SIMULTANEOUS OPTIMIZATION OF FLOW CONTROL AND SCHEDULING IN QUEUES

by

Yves De Serres
M.Sc. Mathematics, 1974
M.Sc. Telecommunications, 1984

Department of Electrical Engineering
McGill University
Montreal, Canada

June 1991

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Doctor of Philosophy

SIMULTANEOUS OPTIMIZATION OF FLOW CONTROL AND SCHEDULING IN SINGLE-SERVER QUEUES

by

Yves De Serres
M.Sc. Mathematics, 1974
M.Sc. Telecommunications, 1984

Department of Electrical Engineering
McGill University
Montreal, Canada

June 1991

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Doctor of Philosophy

"Le génie n'existe pas.
Ce qui existe, c'est le travail
Travailler trois, quatre, cinq fois plus que tout le monde,
voilà le génie."

Hideyo Noguchi

"La méthode est lente, difficile, aléatoire
Elle expose à beaucoup d'échecs
mais un chercheur qui renonce à perdre son temps
est perdu pour la recherche"

Jean Delay

Acknowledgements

I would like to thank Bell Northern Research for making available its computing facilities. I would also like to thank l'Institut National de la Recherche Scientifique (INRS Télécommunications) for funding this research and providing an office space.

I would like to express my gratitude to my thesis advisor, professor Michael Kaplan. His enlightening lectures at the undergraduate level sparked my interest in telecommunication networks. The breadth and depth of his expertise made my association with him at the master and doctorate levels both a challenging and a rewarding experience. His continuous support and assistance are greatly appreciated.

Abstract

This thesis deals with dynamic control of multi class single server queues. In a broad class of such systems, fixed-priority rules, although static, have been shown to be optimal among dynamic schedules. This result however rests on the independence between the order of service and the pattern of arrivals, an assumption which is violated when flow control is exercised. We seek to characterize the optimal scheduling in the presence of flow control, when both scheduling and flow control are optimized jointly.

The problem is formulated in a Markovian environment. The cost/benefit structure is such as to penalize delay and reward throughput. It is first shown that the pervasive μc-rule is not optimal in general, even when flow control is jointly optimal. Restricting the setup to class independent user fees and waiting cost-rates, the problem amounts to characterizing the combination of preemptive scheduling and flow control optimizing the tradeoff between delay and throughput in multi-class single server queues. In the special case of two queues, one being saturated, we establish the joint optimality of SEJF (Shortest Expected Job First) scheduling and threshold type flow control. We conjecture that SEJF remains optimal in the presence of multiple independent Poisson arrival processes, provided that the flow control is simultaneously optimal. The conjecture is supported by an extensive computational investigation which reveals as well that the jointly optimal monotonic flow control is characterized by switching curves that are very nearly linear. From this observation we derive an approximate performance analysis based on a steady-state assumption; the approximation is shown to be very robust and capable of remarkable accuracy. The sensitivity of the performance to the scheduling rule is quantified. The class of window flow controls is observed to be only slightly suboptimal.

The conjectured optimality of SEJF is not proved yet. We show that well established methods either fail completely in the present setting or present unexpected difficulties in their application. Experiments with more general models suggest that the optimality of SEJF is insensitive to the exponential assumption for arrival processes, provided that the flow control is jointly optimal. It is also observed that the optimality of SEJF persists at the sink node of a tandem network of two queues with scheduling and flow control at both nodes; again, this optimality of SEJF depends on simultaneously optimal monotonic flow control at each node.

Sommaire

Cette thèse traite du contrôle dynamique dans les files d'attente à serveur unique et à population hétérogène. L'ordonnancement des tâches selon des priorités fixes est optimal pour une classe générale de systèmes de files d'attente. Cependant la validité de ce résultat dépend de l'indépendance entre le processus de service et le processus des arrivées; il est clair que cette condition n'est pas satisfaite lorsque le processus des arrivées est soumis à un contrôle. Nous nous proposons de déterminer la structure de l'ordonnancement optimal en présence d'un contrôle dynamique des admissions, lui-même sujet à optimisation.

Le problème est formulé dans un contexte markovien. Une structure économique est introduite pour traduire la pénalité attribuée à un long délai et le bénéfice correspondant à un débit élevé. A l'aide d'exemples, nous montrons que la règle μc n'est pas optimale de façon générale, même si le contrôle des admissions est simultanément optimal. Dans le contexte restreint où les frais d'admission et les coûts d'attente dans la file ne dépendent pas de la classe des tâches, l'objectif est de déterminer l'ordonnancement préemptif et le contrôle des admissions dont la combinaison produit le compromis optimal entre délai et débit.

Le cas spécial de deux files d'attente, l'une étant saturée, est considéré en premier lieu. Il est établi que l'ordonnancement des tâches dans l'ordre des temps moyens de service (SEJF) est optimal lorsque jumelé à un contrôle des admissions de type seuil conjointement optimal. Une analyse numérique d'un large échantillon de paramètres supporte la conjecture que la règle SEJF demeure optimale dans le cas de deux processus d'arrivées de type Poisson indépendants et contrôle des admissions conjointement optimal. De plus, cette analyse révèle que la frontière entre le domaine optimal d'admission et le domaine optimal de rejet est presque linéaire. Ces observations sont mises à profit dans la conception d'une procédure approximative pour le calcul du compromis optimal entre délai et débit. Cette approximation s'avère très robuste et possède un erreur relative infime.

La preuve de l'optimalité de la règle SEJF, jumelée à un contrôle des admissions de type seuil, est jonchée de difficultés. On montre comment le problème sort du champ d'application de certaines méthodes; d'autres techniques, bien qu'applicables en théorie, présentent des difficultés analytiques qui sont discutées. Enfin, on considère quelques généralisations du modèle qui indiquent que le contexte dans lequel la règle SEJF est optimale peut être considérablement élargi.

Table of contents

List of Figures

List of Tables

Chapter 1 Introduction

New communication networks like ISDN, broadband ISDN, and high speed Local Area Networks are designed to provide a flexible environment for the integration of a multitude of services. While advances in fiber optics and VLSI increase transmission and switching capabilities, efficient resource management is necessary to ensure that the multiple and heterogeneous demands placed on these networks are satisfied within their respective grades of service. To achieve this, network activity must be controlled to avoid, or at least to diminish, congestion that would result in reduced performance.

Congestion control has always been an important part of the design and analysis of telecommunication networks in general, and of computer communication networks in particular. The evolution towards packet speech and video ([78]) and the promise of broadband ISDN based on the new ATM technique ([4], [5]) have created new problems of traffic management and fueled research in this area. On the other hand, congestion control has a role in the design and operation not

only of telecommunication networks, but of many other systems as well, such as industrial production lines.

Stochastic queueing models are widely used to study the performance of these systems. Variations of job interarrival and service times can create congestion, manifested by long delays and reduced throughput, in an uncontrolled system. Ways to temper congestion in networks include routing, which involves the selection of the best paths for jobs already accepted into the network, flow control, which determines the amount of traffic accepted into the network, and scheduling, which reallocates network delay among the job classes by specifying the order of transmission.

1.1 Scope and purpose

A particular control is said to be *dynamic* if it responds to actual network operating conditions, and *static* otherwise. The implementation of a dynamic control requires some form of state feedback; the implementation of a static control does not. Dynamic controls are of interest partly because they represent the very best that can be done when information is abundant and free, and partly because they are occasionally relatively insensitive to lack of precision in the prescription of network operating parameters. On the debit side, dynamic controls are notoriously difficult to optimize and frequently costly to implement.

Frequently, however, it can be shown *a priori* that the search for an optimal

dynamic control can be restricted to the members of some parametric family. The so-called "bang-bang" controls are one such family. So too are threshold or switch-curve policies. In showing that the search can indeed be so restricted, one is said to have "characterized" the optimal control. Such characterizations are considered interesting first, because they tend to confirm that the optimal control behaves qualitatively as one might have expected, and second, because of the possibility that the particular form of the optimal control can be exploited to reduce the computational complexity of the optimization.

The characterization of optimal policies for flow control, routing or scheduling has generated a substantial recent literature; see [69] for a comprehensive survey With few exceptions, the three problems are considered separately; the models dealt with are typically simpler than Jackson networks, and feature only one of the three types of control. The work reported here continues the focus on simple models, but with the difference that flow control and scheduling are optimized *together*. Its goal is to characterize the *jointly* optimal policies and to suggest simple guidelines for designing near-optimal ones. The models considered have in common that there are rewards to the system for accepting new tasks, and penalties proportional to response time for each task admitted. An *optimal* flow control/schedule maximizes the system's rate of return. The following summarizes the results:

- The so-called μc-rule, by which a task is ranked for service according to

the product of its service-completion rate μ and waiting cost rate c, is known to be optimal in the class of dynamic and static schedules for a variety of lossless single server queues ([1], [11], [51]). It is known to be (strictly) sub–optimal where there are finite, separate buffers for each job class ([75]). Such buffering itself is a sub optimal form of flow control. We show by example that the μc–rule remains sub optimal when flow control and schedule are optimized together.

• When rewards and waiting cost rates are independent of job class, the control problem amounts to the optimization of the tradeoff between delay and throughput in multi class, single–server queues. The μc–rule in this case specializes to what we call the μ rule, or SEJF (Shortest Expected Job First); priority goes to the task in queue with the largest service–completion rate. We show by example that the μ rule is sub-optimal in this setting when flow control takes the form of finitary buffering of the individual classes. This time, however, the μ-rule seems to recover its optimality when the flow control is optimized as well. We offer a proof of this fact in the case that there are two job classes, one Poisson (meaning that the corresponding arrival instants form a Poisson point process) and the other saturated, in the sense that there are always tasks from that class available for service. Extensive computational work suggests that the result is valid more generally, when there are multiple Poisson flows, but we have not succeeded in finding a proof; we describe some of the approaches tried, and where they seem to fail.

• The form of the optimizing flow control is investigated assuming multiple independent Poisson flows. The numerical work suggests that the optimal policy is characterized by a family of switch-curves, one for each job class. Moreover, it reveals that the boundaries between the acceptance and rejection regions (for each class of jobs) in the multi-dimensional state space are very nearly linear. This suggests a *steady-state* approximation which greatly simplifies the computation of the optimal flow control and of the corresponding performance. This approximation is shown to be very robust and capable of remarkable accuracy.

• Window flow controls are attractive because they are easy to apply. Assuming SEJF service, the class of window flow controls is observed to be only slightly suboptimal; any point on the overall optimal delay/throughput tradeoff curve can be achieved very closely by an appropriate window flow control. This observation illuminates the relative value of state feedback in the control of queueing systems.

1.2 Background

We review here the main results concerning dynamic scheduling and dynamic flow control, as well as the few recent contributions where both scheduling and flow control are studied jointly.

