z} + \underline{e}_i + \underline{e}_k) - V_n(\underline{z} + \underline{e}_k), \qquad (6.16)$$

$$V_n(\underline{z} + \underline{e}_j + \underline{e}_k) - V_n(\underline{z} + \underline{e}_j) \geq V_n(\underline{z} + \underline{e}_i + \underline{e}_k) - V_n(\underline{z} + \underline{e}_i), \qquad (6.17)$$

where $n \geq 0$, $\underline{z} \geq \underline{0}$ and $1 \leq k \leq i \leq j \leq J$. Note that (6.16) is the natural generalization of (6.3) and that (6.17) is the natural generalization of (6.4).

To be able to make an inductive proof work, this set of inequalities will have to be supplemented with other inequalities. Moreover, it may turn out that additional model assumptions are required to be able to establish the complete set of inequalities, e.g., analogous to the multi-server extension, it might be necessary to assume that $c_i = 0$ for all $1 \leq i \leq J$.

7

Conclusions and outlook

MOTIVATED BY lack-of-capacity problems in workflow environments, we have constructed and studied a collection of workload models with admission as well as termination control. The objective has been two-fold. Our first goal has been to provide an impetus to the quantitative analysis of workflow processes, in particular with respect to on-line decision-making involving partial execution and premature termination of work. Our second goal has been to extend the literature on the dynamic control of queueing systems, by introducing and studying a new type of control and by focusing on the derivation of monotonicity properties of the optimal control policy.

We would like to conclude this dissertation with a brief discussion of some natural directions for further research, where we distinguish between on the one hand research aimed specifically at the analysis of workflows and on the other hand research in the area of the derivation of structural results for the dynamic control of queueing systems.

7.1 Design and control of workflow processes

By far the most popular framework for the analysis of workflow processes is Petri net theory. Petri nets can be used to analyse qualitative properties, and are also commonly used as system specification tools. We refer to Van der Aalst [1] for a more detailed discussion of the application of Petri nets to workflows. Depending on the level of planning, Petri net theory can also support quantitative performance optimization. Once a feasible workflow

has been constructed by means of Petri nets, simulation can be used as a means to investigate its behaviour in terms of performance measures such as the average throughput time of process instances. Consider, for example, the one-time hiring of resources, where the question is how many resources to hire. If the structure of the workflow and the operational control rules are fixed, then one may vary the number of resources and conduct simulations until, as a result of trial-and-error, a satisfactory or possibly the optimal number of resources has been determined. If the focus is not on one-time decision-making, but on dynamic decision-making, based on the state of the process, then the method of simulation is far less useful. Namely, a major drawback of simulation is the fact that one can consider only one set of operational control rules at a time. This means that, when in search of an optimal set of control rules, one has to construct, simulate and compare all possible schedules of states and decisions to be taken in those states, of which there are in general (virtually) infinitely many. There remains a clear challenge and need to construct a framework for quantitative performance optimization of workflows, which can support the design of new process definitions as well as the on-line management of existing processes.

Two of the very few papers studying quantitative design rules for workflow processes are Buzacott [14] and Van der Aalst [2]. These are also mentioned in Reijers [45], who lists a number of general (re)design rules. In these studies, the focus is exclusively on *tactical*, static design aspects. In contrast, in our thesis, we have focused on *operational*, dynamic decision-making in workflow environments, which comes into play once a workflow definition has been constructed. An example of a tactical decision is the decision to pool the available resources. The operational decision-making then involves the allocation of resources to activities in real time, where it may possibly be allowed to let several resources work together on a case. This leads to dynamic work-sharing. Another example is the subcontracting of intricate cases to third-party specialists. The operational decision-making then involves the question when to subcontract and when to carry out the examination or investigation on one's own. For small-scale problems, it will be possible to build a decision model and to compute an optimal set of rules. However, given the size and complexity of real-life workflow definitions, it is more useful to focus on the construction of good heuristics. Nevertheless, in that process, a good understanding of the performance of more basic models may be very helpful.