1.2.1 Scheduling

The pure scheduling problem in multi-class, single-server queues consists in specifying the order of service so as to optimize some economic objective func-

tion. This function depends on class-dependent waiting cost-rates and service-completion rewards; the goal is to minimize either long-run average costs or infinite horizon expected discounted costs.

The static problem was solved in [17], where it was established that the priority assignment minimizing long-run average costs in $M/G/1$ queues is the so-called μc-rule. The μc-rule assigns priorities in decreasing order of the product of class parameters μ (inverse mean service time) and c (per-second cost of delay). This result was obtained in [17] by an interchange argument, a technique that has since been applied in much more general settings [79].

The most general results for dynamic scheduling come from the theory of bandit processes. The fundamental result establishes the optimality of Gittins *index* policies [21], [83], [84]. When the arrival process of each class is Poisson, the dynamic scheduling problem in multi-class, single-server queues, can be formulated as an arm acquiring bandit [25], [79]. In the context of multi-class queues, an index policy is characterized by an algorithm which assigns to each arrival a single real index depending on its class and on the arrival, cost and service rates in the system; priority goes to the task in queue whose index is largest. The key point is that although index policies are static, they are nonetheless optimal in the class of dynamic schedules in a variety of important models ([10], [32], [33], [40], [74], [79]). In some models, the optimal scheduling is more complex but heuristic index rules are only slightly suboptimal [52] [60].

The μc- rule is a particular instance of an index policy. The μc rule minimizes the expected discounted cost over an infinite horizon in $M/G/1$ queues without preemption ([51]), and in $G/M/1$ queues with preemption ([1], [11]), as well as at the second node of a tandem network of two multi-class $G/M/1$ queues with preemptive service ([53],[51]).

The optimality of fixed-priority rules among dynamic schedules in multi class $M/G/1$ queues had been established in [28], [39], and [76] by arguments from the theory of semi-Markov Decision Processes.

The optimal dynamic scheduling problem with an average waiting time constraint for some classes of jobs is studied in [55], [62], [63]. The general result in this context is the optimality of randomized fixed-priority rules. See also [15], [21], and [58]. When messages have *individual* constraints on their waiting time, that is, in systems with impatient customers, the EDF-rule (Earliest Deadline First) optimizes a variety of objective functions [3], [13], and [57]. See also [86].

There is very little work on the scheduling problem in networks of queues. Bounds were obtained in [60] for the minimum long-run average cost in a network version of Klimov's model. A reformulation of the cost function was used to devise a heuristic index rule whose performance is evaluated by the bounds.

The so- called *Brownian network* ([29], [30], [31]) is a heavy traffic approxi-

mation to a multi class queueing network with dynamic scheduling. The model was applied in [31] to a two station closed network in heavy traffic; the objective is to schedule the two servers to maximize the long-run average throughput of the network. The heavy-traffic solution is used to devise a *good* sequencing under less loaded conditions; this scheduling is a static priority rule at each node.

1.2.2 Flow control

The main theoretical results on dynamic flow control concern the optimality of switch curve policies. A switch-curve policy is characterized by a surface in state space; a new task is admitted if and only if the state at the time of arrival lies below the surface ([14], [23]). Window policies, which are used in practice, are the simplest examples ([42], [88]). These results establish the monotonicity of the surface defining the boundary between the acceptance and rejection regions; this confirms the intuition that, if it is optimal to accept an arriving task in a given state, then it should be optimal also to accept this task if one (or more than one) task is removed from the system.

In general, the cost/benefit structure involves waiting cost-rates for jobs already in the system and rewards for each admitted task. This structure reflects the fact that delay is bad and that throughput is good. The objective — to maximize either the expected discounted benefit over finite or infinite horizons, or the long run average benefit — amounts to characterizing the tradeoff between delay and throughput.

For an $M/M/1$ queue, the admission policy maximizing long-run average benefits, within the class of threshold policies, was obtained in [56]. A threshold type policy admits a job only if the number already in queue is smaller than the threshold. Extensions to $GI/M/1$ and $GI/M/c$ queues are found in [89] and [90] where the restriction to threshold policies was shown to be without loss of optimality.

The Inductive Approach has been applied to a variety of Semi Markov Decision Process formulations of the flow control problem, [23], [47], [48], and [71]. The monotonicity property of finite-horizon optimal flow controls carries over to infinite-horizon and long-run average problems by general results of the theory of semi-Markov Decision Processes [45], [46].

There are cases where it is difficult to derive monotonicity from the dynamic programming optimality equation and induction, [61]. The Linear Programming Approach introduced in [61], and later generalized in [80] is an alternative technique applied with success to a wide variety of models in [80]. See also [14], [41], and [49].

The dynamic flow control problem is formulated in [42] as a constrained optimization problem. The objective is to maximize the expected throughput of an $M/M/1$ queue in equilibrium, subject to a bound on the average delay and a bound on the maximum admissible offered-load. This is closely related to the

maximization of the long run average net benefit. The optimal delay/throughput tradeoff is achieved by a randomized window flow control; randomization is necessary at the window boundary to achieve the delay constraint exactly. The analysis generalizes to $M/M/n$ queues and to networks of queues ([43], [44], [59], [88]). Dynamic flow control for delay-constrained multipacket messages is also studied in [37].

1.2.3 Scheduling and flow control

The joint optimization of scheduling and input/output rate control in a multi-class, single server queue, with linear holding costs is considered in [12] and [87]. The μc rule is shown to minimize the expected discounted cost over an infinite horizon when the input and output rates are bounded below (throughput requirement) and above (capacity requirement) respectively. Since there is no reward for serving a task, it is not surprising that discounted costs are minimized by setting the input rates equal to their lower bound. In our setting, the flow control must decide whether to accept or reject jobs upon arrival; this decision is based on the reward brought to the system by the admission of the arriving job, compared to the accrued waiting cost resulting from this admission.

We have seen above that the μc-rule is broadly optimal in multi-class single-server queues, provided that the input process to the queue is unaffected by the order of service. In the presence of flow control, the powerful results from the theory of bandit processes are not directly applicable. It is shown in [75] that the

μc rule is not generally optimal in multi-class $M/M/1$ queues with finite separate buffers for each class (a form of flow control). However, the flow control in [75] is not optimal. We show here that the μc-rule is not generally optimal even when both scheduling and flow control are optimized together.

Scheduling and flow control are optimized together in [82], where the Browman network approximation is applied to the dynamic scheduling of a multi class two station network with controllable inputs. This model incorporates both input and sequencing decisions. The sequencing decisions specify, at any time, which class of jobs to process at each station of the network. There is an endless line of jobs waiting to gain entry into the network. The proportion of jobs of each type in the waiting line is pre-specified and determines the order in which jobs are admitted into the network. The role of input control is to determine the timing of release of the job at the head of the waiting line; the input control is not allowed to select the class of customer to admit next. Thus the setup in [82] differs from the classical dynamic flow control problem in network of queues, where the decision option is whether to accept or reject jobs at arrival epochs. Both types of flow control are considered in this thesis.

1.3 Thesis overview

In Chapter 2, we propose a multi-class, single-server queue as a simple model of a system with combined flow control and scheduling. The rationale for our choice of model is discussed. A cost/benefit economic structure is defined to

capture the tradeoff between delay and throughput. We derive the dynamic programming optimality equation which characterizes the flow control and *preemptive* service policy which together minimize the long-run average net cost. We show by example that the μc rule is not generally optimal in the presence of flow control, even if scheduling and flow control are optimized together. We show further that the μ rule (SEJF rule), to which the μc–rule reduces when the waiting–cost rate c is independent of job class, is strictly suboptimal when flow control takes the form of separate buffering for each class.

In Chapter 3, we consider the special case in which there are only two queues, of which one is saturated. When waiting cost–rates and user fees are class–independent, we show that the optimal policy combines SEJF scheduling with a threshold type flow control; in particular, that the μ–rule recovers its optimality provided that the flow control is simultaneously optimal.

The more general case of two independent Poisson arrival processes is considered in Chapter 4. An extensive computational investigation suggests that SEJF remains optimal when coupled to simultaneously optimal monotonic flow controls. It also reveals that the optimal admission policies have a simple form. Assuming SEJF service, we show that restricting the optimization to the class of threshold–type policies or to the class of window policies is only slightly suboptimal.

Exploiting the simple form of the jointly optimal flow control, a steady–state

approximation is derived in Chapter 5 for multi- class queues under SEJF scheduling and monotonic flow control. The approximation greatly simplifies the computation of the best threshold-type flow control. The robustness and accuracy of this approximation are evaluated by comparing performance with truly optimal flow controls.

Extensions to more than two job classes are considered in Chapter 6. The application of a variety of techniques to prove the optimality of SEJF when combined with jointly optimal flow controls is investigated in Chapter 7. Some of these techniques are shown to fail in our setting, the unexpected difficulties encountered in the application of others methods are discussed.

Chapter 8 considers few generalizations of our basic model. SEJF appears to remain optimal when the arrival processes are Markov-Modulated Poisson Processes (MMPP), rather than simply Poisson. SEJF also seems to be optimal at the second node of a network of two multi class $M/M/1$ queues in tandem. In both cases, the corresponding optimal flow controls are monotonic, but with a more complex form than in the case of a simple $M/M/1$ system.

Chapter 2

A multi–class single–server queue with dynamic flow control and scheduling

The problem of simultaneously optimizing flow control and scheduling is formulated for a single–server queue. The dynamic programming optimality equation characterizing the optimal policy is obtained and used to establish that the μc–rule is not optimal in general in the presence of flow control even if scheduling and flow control are optimized together.

2.1 The model

Our study of the joint dynamic control of admissions and service order in queues is based on Dynamic Programming. We chose a simple Markovian setting featuring a single server, m job classes, where m is an integer bigger than one, and potentially unlimited buffering. The arrival processes (one for each job class) are independent and Poisson; service times within each class are *iid* and exponentially

distributed. Performance is measured by the volume of carried traffic in each class and by the class-dependent mean delays. The goal of the system designer is to achieve a favorable tradeoff among these variables by appropriately selecting the admission controller, which discards jobs on the basis of class and of network congestion, and the scheduler, which decides to which class the server should be assigned. We assume that service is preemptible, so that the scheduler revises its decisions continually, and that service of a preempted job resumes at the point of interruption. We assume also that remaining service times are unknown to the system; the information available to the controllers is the record of arrivals and departures (by class) to the present time, and thus, in particular, the number of jobs of each class in the system.

The multi-dimensional performance measure is reduced to a single scalar index in the usual way, by supposing that the system is rewarded for each job processed and penalized for the delay incurred. The efficacy of a particular compromise between throughputs and delays is measured by the corresponding net cost rate (the negative of the net income rate) — the difference between the average rate at which money leaves in the form of compensation to jobs which wait, and the average rate at which rewards accrue due to the entry of new jobs. We are interested exclusively in steady-state behaviour; the averages referred to are with respect to the stationary probabilities induced by the choice of control. These stationary probablities exist in all but degenerate instances of our problem.

Each job class is thus characterized by its mean arrival rate (prior to control), its service completion rate (the inverse mean service time), the reward to the system for each job processed and the penalty for delay. We assume, regarding the latter, that the penalty is linear in delay. The parameters for class i are λ_i (the arrival rate), μ_i (the service completion rate), r_i (the reward per job processed) and c_i (the penalty per job per second of delay). The effect of the choice of a particular control π (a combination of an admission controller and a scheduler) is expressed through the mean throughputs $\overline{\gamma}_i(\pi)$, the mean occupancies $\overline{x}_i(\pi)$ and the mean delays $\overline{D}_i(\pi)$ associated in steady state with the job classes indexed by i. These are well-defined and finite whenever the c_i are strictly positive, which is the situation of interest. They are related by Little's Theorem applied separately to each class:

$$\overline{x}_i(\pi) = \overline{\gamma}_i(\pi)\overline{D}_i(\pi); \tag{2.1}$$

implicit in this is the convention, on which we comment below, that the ***delay*** incurred by a job in its sojourn through the system includes service as well as waiting time, and that the ***occupancy*** of class i includes the job (if any) in service as well as the jobs in queue.

The performance of the control π, relative to the reward/cost structure defined by the r_i, c_i, is given by

$$v(\pi) \triangleq \sum_{i=1}^{m} c_i\overline{x}_i(\pi) - r_i\overline{\gamma}_i(\pi) \tag{2.2a}$$

$$= \sum_{i=1}^{m} \overline{\gamma}_i(\pi)\left(c_i\overline{D}_i(\pi) - r_i\right). \tag{2.2b}$$

It is this quantity which is to be minimized by optimization of π. In view of the preemptive-resume character of the service discipline and the memoryless property of the inter-arrival and service time statistics, the optimal π is found among those strategies whose action at time t depends solely on the occupancies $X_i(t)$ of the different classes (Section 2.2). Such π, said to be *stationary* and *Markov* ([85]), give rise to occupancy processes $\mathbf{X}^{(\pi)}(t) = \left(X_1^{(\pi)}(t), ..., X_m^{(\pi)}(t)\right)$ which are Markov. This observation, and the fact that the cost rate $v(\pi)$ for stationary Markov π is just the steady state mean of some memoryless functional $f\left(\pi, \mathbf{X}^{(\pi)}(t)\right)$ of the occupancy process, together place the problem of optimizing π within the purview of the theory of Markov Decision Processes and Dynamic Programming.

Remarks

• The assumed preemptibility of service simplifies the state description. In models with preemption, the occupancy vector $\mathbf{X}$ is Markov — in fact, an m-dimensional Birth-Death process. In models without preemption, $\mathbf{X}$ is not Markov; it can be made Markov by adding a coordinate to identify the class in service at the present time, or by subsampling at departure instants, in which case the Birth Death property is lost.

• The preemptible version of the problem is less constrained, and therefore a more favorable setting in which to evaluate the merit of dynamic control, than

the non preemptible one. The performance figures cited in later chapters are thus optimistic. It will be seen that there are ranges of parameter values in which the gains due to scheduling are marginal; this will be so *a fortiori* when the service is non-preemptible.

• Our primary practical interest is in packet-switching telecommunication networks, where the notion of "service" is identified with transmission on a physical channel. Packet transmissions are usually (but not always) non-preemptible. But the entities, or jobs, submitted to the network by its subscribers are not packets, but *messages*, which typically are aggregates of packets. Viewed from the message level, service in a packet network is indeed preemptible — with preemptions restricted to packet boundaries. Our model can be thought of as an idealized packet multiplexor in which packets are infinitessimally small, or at least very much shorter than messages.

• The reward/cost structure in the model assigns cost to delay (waiting time plus service time) rather than to waiting time alone. The two options are not equivalent. In either case there will be a bias (depending on the rewards and cost rates) against admitting long jobs. Choosing delay, rather than waiting time, as the measure of grade-of-service, reinforces that bias. There are no constraints in the formulation of the optimization problem to ensure minimal levels of service for individual users. The volume of traffic admitted into the system from each class is whatever makes the operation of the system most profitable.

• In the absence of flow control, delay $\overline{D}_i(\pi)$ and backlog $\overline{x}_i(\pi)$ are equivalent objective functions, differing only by a constant scale factor -- the class-i arrival rate. In the present case, the scale factor itself is subject to optimization. Nonetheless, the equivalence between delay and backlog persists, in the following sense. Say that control π is *optimal* for given $\mathbf{r} = (r_1, r_2, ..., r_m)$, $\mathbf{c} = (c_1, c_2, ..., c_m)$ if and only if π minimizes the corresponding cost rate $v(\cdot)$. Say that π is *admissible* if and only if there is *no* other control π' satisfying

$$\overline{\gamma}_i\left(\pi'\right) \geq \overline{\gamma}_i(\pi), \quad \overline{D}_i\left(\pi'\right) \leq \overline{D}_i(\pi) \tag{2.3}$$

for all i, with at least one of the inequalities strict, it being understood that $\overline{D}_i(\pi) = 0$ whenever $\overline{\gamma}_i(\pi) = 0$. The claim is that all optimal controls are admissible.

The claim is proved by observing that each class-i job admitted to the system brings a net reward equal to $r_i - c_i d$, where d is the corresponding delay, and that strict positivity of $r_i - c_i d$ is *necessary* (but not sufficient) for admission. So if π is optimal, then for every i either $\overline{\gamma}_i(\pi) = 0$ or else $r_i - c_i \overline{D}_i(\pi) > 0$. It follows from this, for π' satisfying the inequalities above, that $v\left(\pi'\right) \leq v(\pi)$; strict inequality in any of the above would imply $v\left(\pi'\right) < v(\pi)$, violating the assumed optimality of π. So π is necessarily admissible.

2.2 Formal description of the Markov decision problem

The population process $\mathbf{X}(t)$ defined in Section 2.1 evolves as follows. Potential transitions in $\mathbf{X}(t)$ occur at arrival and departure epochs. Whenever the population process enters state $\mathbf{x}$ an action $\mathbf{a}$ is chosen. The cost to the system for selecting action $\mathbf{a}$ in state $\mathbf{x}$ is $K(\mathbf{x},\mathbf{a})$. After an *exponentially* distributed amount of time, with parameter $\lambda(\mathbf{x},\mathbf{a})$, the population becomes $\mathbf{y}$ with probability $p(\mathbf{x},\mathbf{a},\mathbf{y})$. Upon entering state $\mathbf{y}$ an action is taken, the corresponding cost is incurred, and the process continues.

The action $\mathbf{a}$ is an $(m+1)$ tuplet

$$\mathbf{a} = (a_1, a_2, ..., a_{m+1}) \in \{0,1\}^m \times \{1,2,...,m\}.$$

The component a_i, $1 \leq i \leq m$ indicates whether a type-i arrival should be accepted ($a_i = 1$) or rejected ($a_i = 0$). The last component, a_{m+1}, indicates what type of customer is given priority for service — type i, if $a_{m+1} = i$.

Transitions occur at rate

$$\lambda(\mathbf{x},\mathbf{a}) = \sum_{i=1}^{m} a_i\lambda_i + \mu_{a_{m+1}} I[\mathbf{x} \neq \mathbf{0}] \tag{2.4}$$

where $I[\cdot]$ is the indicator function. The transition probabilities are

$$p(\mathbf{x},\mathbf{a},\mathbf{y}) = \begin{cases} \lambda_i/\lambda(\mathbf{x},\mathbf{a}) & \text{if} \quad \mathbf{y} = A_i\mathbf{x}, \quad a_i = 1, \\ \mu_i/\lambda(\mathbf{x},\mathbf{a}) & \text{if} \quad \mathbf{y} = D_i\mathbf{x}, \quad a_{m+1} = i, \end{cases} \qquad 1 \leq i \leq m, \tag{2.5}$$

and the cost $K(\mathbf{x},\mathbf{a})$ is the expected net cost until the next transition, that is

$$K(\mathbf{x},\mathbf{a}) = \frac{\mathbf{cx}}{\lambda(\mathbf{x},\mathbf{a})} - \sum_{j=1}^{m} r_j p(\mathbf{x},\mathbf{a},A_j\mathbf{x}), \tag{2.6}$$

where

$$\mathbf{cx} = c_1x_1 + c_2x_2 + ... + c_mx_m, \tag{2.7}$$

and $A_i\mathbf{x}$, $D_i\mathbf{x}$, are the population vectors obtained from $\mathbf{x}$ by adding or deleting a type-i customer.

A policy π specifies the control applied at each time instant t. Let $A_i(t)$ be the number of arrivals in $[0,t]$ in a Poisson process with rate λ_i, $1 \leq i \leq m$, and let $\mathbf{X}^{(\pi)}(t) = (X_1^{(\pi)}(t), X_2^{(\pi)}(t), ..., X_m^{(\pi)}(t))$ denote the population process under policy π; $\mathbf{X}^{(\pi)}(\cdot)$ is taken to be left-continuous. $V_T^{(\pi)}(\mathbf{x}_0)$ stands for the net cost accumulated during the interval $[0,T]$ when the initial backlog is $\mathbf{x}_0 = (x_{1,0}, x_{2,0}, ..., x_{m,0})$ and the policy is π:

$$V_T^{(\pi)}(\mathbf{x}_0) = E_{\mathbf{x}_0} \sum_{i=1}^{m} \left\{ \int_0^T c_i X_i^{(\pi)}(t)dt - \int_0^T r_i \pi_i(\mathbf{X}^{(\pi)}(t))dA_i(t) \right\}, \tag{2.8}$$

where $E_{\mathbf{x}_0}$ denotes expectation with starting backlog $\mathbf{x}_0$. The goal is to determine the policy minimizing the asymptotic cost-rate

$$\limsup_{T\to\infty} \frac{1}{T} V_T^{(\pi)}(\mathbf{x}_0), \tag{2.9}$$

where lim sup is used to ensure the existence of the limit.

In a series of papers ([6]-[9]), Borkar has studied the control of Markov chains under long-run average cost criterion. The basic assumptions he introduced were devised to cover average cost optimization in queueing networks. The key assumption is that the cost-rate function is a non-negative, unbounded function of the

state, a condition which usually prevails in queueing models. The assumptions in [6] are trivially verified in the present context.

From general results in [6], the optimal policy minimizing the asymptotic cost-rate (2.9) is stationary, non-randomized, and Markov. Moreover, the optimal policy is specified by the average-cost optimality equation of dynamic programming.