7.2 Structural results for the dynamic control of queueing systems

7.2.1 A general framework for the derivation of structural results

For the models considered in this thesis, we employed a general analytic approach—inductive Dynamic Programming—to obtain monotonicity properties of the value function and the optimal policy. The models themselves, however, were not general, as they involved *specific* control models. This is in line with the vast majority of literature on the dynamic control of queueing systems; cf. the references in Sections 1.4 to 1.7. First efforts to construct a general framework for the derivation of monotonicity properties using inductive DP were made by Glasserman and Yao [21], and Altman and Koole [5]. Embroidering on this theme, Koole [34] proposed a unified treatment of one-dimensional and two-dimensional models, which he termed *event-based dynamic programming*. This is a systematic approach that focuses on system events and the form of the value function, rather than on the value function itself. Possible events are captured by *event operators*. From these operators, specific models can be built. First for the one-dimensional case and subsequently for the two-dimensional case, Koole considers a collection of common operators, such as 'a departure from the system'. He shows for each operator separately that it satisfies certain monotonicity properties. If operators have certain monotonicity properties in common, then the value function of a specific model built from those operators will satisfy these monotonicity properties as well, as is shown. The operators discussed by Koole cover many specific models studied in literature, e.g., those of Lippman [40], discussed in Section 1.7.4.1.

An interesting topic for further research would be to investigate if and to what extent the notion of termination control and the various models with admission and termination control we have studied in this thesis fit within this general framework. For example, for some value function $V(i)$, where i is the number of jobs in the system, Koole introduced the operator T_{AC}, defined by $T_{AC}V(i) := \mathbf{min}\{V(i)+c,\, V(i+1)+c'\}$. T_{AC} models admission control, where rejection incurs costs c, admission incurs costs c' and the objective is to minimize costs. In the same fashion, we could introduce an operator T_{TC} which models termination control. For example, regarding the M|E$_N$|1 model studied in Chapter 3, we could write

$$T_{TC}V(i,k) := \mathbf{max}\{V(i,k),\, \max_{1\le j\le i} V(i-j,0) + r(k) + (j-1)r(0)\}.$$

7.2.2 Higher-dimensional state space models

In the final section of each of the Chapters 2 to 6, we made some suggestions for further research. Some of the models we proposed there concern models whose natural description involves a state space of some dimension larger than 2. An example is the multi-class extension of the $\mathrm{M}^{\lambda_1,\lambda_2}|\mathrm{M}^{\mu}|1$ model studied in Chapter 6. To be able to obtain a monotonic characterization of the optimal control policy for this $\mathrm{M}^{\lambda_j}|\mathrm{M}^{\mu}|1$ model via inductive DP, monotonicity properties beyond submodularity and subconcavity are required. We touched briefly upon this subject in Section 1.7.4.2. In the context of higher-dimensional tandem queues, Koole [35] uses the notion of *directional convexity*. Following Shaked and Shanthikumar [51], a function $f(\underline{x})$ defined on $\mathbb{N}^m$ is said to be *directionally convex* if

$$f(\underline{x}+\underline{z}) - f(\underline{x}) \quad \leq \quad f(\underline{y}+\underline{z}) - f(\underline{y}) \tag{7.1}$$

for all $\underline{x}$, $\underline{y}$, $\underline{z} \in \mathbb{N}^m$, where $x_i \leq y_i$ for all $1 \leq i \leq m$. Note that (7.1) is a sufficient condition for (6.16), which we formulated in the context of the $\mathrm{M}^{\lambda_j}|\mathrm{M}^{\mu}|1$ model, where the inequality sign is read the other way around, because we consider the objective of maximizing rewards, whereas Koole considers the objective of minimizing costs.

We would like to mention that the higher-dimensional tandem queues studied by Koole can be very useful in the context of work-sharing problems; cf. Section 7.1. If each case has to pass through a fixed number of consecutive stages, corresponding to tasks 1 through N say, where each task may be executed by a different resource, then this can be modelled by means of N tandem queues. Suppose the resources can be classified into resource classes 1 through R, where class $r \in \{1,\ldots,R\}$ is cross-trained to carry out the set of tasks $S(r) \subset \{1,\ldots,N\}$. The dynamic work-sharing control problem then involves the question where all these flexible resources will be put to work. For example, consider a process consisting of 3 stages and 2 resources, where $S(1) = \{1,2\}$ and $S(2) = \{2,3\}$. This gives rise to a dynamic control model with a three-dimensional state space, featuring states of the form (i,j,k), where i, j and k represent the number of work items at stage 1, 2 and 3, respectively. The action set may be written as $A(i,j,k) = \{(a_1,a_2) \mid a_1 \in \{1,2\}, a_2 \in \{2,3\}\}$ for all $i,j,k \geq 0$, where for both resources we have that working at stage m corresponds to idling if there is no work item available for that resource at that particular stage. If work items can only be processed by one resource at a time, then decision $(2,2)$ corresponds to the decision to process two work items simultaneously

at stage 2, unless there is only 1 work item available, in which case one of the two resources will idle. Righter et al. [47] report some results for a particular version of this two-server three-task model, but there remain many interesting open problems with respect to the characterization of the optimal policy for her model as well as alternative versions and generalizations of this model.