For a stationary Markov policy f, the population process $\mathbf{X}^{(f)}(t)$ is a Markov process on state space

$$S = \{\mathbf{x} = (x_1, x_2, ..., x_m) | x_i \geq 0, 1 \leq i \leq m\}. \tag{2.10}$$

A non randomized stationary Markov policy f selects action $f(\mathbf{x})$ whenever the process is in state $\mathbf{x}$. A stationary Markov policy induces a Markov process with mean sojourn time $1/\lambda(\mathbf{x}, f(\mathbf{x}))$ in state $\mathbf{x}$.

Uniformization is a devise which converts a continuous-time Markov decision problem into one for which times between transitions are independent of the state and of the action taken. This uniformized process is in turn easily shown to be equivalent to a controlled Markov chain ([36], [67]). Uniformization of a controlled Markov process amounts to observing the system at each tick of an exponential clock which runs faster than the process itself. Therefore, between two successive transition epochs of the original process there will be many clock ticks which do not correspond to a transition epoch of the original process; these are referred to as false or potential transitions, and have no effect on the evolution of the process.

For our problem, the uniformized clock rate (also called the total event rate) can be taken to be

$$\Lambda = \sum_{i=1}^{m} (\lambda_i + \mu_i). \tag{2.11}$$

Let τ_k denote the sequence of clock ticks. For a given stationary policy f, let $\widehat{\mathbf{X}}^{(f)}(t)$ be the uniformized Markov process. With initial state $\mathbf{x}_0$, the expected net cost over the first H clock ticks (the horizon) is,

$$V_H^{(f)}(\mathbf{x}_0) = E_{\mathbf{x}_0} \sum_{k=1}^{H} \sum_{i=1}^{m} \Big\{ c_i \widehat{X}_i^{(f)}(\tau_{k-1})(\tau_k - \tau_{k-1}) - r_i I\left[\widehat{X}_i^{(f)}(\tau_k) - \widehat{X}_i^{(f)}(\tau_{k-1}) = 1\right] \Big\}. \tag{2.12}$$

Applying general results from [6], we have that

$$\limsup_{T\to\infty} \frac{1}{T} V_T^{(f)}(\mathbf{x}_0) = \Lambda \limsup_{H\to\infty} \frac{1}{H} V_H^{(f)}(\mathbf{x}_0), \tag{2.13}$$

and there is a unique solution g, $v(\mathbf{x})$ to the average–cost optimality equation

$$v(\mathbf{x}) = \frac{\mathbf{cx} - g}{\Lambda} + \sum_{i=1}^{m} \frac{\lambda_i}{\Lambda} \min\{v(A_i\mathbf{x}) - r_i, v(\mathbf{x})\} + \min_{1\le i\le m} \left\{ \frac{\mu_i}{\Lambda} v(D_i\mathbf{x}) + \sum_{j\neq i} \frac{\mu_j}{\Lambda} v(\mathbf{x}) \right\}. \tag{2.14}$$

The stationary policy determined by the optimality equation is optimal, and the scalar g is the minimum average cost per unit of time (independent of the starting state); that is,

$$g = \inf_{\pi} \limsup_{T\to\infty} \frac{1}{T} V_T^{(\pi)}(\mathbf{x}_0). \tag{2.15}$$

The so called relative values $v(\mathbf{x})$ can be interpreted as the expected first-passage cost from state $\mathbf{x}$ to state $\mathbf{0}$, assuming that the optimal policy is applied and that the holding cost rate in state $\mathbf{x}$ is $\mathbf{cx} - g$.

2.3 Computation of the optimal policy

Policy iteration is an algorithm which uses the average cost optimality equation to produce a sequence of policies with decreasing average cost ([77]). At each step of the algorithm, policy-iteration improves the current policy, that is it determines a policy with smaller average cost per unit time. The policy-iteration algorithm converges after a finite number of iterations to the optimal policy minimizing the long-run average cost. The algorithm is found to be very robust and to converge very fast in specific applications.

An application of policy-iteration to determine the jointly optimal flow control and scheduling in our model requires that we truncate the countably infinite multi-dimensional state space. We are guided by known results on the optimal flow control of queueing systems with a cost/benefit structure similar to ours; these results confirm the intuition that the number of jobs in the system should not be allow to grow indefinitely. Thus we apply policy-iteration to the truncated multi class single-server queue in which a finite buffer of length L is available to the community of m job-classes. The state space for this truncated queue is

$$S_L = \{(x_1, x_2, ..., x_m) \in S | x_1 + x_2 + ... + x_m \leq L\}$$

The boundary B_L of S_L is the subset of states $(x_1, x_2, ..., x_m)$ for which $x_1 + x_2 + ... + x_m = L$, that is for which the buffer is full. The optimal policy π_L^{opt} minimizing the average cost for the truncated queue is the restriction to S_L of the optimal policy π_∞^{opt} for the infinite queue, provided that the admission region $A_L(\pi_L^{opt})$ - the subset of states $(x_1, x_2, ..., x_m) \in S_L$ in which π_L^{opt} admits a type-i customer for some $i = 1, 2, ..., m$ — does not touch the boundary B_L. This follows from the observation that a solution g, $v(\mathbf{x})$, to the truncated version of the optimality equation (2.11), specifying a policy not touching the boundary, can be extended to a solution of the original (not truncated) optimality equation; indeed, values of $v(\mathbf{x})$ for *upper* states outside the admission region depend only on values of $v(\mathbf{x})$ at *lower* states, and can thus be defined recursively from the dynamic programming optimality equation.

Each iteration of the policy improvement algorithm requires the solution of a system of linear equations. This system of equations is sparse by nature. The SPARSPAK package [22] was used in our programs.

2.4 Is the μc–rule optimal?

As mentioned in Chapter 1, the μc–rule is optimal in a large class of models. In the absence of flow control, the μc–rule is known to minimize the long run average cost in our multi-class single server queue ([11]). The proof hinges on the independence between service order and arrival process.

That the μc rule is not generally optimal in the presence of input constraints was first observed in [75] where the pure scheduling problem when only finite separate buffers are available is considered. Imposing a limit to the number of type i jobs in the system is a particular form of flow control. The question remains about the optimality of the μc–rule when scheduling and flow control are optimized together.

Application of policy iteration reveals that the μc–rule is not generally optimal in the presence of flow control, even if the flow control is simultaneously optimal. Figures 2.1 to 2.3 show the optimal policy when flow control and scheduling are optimized together in a single–server queue with two job classes with parameters:

$$\begin{array}{ll} \mu_1=5.0, & \mu_2=2.5, \\ \rho_1=0.4, & \rho_2=0.4, \\ c_1=100.0, & c_2=195.0, \\ r_1=800.0, & r_2=800.0, \end{array} \tag{2.16}$$

where $\rho_i = \lambda_1/\mu_i$, $i = 1,2$. In Figures 2.1 and 2.2, • indicates states where admission is the optimal action. In Figure 2.3, • indicates states where serving type 2 is the optimal action.

Since $\mu_1 c_1 > \mu_2 c_2$, the presence of states other than $(0, x_2)$ where providing service to type-2 is the optimal action, shows that the μc–rule is not optimal.

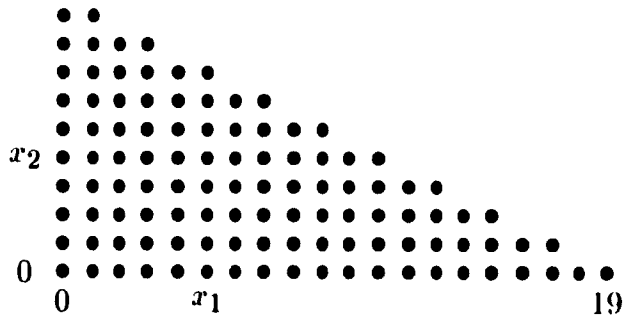

Fig. 2.1 Optimal flow control for class–1 jobs.

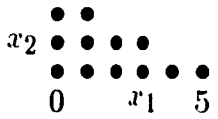

Fig. 2.2 Optimal flow control for class–2 jobs.

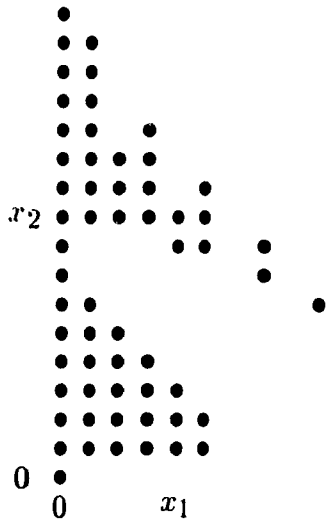

Fig. 2.3 Optimal scheduling.

The minimum cost for the selected set of parameters is -1984.5. It turns out that the admission policy of Figures 2.1 and 2.2 also minimizes the long-run average cost when service is provided according to the μc-rule; the corresponding average cost is -1980.5

When holding cost rates are class-independent, the μc-rule becomes the Shortest Expected-Job-First rule. The following example shows that SEJF is not generally optimal when flow control is exercised by finite separate buffers. The scheduling policy minimizing long-run average costs for buffers of length 5 is shown in Figure 2.4 for parameters:

$$\begin{aligned} \mu_1 &= 1.0, & \mu_2 &= 0.5, \\ \rho_1 &= 0.8, & \rho_2 &= 0.8, \\ c_1 &= 1.0, & c_2 &= 1.0, \\ r_1 &= 1.5, & r_2 &= 1.5. \end{aligned} \tag{2.17}$$

In Figure 2.1, o indicates states where serving type-1 jobs is the optimal action. The corresponding average cost is 4.9. The optimal cost is $-2/9$, obtained by never admitting any type-2 job into the system and admitting a type-1 job only in state $(0,0)$.

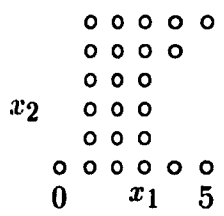

Fig. 2.4 Optimal scheduling for finite buffers.

2.5 The thesis setting

In the following chapters we study a special case of the basic model described above; namely, the case when holding cost-rates and admission rewards are class-independent. Minimizing the long-run average net cost amounts then to optimizing the tradeoff between delay (averaged over jobs from all classes) and throughput. We begin by investigating, in the next chapter, the case of only two queues, of which one is saturated.

Chapter 3

The saturated queue: a special case †

We consider here a special case of the model described in Chapter 1; namely two queues, one of which is saturated. The other is a Poisson queue with finite arrival rate.

The setting is particular in that jobs from one class are always available for admission into the system. One may think of this model as a limiting version of the basic model, in which the arrival rate of jobs from one class greatly exceeds both the arrival rate of jobs from the other class and the maximum service rate.

The saturated queue assumption amounts to controlling the timing of the release of these tasks into the system, rather than controlling whether to accept or reject new arrivals. Such an input control, combined with scheduling, was studied by Wein [82] in his heavy traffic analysis of a two-station network. He showed how controlling the timing of inputs is appropriate in certain specific applications

† The analysis of this chapter has been published in [18]

such as factory scheduling.

In this context, we show that the μc–rule is not optimal in general. On the other hand, when cost–rates and rewards are independent of class, the combination of scheduling and input control jointly minimizing the long–run average net cost is fully characterized. The effect of the saturated queue assumption is to reduced the general two–dimensional controlled process to a one–dimensional process, thus making possible the direct approach presented below. The analysis has some similarity with Naor's study of an $M/M/1$ queue in [56].

3.1 The model

Two streams of customers distinguished by their mean service times compete for a single server. Service times are exponentially distributed with means $1/\mu_1$ and $1/\mu_2$. Type–1 customers arrive according to a Poisson process at rate λ_1. We assume an infinite pool of type–2 customers waiting outside the system and immediately available for admission into the queue upon request from the flow controller. Each customer of type–i ($i = 1, 2$) pays r_i dollars upon entry into the system, and collects c_i dollars per unit of time until its departure. We stress that cost and reward are associated only to jobs admitted into the queue; in particular, type–2 jobs awaiting admission in the infinite pool do not contribute to system costs. Similarly, no cost is incurred by rejecting a type–1 job upon its arrival to the system.

The objective is to determine the structure of the flow control and schedule which *jointly* minimize the long–run average rate at which money leaves the system.

Since a type–2 customer is always available for admission and since rewards are collected upon entry, it cannot be optimal, for long–run average cost minimization, to admit a 2–customer unless to serve it immediately. Consequently, for the long–run average cost criterion, there will never be more than one type–2 job in the system, either being served or waiting in queue because of preemption by type–1 jobs. The state space is thus

$$S = \{(i,j) | i \geq 0, j = 0,1\}. \tag{3.1}$$

A policy specifies at any time what type of customers should be provided service, and whether a type–1 arrival should be accepted or rejected. The control is a pair $(a_1, a_2) \in \{0,1\} \times \{0,1\}$. The first component, a_1, indicates whether a type–1 arrival should be accepted ($a_1 = 1$) or rejected ($a_1 = 0$). The second component, a_2, indicates what type of customers is given priority for service, type-1 if $a_2 = 1$, and type-2 if $a_2 = 0$. Note that action $(a_1, 0)$ in state $(i, 0)$ calls for the input of a type–2 job (from the infinite pool) directly into service, in front of the i type-1 jobs already in the system. In view of the fact that there is never more than one type-2 job in the system, the scheduler is seen to exercise control on the timing of release of type-2 into the system.

A stationary policy π is a pair (π_1, π_2) of binary-valued functions on the state space $\{(i,j)|i \geq 0, j = 0,1\}$, each taking values in $\{0,1\}$. A stationary policy π applies control $(\pi_1(i,j), \pi_2(i,j))$ in state (i,j).

Let $A_1(t)$ be the number of arrivals by time t in a Poisson process with rate λ_1. We denote $\mathbf{X}^{(\pi)}(t) = (X_1^{(\pi)}(t), X_2^{(\pi)}(t))$ the state process under policy π; $\mathbf{X}^{(\pi)}(\cdot)$ is taken to be left-continuous. With starting state $\mathbf{x}_0 = (x_{1,0}, x_{2,0})$ the expected costs $V_T^{(\pi)}(\mathbf{x}_0)$ accumulated during the interval $[0,T]$ when starting in state $\mathbf{x}_0$ and applying policy π is

$$V_T^{(\pi)}(\mathbf{x}_0) = E_{\mathbf{x}_0}\Bigg\{ \int_0^T \left[c_1 X_1^{(\pi)}(t) + c_2 X_2^{(\pi)}(t)\right] dt$$

$$- r_2 \sum_{i=0}^{\infty} (1 - \pi_2(i,0)) \int_0^T I\left[X_1^{(\pi)}(t) = i, X_2^{(\pi)}(t) = 0\right] dt \Bigg\}$$

$$- r_1 \int_0^T \pi_1\left(X_1^{(\pi)}(t), X_2^{(\pi)}(t)\right) dA_1(t), \tag{3.2}$$

where $E_{\mathbf{x}_0}$ denotes the expectation with starting state $\mathbf{x}_0$ and $I[\cdot]$ is the indicator function. The goal is to determine the policy minimizing

$$\limsup_{T \to \infty} \frac{1}{T} V_T^{(\pi)}(\mathbf{x}_0). \tag{3.3}$$

We point out, before going on, that, by its very nature, our model lives in a non-work-conserving environment. Indeed both the total unfinished work and the total system population do depend upon the control policy applied, in particular, upon the admission control exercised, and thus are not conserved. That a policy, upon full depletion of type-1 jobs, prefers to keep the server idle and wait for the

next type 1 job rather than to admit and serve a readily available type–2 job, is simply a characteristic of the class of policies considered, namely those jointly managing admission and service.

3.2 The optimality equation

Uniformization ([36], [67]) reduces the above continuous-time problem to a discrete time problem. Decision epochs occur at rate $\Lambda = \lambda_1 + \mu_1 + \mu_2$. The average cost optimality equations are

$$v(i,0) = \min\Big\{ \frac{c_1 i - g}{\Lambda} + \frac{\lambda_1}{\Lambda}\big[v(i+1,0) - r_1\big] + \frac{\mu_1}{\Lambda} v((i-1)^+,0) + \frac{\mu_2}{\Lambda} v(i,0),$$

$$\frac{c_1 i - g}{\Lambda} + \frac{\lambda_1}{\Lambda} v(i,0) + \frac{\mu_1}{\Lambda} v((i-1)^+,0) + \frac{\mu_2}{\Lambda} v(i,0),$$

$$\frac{c_1 i + c_2 - g}{\Lambda} - r_2 + \frac{\lambda_1}{\Lambda}\big[v(i+1,1) - r_1\big] + \frac{\mu_1}{\Lambda} v(i,1) + \frac{\mu_2}{\Lambda} v(i,0),$$

$$\frac{c_1 i + c_2 - g}{\Lambda} - r_2 + \frac{\lambda_1}{\Lambda} v(i,1) + \frac{\mu_1}{\Lambda} v(i,1) + \frac{\mu_2}{\Lambda} v(i,0)\Big\} \qquad (3.4a)$$

$$v(i,1) = \min\Big\{ \frac{c_1 i + c_2 - g}{\Lambda} + \frac{\lambda_1}{\Lambda}\big[v(i+1,1) - r_1\big] + \frac{\mu_1}{\Lambda} v((i-1)^+,1) + \frac{\mu_2}{\Lambda} v(i,1),$$

$$\frac{c_1 i + c_2 - g}{\Lambda} + \frac{\lambda_1}{\Lambda} v(i,1) + \frac{\mu_1}{\Lambda} v((i-1)^+,1) + \frac{\mu_2}{\Lambda} v(i,1),$$

$$\frac{c_1 i + c_2 - g}{\Lambda} + \frac{\lambda_1}{\Lambda}\big[v(i+1,1) - r_1\big] + \frac{\mu_1}{\Lambda} v(i,1) + \frac{\mu_2}{\Lambda} v(i,0),$$

$$\frac{c_1 i + c_2 - g}{\Lambda} + \frac{\lambda_1}{\Lambda} v(i,1) + \frac{\mu_1}{\Lambda} v(i,1) + \frac{\mu_2}{\Lambda} v(i,0)\Big\} \qquad (3.4b)$$

The first two terms in the *min* correspond to providing service to class 1 while the third and fourth terms correspond to serving class 2. These terms are further distinguished by the admission option with respect to 1-customers.

From general results in [6], there is a unique solution g, $v(i,j)$ to equations (3.4). The stationary policy determined by the optimality equations is optimal, and the scalar g is the minimum average cost (independent of the starting state).

3.3 Computation of the optimal policy

Policy iteration ([77]) can be applied to determine the optimal policy minimizing the long-run average costs for the truncated queue model in which only a finite buffer is available to type-1 customers. The state space for this truncated queue is

$$S_L = \{(i,j) | 0 \leq i \leq L, j = 0,1\}. \tag{3.5}$$

We refer to the boundary as the subset $B_L = \{(L,0),(L,1)\}$. The optimal policy π_L^{opt} minimizing average costs for the truncated queue is the restriction to S_L of the optimal policy π_∞^{opt} for the infinite queue, provided that the admission region $A_L(\pi_L^{opt})$ – the subset of states $(i,j) \in S_L$ in which π_L^{opt} admits a 1 customer does not touch the boundary B_L.

Figure 3.1 shows the optimal policy minimizing average costs for parameters:

$$
\begin{aligned}
L &= 30, \\
\lambda_1 &= 2.0, \\
\mu_1 &= 2.0, \qquad \mu_2 = 1.33, \\
c_1 &= 1.0, \qquad c_2 = 1.5, \\
r_1 &= 10.0, \qquad r_2 = 2.0,
\end{aligned} \tag{3.6}
$$

in which case $\mu_1 c_1 = 2.0 > \mu_2 c_2 = 1.995$. In this figure, • identifies in the first graph, states where admission of type–1 customers is the optimal action, while in the second graph it identifies states where serving type–2 is optimal.

Fig. 3.1 Optimal policy for parameters in (3.6).

It is observed that the μc–rule is not optimal since there are states $(i, 1)$ in which it is optimal to provide service to type–2 customers.

A similar result holds even if the rewards are identical as seen in Figure 3.2 which corresponds to parameters:

$$
\begin{aligned}
L &= 30, \\
\lambda_1 &= 2.0, \\
\mu_1 &= 2.0, \qquad \mu_2 = 1.33, \\
c_1 &= 1.0, \qquad c_2 = 1.5, \\
r_1 &= 10.0, \qquad r_2 = 10.0.
\end{aligned} \tag{3.7}
$$

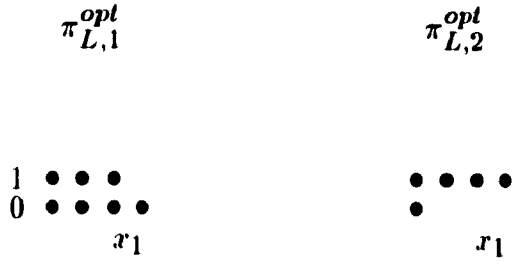

Fig. 3.2 Optimal policy for parameters in (3.7).

We show below that for class-independent holding cost-rates ($c_1 = c_2$) and class-independent rewards ($r_1 = r_2$) long-run average net cost is minimized by the μ-rule coupled to a control-limit admission policy for type-1 customers.