If the number of stages and servers becomes larger, the computation of the optimal control policy will become intractable or even impossible; cf. our study of the $\mathrm{M}|\mathrm{E}_N|s$ model in Chapter 5. In this case, we are interested in heuristics that achieve good or possibly even near-optimal performance. The development of such heuristics for work-sharing models is another natural and useful direction for future research.

Clearly, the search for good approximations is not limited to tandem queues, and the concept of work-sharing may be employed in other or more general process environments as well. Besides processes in which tasks are executed *sequentially*, modelled by means of a series of tandem queues, one can imagine that case investigations may involve the execution of certain tasks *in parallel*. For example, a case may consist of J independent tasks, which may be executed simultaneously. The case is considered to be dealt with to completion only if all of the J tasks have been executed for this case. Such a situation can be modelled by means of a so-called fork-join queue. Fork-join processes give rise to and include processes that feature *enforced waiting*. Included are, for example, processes in which waiting for external information is a key factor in the throughput time of case instances. More complex models involve multiple (compound) forks and joins, and networks containing sequential as well as parallel task execution elements.

In this thesis, we have taken first steps in the construction of a framework for the modelling and analysis of dynamic control problems that are present in deficient-capacity operating environments. Concerning the derivation of analytical (structural) results in this area, our study opens up numerous extensions and generalizations, and related models and results, and we have highlighted some in this final chapter. Moreover, we have acknowledged the development of good heuristics for complex control problems as a topic of particular interest. This topic remains vastly open to future research.

References

[1] Aalst, W.M.P. van der, *The application of Petri nets to workflow management*, *Journal of Circuits, Systems, and Computers* **8**, 1998, 21–66.

[2] Aalst, W.M.P. van der, *Reengineering knock-out processes*, *Decision Support Systems* **30**, 2001, 451–468.

[3] Aalst, W.M.P. van der and K.M. van Hee, *Workflow Management: Models, Methods, and Systems*, MIT Press, 2002.

[4] Alanyali, M. and B. Hajek, *On load balancing in Erlang networks*, *Stochastic Networks: Theory and Applications, ed. by F.P. Kelly, S. Zachary and I. Ziedins*, Oxford University Press, 1996.

[5] Altman, E. and G.M. Koole, *On submodular value functions and complex dynamic programming*, *Comm. Statist. Stochastic Models* **14**, 1998, 1051–1072.

[6] Bellman, R., *Dynamic Programming*, Princeton University Press, 1957.

[7] Bellman, R., *Adaptive Control Processes: A Guided Tour*, Princeton University Press, 1961.

[8] Bertsekas, D.P., *Dynamic Programming—Deterministic and Stochastic Models*, Prentice-Hall, 1987.

[9] Brouns, G.A.J.F., *Featuring Workflow Management—An overview of the distinctive features of workflow processes and their consequences for workflow management*, SPOR-Report 2000-05, Eindhoven University of Technology, 2000.

[10] Brouns, G.A.J.F. and J. van der Wal, *Optimal threshold policies in a workload model with a variable number of service phases per job*, SPOR-Report 2000-20, Eindhoven University of Technology, 2000; will appear in *Math. Methods Oper. Res.* **58**, 2003.

[11] Brouns, G.A.J.F. and J. van der Wal, *Monotonicity and threshold properties in batch workload models with controlled service times consisting of a sum of geometric phases*, SPOR-Report 2001-04, Eindhoven University of Technology, 2001; an excerpt has appeared in *Proc. 4th Aegean Intl. Conf. on Analysis of Manufacturing Systems*, Samos Island, Greece, 2003.