When $c_1 = c_2$, $r_1 = r_2$, and $\mu_2 > \mu_1$, the optimal policy consists in never admitting any type-1 customer and continuously providing service to type 2 customers in steady-state. Consequently, the only recurrent state is $(0, 1)$ and the long-run average cost is $c_2 - \mu_2 r_2$. As for the transient behaviour, starting in state $(i, 1)$ with $i > 0$ it is best to first serve the 2-customer, then to empty the system of the i type-1 customers before devoting service only to 2 customers from then on. The less trivial case $\mu_1 > \mu_2$ is the subject of the following analysis.

3.4 The case of class–independent cost–rate and rewards

From now on, we assume that the holding cost-rates are identical for both classes of customers: $c_1 = c_2 = c$. We also assume that the rewards are identical: $r_1 = r_2 = r$. While the assumption of class–independent cost–rates is essential (in view of the discussion about the non-optimality of the μc–rule in the previous chapter), the assumption of class–independent rewards does not seem necessary for the optimality of the μ–rule, but makes the presentation simpler. Finally, customers from the Poisson stream have shorter mean service times than customers from the infinite pool: $\mu_1 > \mu_2$.

The claim is that the optimal policy minimizing the average cost combines the μ-rule with a control–limit admission policy for type–1 customers. It follows (since $\mu_1 > \mu_2$) that a type–2 customer is served only when the queue is depleted of type 1 jobs. Two cases must be distinguished, according to the policy adopted with respect to type–2 customers:

a) in one case, type–2 jobs are continuously served (one at a time) until an interruption by a type–1 busy period;

b) while in the other case, no 2–customer is ever admitted into the system.

We define below two classes of policies corresponding to the two cases just

mentioned. Within each class, the policy minimizing average costs is determined. That one of these two policies is overall optimal is verified by substitution into the dynamic programming optimality equation.

3.4.1 Two candidates for optimal policy

We denote by Ψ_0 the class of stationary policies which combines the μ rule with a threshold–type admission control for 1–customers, and which never admits a 2–customer. For a policy in Ψ_0, the scheduler which gives priority to type 1 jobs is active only until the type–2 job (if any) present at the beginning has left the queue. From then on, only type–1 jobs are admitted for service and contribute to operating costs.

Similarly, Ψ_1 is the class of stationary policies which combines the μ rule with a threshold–type admission control for 1–customers, and which immediately admits and serves a readily available type–2 job upon service completion of a 2 customer. Since type–1 jobs have preemptive priority, service of the type 2 job might be interrupted by a (random) number of type–1 busy periods. It should be obvious that under a policy in Ψ_1 there is always a type–2 job in the system, either being served or waiting in queue as a result of an interruption by a type 1 busy period.

We point out that the classes Ψ_0 and Ψ_1 of admissible policies do not cover the class of all admissible stationary policies. Their definition is only motivated

by the expectation that the overall optimal policy is either in Ψ_0 or in Ψ_1, its actual position depending on system parameters. Our contribution is to verify, by substitution into the dynamic programming optimality equation, that this is indeed the case.

We first determine the best rules within each class.

3.4.2 Optimal policy in Ψ_0

We denote by $\pi_0(n)$ the stationary policy in Ψ_0 which admits 1–customers according to a control limit policy with threshold n. Under policy $\pi_0(n)$ the set of recurrent states is $\{(0,0),(1,0),...,(n,0)\}$ with steady-state probabilities

$$p(i,0) = \frac{\rho^i - \rho^{i+1}}{1-\rho^{n+1}} \tag{3.8}$$

where $\rho = \lambda_1/\mu_1$. The average cost $\phi_0(n)$ corresponding to $\pi_0(n)$ is

$$\phi_0(n) = c\sum_{k=0}^{n} kp(k,0) - \lambda_1\big[1-p(n,0)\big]r \tag{3.9}$$

which can be expressed as

$$\phi_0(n) = c\left\{\frac{\rho}{1-\rho} - \frac{(n+1)\rho^{n+1}}{1-\rho^{n+1}}\right\} - \mu_1 r\frac{\rho-\rho^{n+1}}{1-\rho^{n+1}} \tag{3.10}$$

It can be shown that $\phi_0(n)$ has a unique minimum. The optimal threshold $n_0 = \arg\min \phi_0(n)$ satisfies

$$\begin{aligned} &\phi_0(n_0) - \phi_0(n_0-1) \le 0, \\ &\phi_0(n_0+1) - \phi_0(n_0) > 0. \end{aligned} \tag{3.11}$$

These inequalities can be combined in

$$\frac{n_0(1-\rho)-\rho(1-\rho^{n_0})}{(1-\rho)^2} \leq \frac{\mu_1 r}{c} < \frac{(n_0+1)(1-\rho)-\rho(1-\rho^{n_0+1})}{(1-\rho)^2}. \qquad (3.12)$$

We denote g_0 the minimum value $\phi_0(n_0)$.

3.4.3 Optimal policy in Ψ_1

Similarly, we denote by $\pi_1(n)$ the stationary policy in Ψ_1 which admits 1 customers according to a control-limit policy with threshold n. Under policy $\pi_1(n)$ the set of recurrent states is $\{(0,1),(1,1),...,(n,1)\}$ with steady- state probabilities

$$p(i,1) = \frac{\rho^i - \rho^{i+1}}{1-\rho^{n+1}}. \qquad (3.13)$$

The average cost $\phi_1(n)$ corresponding to $\pi_1(n)$ is

$$\phi_1(n) = c\sum_{k=0}^{n} kp(k,1) - \lambda_1\big[1-p(n,1)\big]r + c - \mu_2 r p(0,1), \qquad (3.14)$$

which can be expressed as

$$\phi_1(n) = \phi_0(n) + c - \mu_2 r\frac{1-\rho}{1-\rho^{n+1}}. \qquad (3.15)$$

The optimal threshold $n_1 = \arg\min \phi_1(n)$ satisfies

$$\begin{aligned} \phi_1(n_1) - \phi_1(n_1-1) &\leq 0, \\ \phi_1(n_1+1) - \phi_1(n_1) &> 0, \end{aligned} \qquad (3.16)$$

which is equivalent to

$$\frac{n_1(1-\rho)-\rho(1-\rho^{n_1})}{(1-\rho)^2} \leq \frac{(\mu_1-\mu_2)r}{c} < \frac{(n_1+1)(1-\rho)-\rho(1-\rho^{n_1+1})}{(1-\rho)^2}. \qquad (3.17)$$

We denote g_1 the minimum cost $\phi_1(n_1)$.

Summarizing, $\pi_0(n_0)$ is optimal among policies $\pi_0(n)$ in Ψ_0, while $\pi_1(n_1)$ is optimal among policies $\pi_1(n)$ in Ψ_1. The best policy is $\pi_0(n_0)$ or $\pi_1(n_1)$ depending on whether $g_0 < g_1$ or $g_0 > g_1$. If $g_0 = g_1$, $\pi_0(n_0)$ and $\pi_1(n_1)$ are equally good.

3.4.4 The relative values

To establish the overall optimality of g_0 and g_1 we intend to verify the average–cost optimality equation. This requires the computation of the optimal relative values ([77]) corresponding to $\pi_0(n_0)$ and $\pi_1(n_1)$ respectively. This in turn requires that the policies $\pi_0(n_0)$ and $\pi_1(n_1)$ be defined on their respective set of transient states. Since both policies provide service according to the μ–rule, what needs to be specified is the admission rule for 1–customers in transient states. Once $\pi_0(n_0)$ has been extended to all states, its relative values $v_0(i,j)$ can be computed as the expected costs until a first passage into a chosen reference state, $(0,0)$, starting in state (i,j), with holding cost–rate $c(i+j)-g_0$ in state (i,j) and reward r for each admitted customer. Similarly, for $\pi_1(n_1)$ the relative values $v_1(i,j)$ are computed with respect to reference state $(0,1)$ and holding cost–rate $c(i+j)-g_1$ in state (i,j).

3.4.4.1 First passage cost in a simple birth–death process

The candidate optimal policies are obtained by extending $\pi_0(n_0)$ and $\pi_1(n_1)$ to their transient states so that the relative values $v_0(i,j)$ and $v_1(i,j)$ be minimized.

To that end, consider the birth-death process (Figure 3.3) defined on the integers $i \geq 0$, with holding cost-rate $ci - g$ in state i and reward r for each birth.

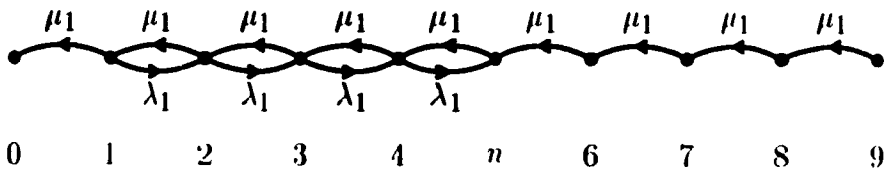

Fig. 3.3 A simple birth-death process on the integers.

The computation of the expected cost $K(i,n,g)$ until a first passage to state 0 when starting in state i is a standard exercise in probability theory.

Case 1: $1 \leq i \leq n$:

$$K(i,n,g) = \frac{c}{\mu_1}\left\{\frac{i}{2}\frac{1}{1-\rho}\left(i + \frac{1+\rho}{1-\rho}\right) - \frac{\rho^{n-i+1} - \rho^{n+1}}{(1-\rho)^2}\left(n + \frac{1}{1-\rho}\right)\right\}$$

$$-\frac{g}{\mu_1}\frac{1}{1-\rho}\left(i - \frac{\rho^{n-i+1} - \rho^{n+1}}{1-\rho}\right) \qquad (3.18a)$$

$$-r\frac{\rho}{1-\rho}\left(i - \frac{\rho^{n-i} - \rho^{n}}{1-\rho}\right),$$

Case 2: $n < i$:

$$K(i,n,g) = \frac{c}{\mu_1}\left(\frac{i-n}{2}\right)(i+n+1) - (i-n)\frac{g}{\mu_1} + K(n,n,g). \qquad (3.18b)$$

One easily obtains

$$K(i,n,g)-K(i,n-1,g)=\begin{cases}\dfrac{\rho^{n-i}-\rho^n}{1-\rho}\dfrac{1}{\mu_1}(nc-g-\mu_1 r), & i<n,\\ \\ \dfrac{\rho-\rho^n}{1-\rho}\dfrac{1}{\mu_1}(nc-g-\mu_1 r), & i\geq n.\end{cases} \tag{3.19}$$

It follows that the optimal n minimizing (for all starting state i) first passage costs to state 0 is determined by

$$nc-g\leq\mu_1 r<(n+1)c-g. \tag{3.20}$$

We point out that if the cost rate in state i is $c(i+1)-g$ rather than $ci-g$, the first passage cost from state i is simply $K(i,n,g-c)$.

3.4.4.2 Relative values corresponding to $\pi_0(n_0)$

Returning to the computation of relative values let us consider first those corresponding to $\pi_0(n_0)$. First, it is obvious that $v_0(i,0)=K(i,n_0,g_0)$ and it is easily verified that equation (3.12) is equivalent to

$$n_0c-g_0\leq\mu_1 r<(n_0+1)c-g_0. \tag{3.21}$$

For the transient states $(i,1)$, $i\geq 0$, the expected first passage cost to $(0,0)$ for a control-limit admission policy for 1-customers with threshold n is

$$w_n(0,1)=\frac{c-g_0}{\mu_2}+\frac{\lambda_1}{\mu_2}\Big\{K(1,n,g_0-c)-r\Big\}, \tag{3.22a}$$

$$w_n(i,1)=K(i,n,g_0-c)+w_n(0,1), \qquad i>0. \tag{3.22b}$$

It is easy to compute

Case 1: $i < n$:

$$w_n(i,1) - w_{n-1}(i,1) = \left(\frac{\rho^{n-i} - \rho^n}{1-\rho} + \frac{\lambda_1}{\mu_2}\rho^{n-1}\right)\frac{1}{\mu_1}\Big[(n+1)c - g_0 - \mu_1 r\Big], \quad (3.23a)$$

Case 2: $n \leq i$:

$$w_n(i,1) - w_{n-1}(i,1) = \left(\frac{\rho - \rho^n}{1-\rho} + \frac{\lambda_1}{\mu_2}\rho^{n-1}\right)\frac{1}{\mu_1}\Big[(n+1)c - g_0 - \mu_1 r\Big]. \quad (3.23b)$$

It is then obvious that the threshold minimizing the expected cost until a first passage to $(0,0)$ from $(i,1)$ is determined by

$$(n+1)c - g_0 \leq \mu_1 r < (n+2)c - g_0, \quad (3.24)$$

which, combined with (3.21), proves that this optimal threshold is $n_0 - 1$. Hence we have

$$v_0(i,0) = K(i, n_0, g_0), \qquad i > 0, \quad (3.25a)$$

$$v_0(0,1) = \frac{c - g_0}{\mu_2} + \frac{\lambda_1}{\mu_2}\Big\{K(1, n_0 - 1, g_0 - c) - r\Big\}, \quad (3.25b)$$

$$v_0(i,1) = K(i, n_0 - 1, g_0 - c) + v(0,1), \qquad i > 0. \quad (3.25c)$$

To summarize, we have shown that the candidate optimal policy corresponding to g_0 is fully characterized by n_0, since the optimal admission threshold in states $(i,0)$ is n_0, while the optimal admission threshold in states $(i,1)$ is $n_0 - 1$.

We will denote π_0^{opt} the policy defined by

$$\pi_0^{opt}(i,0) = \begin{cases} (1,1), & 0 \le i \le n_0, \\ (0,1), & n_0 < i, \end{cases}$$

$$\pi_0^{opt}(i,1) = \begin{cases} (1,0), & i = 0 \\ (1,1), & 0 < i \le n_0 - 1, \\ (0,1), & n_0 \le i. \end{cases} \tag{3.26}$$

Transitions under π_0^{opt} are shown in Figure 3.4.

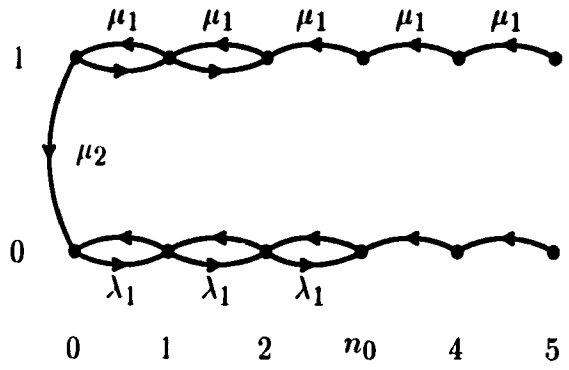

Fig. 3.4 State–transition–rate diagram corresponding to π_0^{opt}.

3.4.4.3 Relative values corresponding to $\pi_1(n_1)$

A similar derivation for $\pi_1(n_1)$ with reference state $(0,1)$ establishes that the optimal admission thresholds for g_1 are n_1+1 in states $(i,0)$ and n_1 in state $(i,1)$. It is also obtained that n_1 is characterized by

$$(n_1+1)c - g_1 \leq \mu_1 r < (n_1+2)c - g_1 \tag{3.27}$$

which is equivalent to (3.17). Finally the relative values $v_1(i,j)$ are

$$v_1(0,0) = -r, \tag{3.28a}$$

$$v_1(i,0) = -r + K(i, n_1+1, g_1), \qquad i > 0, \tag{3.28b}$$

$$v_1(i,1) = K(i, n_1, g_1 - c), \qquad i > 0. \tag{3.28c}$$

We will denote by π_1^{opt} the policy defined by

$$\pi_1^{opt}(i,0) = \begin{cases} (1,0), & i = 0, \\ (1,1), & 1 \leq i \leq n_1+1, \\ (0,1), & n_1+1 < i, \end{cases}$$

$$\pi_1^{opt}(i,1) = \begin{cases} (1,0), & i = 0, \\ (1,1), & 0 < i \leq n_1, \\ (0,1), & n_1 < i. \end{cases} \tag{3.29}$$

Transitions under π_1^{opt} are shown in Figure 3.5.

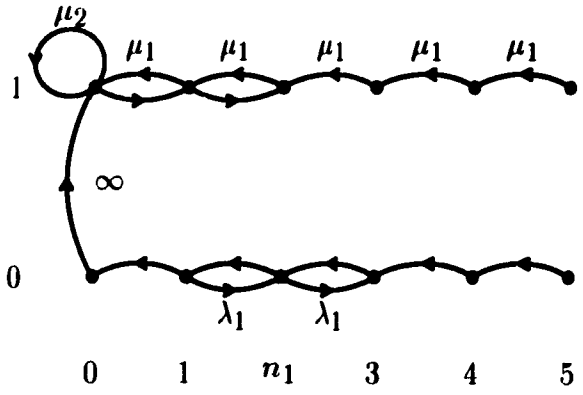

Fig. 3.5 State–transition–rate diagram corresponding to π_1^{opt}.

3.4.5 Verification of optimality

The overall optimality of policies π_0^{opt} and π_1^{opt} is established by showing that under $g_0 < g_1$ the pair (g_0, v_0) satisfies the average-cost optimality equations (3.4), while these equations are satisfied by the pair (g_1, v_1) under $g_1 < g_0$.

The verification involves lengthy and cumbersome algebraic manipulations that will not be presented here. The steps involved in this verification are sketched in Appendix A.

Chapter 4

A numerical investigation of a single–server queue with two job classes†

Using the dynamic programming approach described in section 2.2, we study the case of only *two* independent Poisson arrival processes. An extensive numerical investigation indicates that the optimality of SEJF persists in this case, provided that the flow control is simultaneously optimal. As expected, this jointly optimal flow control is monotonic. These computations also reveal that the boundary between acceptance and rejection regions are very nearly linear. It is shown that, under SEJF scheduling, restricting flow control optimization to these linear boundaries is only slightly suboptimal. It is also observed that the optimal delay/throughput tradeoff is only slightly underestimated by a combination of SEJF scheduling and optimal *window* flow control.

---

† The analysis of this chapter has been published in [19]

4.1 The model

In this chapter, we investigate a special case of the general model described in section 2.1, namely the case of only *two* Poisson arrival streams with intensities λ_1 and λ_2. We recall from section 2.4 that holding cost-rates and admission rewards are assumed independent of class; thus, $c_1 = c_2 = c$, and $r_1 = r_2 = r$. Furthermore we assume here that $\mu_1 > \mu_2$, and define

$$\beta \triangleq \mu_1/\mu_2 > 1. \tag{4.1}$$

4.2 The optimal policy

Extensive computational experiments suggest that the jointly optimal scheduling and flow control minimizing the long-run average net cost for operating the system, combines SEJF preemptive scheduling with monotonic flow controllers.

Conjectured joint optimal flow control and scheduling

scheduling	**SEJF**
flow control for type–1 jobs	**admit iff** $x_1 \leq f_1(x_2)$ **with** f_1 **monotonically decreasing**
flow control for type–2 jobs	**admit iff** $x_2 \leq f_2(x_1)$ **with** f_2 **monotonically decreasing**

Moreover, the optimal flow control for type-1 jobs has the simple form of a threshold on the total number of jobs in the system, that is:

admit a type–1 job in state (x_1, x_2) **iff** $x_1 + x_2 \leq t_1$.

Typical optimal admission policies for type–2 jobs are shown in Figures 4.1 to 4.4. In these figures, • indicates states where admission of type-2 arrivals is the optimal action. Also indicated in each figure is the joint optimal threshold, t_1^{opt}, for type-1 jobs, as well as queue parameters; in particular, $\rho_i = \lambda_i/\mu_i$, $i = 1, 2$. Numbers on the right are values of $b(x_2) = x_1 + \beta x_2$ (function of x_2) on the boundary of the admission region.

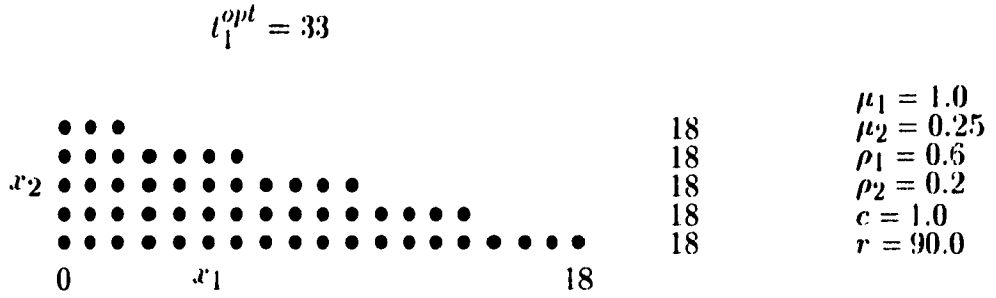

Fig. 4.1 Optimal flow control for type–2 jobs.

It is readily observed from these figures that the boundary between optimal acceptance and rejection regions for type-2 jobs is very nearly of the form

$$x_1 + (\mu_1/\mu_2)x_2 = t_2.$$

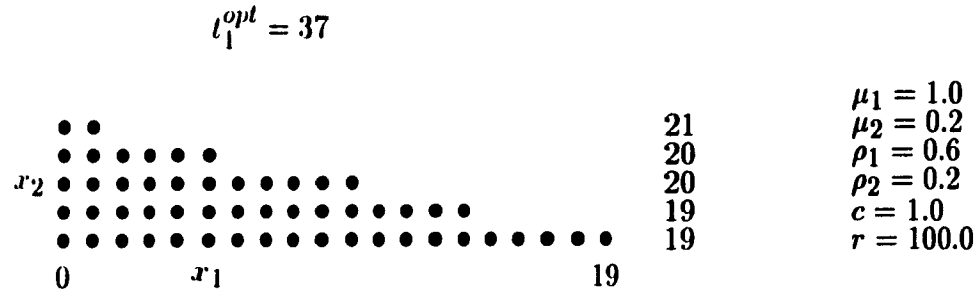

Fig. 4.2 Optimal flow control for type–2 jobs.