[12] Brouns, G.A.J.F. and J. van der Wal, *Optimal threshold policies in a two-class preemptive priority queue with admission and termination control*, SPOR-Report 2002-16, Eindhoven University of Technology, 2002; to appear in *Queueing Syst. Theory Appl.*

[13] Brouns, G.A.J.F., *Optimal control of routing to two parallel finite capacity queues or two parallel Erlang loss systems with dedicated and flexible arrivals*, SPOR-Report 2003-03, Eindhoven University of Technology, 2003; submitted for publication.

[14] Buzacott, J.A., *Commonalities in reengineered business processes: Models and issues*, *Management Sci.* **42**, 1996, 768–782.

[15] Cohen, J.W., *The Single Server Queue*, 2nd ed., North-Holland, 1982.

[16] Crabill, T.B., D. Gross and M.J. Magazine, *A classified bibliography of research on optimal design and control of queues*, *Oper. Res.* **25**, 1977, 219–232.

[17] Davis, E., *Optimal control of arrivals to a two-server queueing system with separate queues*, PhD thesis, North Carolina State University, Raleigh, North Carolina, 1977.

[18] Denardo, E.V., *A Markov decision problem*, *Mathematical Programming (Proc. Adv. Sem., Univ. Wisconsin, 1972), ed. by T. Hu and S. Robinson*, Academic Press, 1973, 33–68.

[19] El-Taha, M. and S. Stidham Jr., *Sample-path Analysis of Queueing Systems*, International Series in Operations Research & Management Science, 11, Kluwer Academic Publishers, 1999.

[20] Ghoneim, H.A. and S. Stidham Jr., *Control of arrivals to two queues in series*, *European J. Oper. Res.* **21**, 1985, 399–409.

[21] Glasserman, P. and D.D. Yao, *Monotone optimal control of permutable GSMPs*, *Math. Oper. Res.* **19**, 1994, 449–476.

[22] Groenevelt, R., G.M. Koole and P. Nain, *On the bias vector of a two-class preemptive priority queue*, *Math. Methods Oper. Res.* **55**, 2002, 107–120.

[23] Hajek, B., *Optimal control of two interacting service stations*, *IEEE Trans. Automat. Control* **29**, 1984, 491–499.

[24] Hariharan, R., V.G. Kulkarni and S. Stidham Jr., *A survey of research relevant to virtual-circuit routing in telecommunication networks*, Technical Report, University of North Carolina, Chapel Hill, 1990.

[25] Harrison, J.M., *A priority queue with discounted linear costs*, *Oper. Res.* **23**, 1975, 260–269.

[26] Hordijk, A. and G.M. Koole, *On the optimality of the generalized shortest queue policy*, *Probab. Engrg. Inform. Sci.* **4**, 1990, 477–487.

[27] Howard, R.A., *Dynamic Programming and Markov Processes*, The Technology Press of M.I.T., Wiley, 1960.

[28] Johansen, S.G. and C. Larsen, *Computation of a near-optimal service policy for a single-server queue with homogeneous jobs*, *European J. Oper. Res.* **134**, 2001, 648–663.

[29] Johri, P.K., *Optimality of the shortest line discipline with state-dependent service times*, *European J. Oper. Res.* **41**, 1989, 157–161.

[30] Kakalik, J.S., *Optimal dynamic operating policies for a service facility*, Technical Report No. 47, Operations Research Center, Massachusetts Institute of Technology, 1969.

[31] Kendall, D.G., *Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain*, *Ann. Math. Statist.* **24**, 1953, 338–354.

[32] Kleinrock, L., *Queueing Systems—Volume I: Theory*, Wiley, 1975.

[33] Koole, G.M., *Stochastic Scheduling and Dynamic Programming*, CWI Tract 113, CWI, Amsterdam, 1995.

[34] Koole, G.M., *Structural results for the control of queueing systems using event-based dynamic programming*, *Queueing Syst. Theory Appl.* **30**, 1998, 323–339.

[35] Koole, G.M., *Convexity in tandem queues*, Report WS-516, Vrije Universiteit Amsterdam, 1999.

[36] Koole, G.M., *On the static assignment to parallel servers*, *IEEE Trans. Automat. Control* **44**, 1999, 1588–1592.

[37] Koole, G.M., P.D. Sparaggis and D. Towsley, *Minimizing response times and queue lengths in systems of parallel queues*, *J. Appl. Probab.* **36**, 1999, 1185–1193.

[38] Lawrence, P., editor, *Workflow Handbook 1997*, Wiley, 1997.

[39] Leeuwaarden, J. van, S. Aalto and J. Virtamo, *Load balancing in cellular networks using first policy iteration*, Technical Report, Helsinki University of Technology, 2001.

[40] Lippman, S.A., *Applying a new device in the optimization of exponential queueing systems*, *Oper. Res.* **23**, 1975, 687–710.

[41] Liu, Z., P. Nain and D. Towsley, *Sample path methods in the control of queues*, *Queueing Syst. Theory Appl.* **21**, 1995, 293–335.

[42] Menich, R. and R.F. Serfozo, *Monotonicity and optimality of symmetric parallel processing systems*, *Queueing Syst. Theory Appl.* **9**, 1991, 403–418.

[43] Naor, P., *On the regulation of queue size by levying tolls*, *Econometrica* **37**, 1969, 15–24.

[44] Puterman, M.L., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, 1994.

[45] Reijers, H.A., *Design and Control of Workflow Processes: Business Process Management for the Service Industry*, PhD thesis, Eindhoven University of Technology, 2002.

[46] Righter, R., *Expulsion and scheduling control for multiclass queues with heterogeneous servers*, *Queueing Syst. Theory Appl.* **34**, 2000, 289–300.

[47] Righter, R. *et al.*, *Dynamic load balancing with flexible workers*, working paper, 2003.

[48] Ross, S.M., *Introduction to Stochastic Dynamic Programming*, Academic Press, 1983.

[49] Ross, S.M., *Stochastic Processes*, 2nd ed., Wiley, 1996.

[50] Serfozo, R.F., *An equivalence between continuous and discrete time Markov decision processes*, *Oper. Res.* **27**, 1979, 616–620.

[51] Shaked, M. and J.G. Shanthikumar, *Parametric stochastic convexity and concavity of stochastic processes*, *Ann. Inst. Statist. Math.* **42**, 1990, 509–531.

[52] Stidham Jr., S., *Optimal control of admission to a queueing system*, *IEEE Trans. Automat. Control* **30**, 1985, 705–713.

[53] Stidham Jr., S. and R.R. Weber, *A survey of Markov decision models for control of networks of queues*, *Queueing Syst. Theory Appl.* **13**, 1993, 291–314.

[54] Takagi, H., *Queueing Analysis: A Foundation of Performance Evaluation*, 3 volumes, North-Holland, 1991, 1993, 1993.

[55] Teghem Jr., J., *Control of the service process in a queueing system*, *European J. Oper. Res.* **23**, 1986, 141–158.

[56] Topkis, D.M., *Minimizing a submodular function on a lattice*, *Oper. Res.* **26**, 1978, 305–321.

[57] Towsley, D., P.D. Sparaggis and C.G. Cassandras, *Optimal routing and buffer allocation for a class of finite capacity queueing systems*, *IEEE Trans. Automat. Control* **37**, 1992, 1446–1451.

[58] Walrand, J., *An Introduction to Queueing Networks*, Prentice-Hall, 1988.

[59] WfMC, Workflow Management Coalition, *The Workflow Reference Model*, Technical Report TC00-1003, 1995.

[60] WfMC, Workflow Management Coalition, *Terminology & Glossary*, Technical Report TC00-1011, 1999.

[61] Whitt, W., *Deciding which queue to join: Some counterexamples*, *Oper. Res.* **34**, 1986, 55–62.

[62] Winston, W., *Optimality of the shortest line discipline*, *J. Appl. Probab.* **14**, 1977, 181–189.

[63] Xu, S.H. and J.G. Shanthikumar, *Optimal expulsion control—A dual approach to admission control of an ordered-entry system*, *Oper. Res.* **41**, 1993, 1137–1152.

[64] Xu, S.H., *A duality approach to admission and scheduling controls of queues*, *Queueing Syst. Theory Appl.* **18**, 1994, 273–300.

Samenvatting

Dienstverlenende ondernemingen en instanties hebben veelal—en bovendien in toenemende mate—te maken met een overvloedige en onvermijdbare toestroom van werk. Ondanks een scala aan mogelijke maatregelen, zoals de invoering van overwerk, tijdelijk werk en meervoudige inzetbaarheid van personeel, is er structureel onvoldoende capaciteit om alle aangeboden werk tot in detail te kunnen afhandelen. Derhalve zijn tijd en capaciteit kostbaar. Er zullen voortdurend beslissingen genomen moeten worden ten aanzien van welke opdrachten uitgevoerd worden en welke niet, en hoe de beschikbare capaciteit over de verschillende uit te voeren taken verdeeld zal worden. Dit zal afhangen van de werkdruk enerzijds en het voordeel dat gepaard gaat met het voortzetten van de afhandeling van in behandeling zijnde opdrachten anderzijds. Afhankelijk van de hoeveelheid werk in portefeuille kan besloten worden meer of juist minder aandacht te besteden aan de met de diverse opdrachten corresponderende taken. Dit geeft aanleiding tot een gedeeltelijke (of *partiële*) verwerking van opdrachten, met als doel om een optimale verhouding te creëren tussen de factoren doorlooptijd en behaalde kwaliteit, oftewel *Quality of Service*. Hierbij is doorlooptijd een synoniem voor gemaakte kosten, en is Quality of Service een synoniem voor opbrengst.

In dit proefschrift richten we ons specifiek op twee essentiële dynamische beslissingen die uit noodzaak genomen dienen te worden in werklastsystemen waarin de capaciteit ontoereikend is om alle aangeboden werk tot voltooiing te kunnen brengen:

(I) acceptatie (*admission*) of afwijzing (*rejection*) van nieuw werk, en
(II) voortzetting (*continuation*) of beëindiging (*termination*) van werk in portefeuille.

De aan beslissing (I) ten grondslag liggende besturing staat in de literatuur algemeen bekend onder de naam *admission control*. Beslissing (II) betreft een nieuw type besturing, dat we zullen aanduiden met de term *termination*

control. Dit type besturing is veel dynamischer en flexibeler dan de typen besturing die het voortijdig beëindigen van werk toestaan die tot nu toe in de literatuur bestudeerd zijn. Met de introductie van dit concept geven we de eerste aanzet tot een nieuwe klasse van beslissingsmodellen, die op een natuurlijke wijze binnen het onderzoeksgebied van de dynamische besturing van wachtrijen valt.

In onze analyse van het concept termination control ligt de nadruk op het bestuderen en karakteriseren van de optimale strategie met betrekking tot de twee voornoemde dynamische beslissingselementen. Hierbij maken we veelvuldig gebruik van een combinatie van twee bekende technieken, te weten Dynamische Programmering (DP) en volledige inductie. In het bijzonder wenden we ons tot deze methode om zekere monotonie-eigenschappen van de waardefunctie van het beschouwde model te bewijzen. Gezamenlijk impliceren deze eigenschappen vervolgens bepaalde monotonie-eigenschappen van de optimale strategie voor het model, bijvoorbeeld een drempelwaarde-structuur.

In Hoofdstuk 2 bestuderen we modellen met admission control, maar nog zonder termination control. We beschouwen twee nauw verwante systemen, die beide bestaan uit twee parallelle deelsystemen waarnaar aankomende opdrachten gerouteerd moeten worden. Er zijn twee categorieën opdrachten. De eerste betreft zogeheten 'flexibele' opdrachten. Deze kunnen door elk der beide deelsystemen uitgevoerd worden. De tweede betreft zogeheten 'aangewezen' opdrachten. Er zijn twee soorten aangewezen opdrachten, namelijk opdrachten die alleen door het ene deelsysteem uitgevoerd kunnen worden en opdrachten die alleen door het andere deelsysteem uitgevoerd kunnen worden. Beide systemen hebben eindige bufferruimte, en zijn dus onderhevig aan blokkering. Gegeven de doelstelling het aantal geblokkeerde opdrachten te minimaliseren, laten we zien dat in beide modellen de optimale routeringsstrategie een drempelwaarde-structuur heeft.

In Hoofdstuk 3 breiden we de beslissingsstructuur uit met termination control. We introduceren het concept termination control in de context van een M|E$_N$|1 wachtrijsysteem waarin de executie van een opdracht op elk moment beëindigd mag worden en waarin tevens op elke moment een willekeurig aantal opdrachten uit de wachtrij verwijderd mag worden. Onder zekere regulariteitsaannamen ten aanzien van de kosten- en opbrengststructuur leiden we een reeks monotonie-eigenschappen van de waardefunctie af, en laten we zien dat er optimale drempelwaarde-strategieën bestaan voor zowel de

beslissing om een opdracht te accepteren of te weigeren, als de beslissing om de afhandeling van een opdracht te continueren of juist te termineren.

In aansluiting hierop bespreken we in Hoofdstuk 4 een aantal uitbreidingen van het in Hoofdstuk 3 bestudeerde basismodel. Onder deze uitbreidingen bevinden zich groepsaankomsten, fase-type aankomsten, en een algemener bedieningsproces, dat gekarakteriseerd wordt door een Markov *feed-forward* routeringsmechanisme. Met behulp van DP leiden we gegeneraliseerde monotonie- en drempelwaarde-resultaten af voor elk van deze uitbreidingen. Daarnaast karakteriseren we de structuur van de optimale strategie voor een zuiver discrete versie van het model met feed-forward routering. In dit model bestaat de werklast van een opdracht uit de som van ten hoogste N geometrische bedieningsfasen en wordt de toestand van het systeem op deterministische beslissingstijdstippen geïnspecteerd.

Hoofdstuk 5 staat in het teken van een natuurlijke *multi-server* uitbreiding van het model uit Hoofdstuk 3. Deze uitbreiding blijkt zowel analytisch als computationeel onhandelbaar. Alhoewel het mogelijk is enkele elementaire monotonie-eigenschappen af te leiden voor dit model, richten we ons op numerieke aspecten van het model en de optimale strategie. In het bijzonder bespreken we een heuristiek voor de berekening van de optimale strategie. Deze heuristiek is gebaseerd op een nauw verwant model, waarvoor de optimale strategie eenvoudig te berekenen is. We evalueren en verfijnen de heuristiek aan de hand van een numerieke studie. De numerieke resultaten geven aan dat de prestaties van de heuristiek die van de optimale strategie zeer dicht benaderen.

In Hoofdstuk 6 bestuderen we een $\mathrm{M}^{\lambda_1,\lambda_2}|\mathrm{M}^{\mu}|1$ *preemptive priority* wachtrij-systeem met admission control alsmede termination control. Er zijn twee soorten opdrachten, die zich onderscheiden door hun opbrengst. Het systeem mag opdrachten van beide typen accepteren of weigeren en op elk moment mag een willekeurig aantal opdrachten van een der beide typen of zelfs beide typen uit het systeem verwijderd worden. We tonen aan dat de optimale beslissingen voor beide typen besturing zich laten karakteriseren door drempelwaarde-strategieën. Vervolgens breiden we dit resultaat onder een zekere restrictie ten aanzien van de kostenstructuur of de admission control structuur uit naar de natuurlijke multi-server variant van het model.

We besluiten het proefschrift met Hoofdstuk 7, waarin we in het kort enkele natuurlijke richtingen voor verder onderzoek bespreken. Daarin maken we onderscheid tussen enerzijds onderzoek dat specifiek gericht is op de analyse van werkstromen en anderzijds onderzoek naar structuur-resultaten voor de

dynamische besturing van wachtrijsystemen. In het kader hiervan staan we stil bij het feit dat de problemen waar men in de praktijk mee geconfronteerd wordt, veel groter en complexer van aard zijn dan die welke gerepresenteerd kunnen worden middels de (generieke) modellen die we beschouwen in dit proefschrift. We zijn er desalniettemin van overtuigd dat onze analyse een essentiële stap in de richting van meer praktische uitbreidingen vormt. Een beter (wiskundig) inzicht in (de structuur van) de optimale strategieën voor vereenvoudigde problemen zal uitermate bruikbaar zijn in gecompliceerdere situaties, waarin nauwkeurige heuristieken voor het omgaan met een ontoereikende capaciteit niet alleen gewenst, maar zelfs noodzakelijk zijn.

About the author

The author of this dissertation was born on the 4^{th} of November, 1976. Between 1988 and 1994, he attended the Pax Christi College in Druten, Gelderland. Subsequently, he studied Applied Mathematics at Eindhoven University of Technology. He carried out his final project at and for Consul Risk Management, Delft, the largest independent security software company in The Netherlands. His Master's thesis is titled "Performance analysis of mainframe computer systems". After obtaining the MSc degree (cum laude) in August of 1998, he joined the faculty staff to become a PhD student in the field of Stochastic Operations Research, and the dynamic control of queueing systems in particular. This dissertation is the result of research conducted at the Department of Mathematics and Computer Science of Eindhoven University of Technology, and the Department of Economics and Econometrics of the University of Amsterdam, where the author was employed from 2002 to 2003. As of October 1^{st}, he has embarked upon a career in the insurance business.