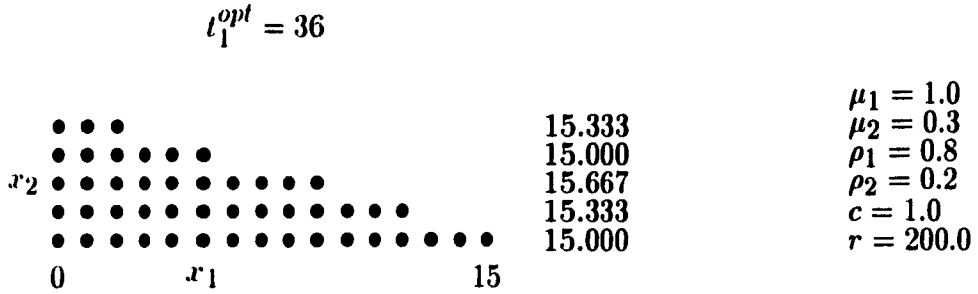

Fig. 4.3 Optimal flow control for type–2 jobs.

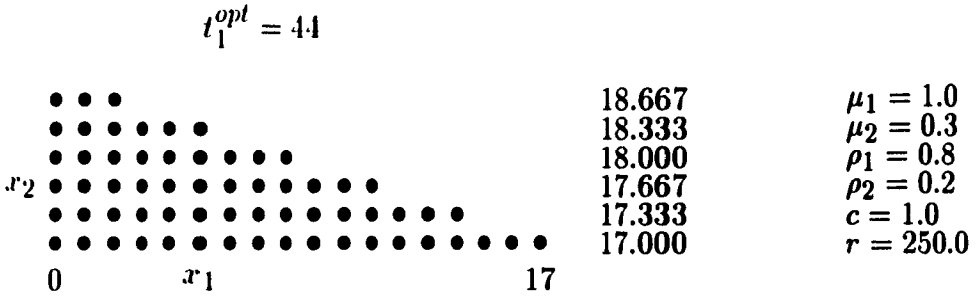

Fig. 4.4 Optimal flow control for type–2 jobs.

Should that boundary be exactly of this form, the optimal flow control would have

the very appealing structure

admit a type–1 job in state (x_1, x_2)	**iff**	$\dfrac{x_1 + x_2}{\mu_1} \leq w_1,$
admit a type–2 job in state (x_1, x_2)	**iff**	$\dfrac{x_1}{\mu_1} + \dfrac{x_2}{\mu_2} \leq w_2,$

for thresholds w_1, w_2. The appeal of such an admission policy comes from the fact that, under SEJF scheduling, the expressions $c(x_1 + 1 + x_2)/\mu_1$ and $(cx_1/\mu_1) + (c(x_2 + 1)/\mu_2)$ are easily interpreted as the accrued cost to the system resulting from the admission of a type–1 job and a type–2 job in state (x_1, x_2) respectively, *in the absence of further arrivals*. The sum $(x_1/\mu_1) + ((x_2 + 1)/\mu_2)$ is also recognized as the expected work in the system. In the absence of arrivals, these accrued costs are compared to the reward r to determine whether it is beneficial for the system to start with one more job.

It is easy to determine if the optimal flow control for type–2 jobs is of the form $x_1 + \beta x_2 \leq t_2$. If β is an integer, threshold t_2 can be restricted to integer values. If β is not an integer, let $\beta = \lfloor \beta \rfloor + (m_\beta/n_\beta)$ where $\lfloor \beta \rfloor$ denotes the integer part of β, and m_β and n_β are integers with $0 < m_\beta < n_\beta$. In this case, t_2 can be restricted to multiples of $1/n_\beta$.

To determine if the optimal type–2 flow controller is of threshold type, one examines the values $b(x_2) = x_1 + \beta x_2$ on the boundary of the admission region.

If β is an integer, a threshold policy is obtained if all these values are equal, the common value being the threshold. It follows that the flow control of Figure 4.2 is not of threshold type while the one of Figure 4.1 is.

If β is not an integer, the sequence $b(0), b(1), b(2), \ldots$ must have $b_{\max} - b_{\min} < 1$, and must be a subsequence of the periodic sequence of period n_β, with values

$$b_{\max} - F((n_\beta - 1)m_\beta/n_\beta), \ldots, b_{\max} - F(2m_\beta/n_\beta), b_{\max} - F(m_\beta/n_\beta), b_{\max},$$

where $F(m/n)$ denotes the fractional part of the rational number m/n. The threshold is then $b_{\max}$. It follows that the optimal flow control in Figure 4.3 ($\beta = 3\frac{1}{3}$) is of threshold type, with threshold equal to $15\frac{2}{3}$, while the optimal flow control in Figure 4.4 is not.

Under SEJF service, we expect that it cannot be optimal to admit a *long* job in a given state if it is optimal to reject a *short* job in this state (holding cost and reward being class-independent); this means that we expect the admission region of type-2 jobs to be included in the admission region of type-1 jobs. This was indeed verified throughout our computations and is obvious in the examples considered here. When the optimal flow control for type-2 jobs is of the form $x_1 + \beta x_2 \leq t_2^{opt}$, this translates to $\lfloor t_2^{opt} \rfloor \leq t_1^{opt}$.

These observations motivate us to study our two-class single-server queue under SEJF preemptive scheduling, to determine the flow control minimizing, within the class of threshold policies, the long-run average cost.

4.3 The optimal threshold–type flow control under SEJF scheduling

We apply SEJF preemptive scheduling to our single–server queue to determine the flow control minimizing long–run average cost. The minimization is restricted to the class of threshold–type flow controls, namely those specified by thresholds t_1, and t_2:

Threshold–type flow control

admit a type–1 job in state (x_1, x_2) **iff** $x_1 + x_2 \leq t_1$,

admit a type–2 job in state (x_1, x_2) **iff** $x_1 + \beta x_2 \leq t_2$.

Given a threshold–type flow control, the corresponding set of recurrent states for the queue process under SEJF scheduling is easily determined. The mean number of jobs in queue $\overline{x}_i$ and the mean throughput $\overline{\gamma}_i$ for type i jobs ($i = 1, 2$) are computed by solving the system of global–balance linear equations for the steady–state probabilities.

The best thresholds are obtained by minimizing the expected net cost

$$c(\overline{x}_1 + \overline{x}_2) - r(\overline{\gamma}_1 + \overline{\gamma}_2)$$

as a function of the thresholds t_1 and t_2.

This optimization assumes that the net cost has a dependence on t_1 and t_2 that

is *discretely unimodular*, meaning that a local minimum is a global minimum. This would obviously be the case if both $\overline{x} = \overline{x}_1 + \overline{x}_2$ and $\overline{\gamma} = \overline{\gamma}_1 + \overline{\gamma}_2$ were monotonically increasing functions of t_1 and t_2. $\overline{x}$ is indeed monotonically increasing in t_1 and t_2; on the other hand, while $\overline{\gamma}$ is also increasing in t_1, it may be slightly decreasing in t_2 over a certain range. These properties were observed numerically. The behaviour versus t_2, for t_1 fixed, can be explained by the fact that, in increasing t_2 by a minimal amount $1/n_\beta$, we increase the probability of trading a short job for a long job. This trade maintains the number of customers. On the other hand, the loss of throughput for short jobs may outweigh the gain of throughput for long jobs, resulting in a net decrease of total throughput.

Figures 4.5 and 4.6 display the best threshold-type flow controls corresponding to the non-threshold-type admission policies of Figures 4.2 and 4.4 respectively.

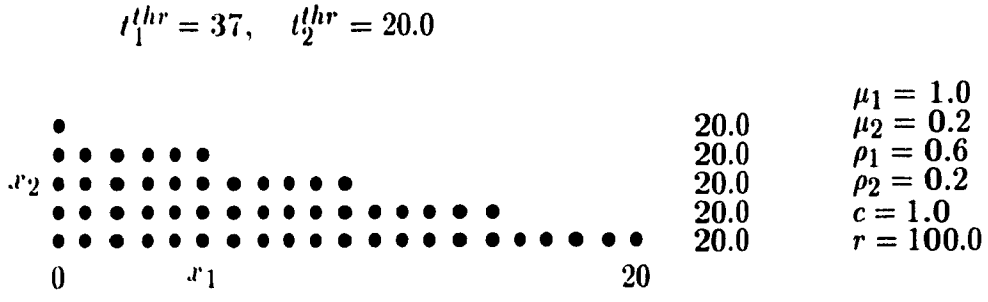

Fig. 4.5 Best threshold-type flow control corresponding to optimal control of Figure 4.2.

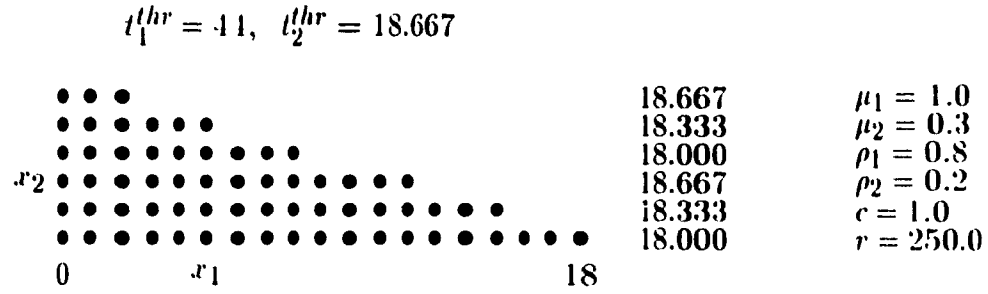

Fig. 4.6 Best threshold–type flow control corresponding to optimal control of Figure 4.4.

As a typical example, the performance of the queue with parameters

$$\mu_1 = 1.0, \qquad \rho_1 = 0.95, \qquad c = 1.0,$$

$$\mu_2 = 0.2, \qquad \rho_2 = 0.40, \qquad r = 300.0,$$

is presented in Figures 4.7 to 4.14. The best threshold-type flow control and its performance are

$$t_1^{thr} = 27, \qquad \overline{\gamma} = 0.9436,$$

$$t_2^{thr} = 6, \qquad \overline{D} = \overline{x}/\overline{\gamma} = 12.098,$$

$$P = r\overline{\gamma} - c\overline{x} = 271.67,$$

where superscript *thr* emphasizes the fact that the optimization was restricted to threshold–type admission policies.

Figures 4.7 and 4.8 show the variation $\overline{x}$ with t_1 and t_2; Figures 4.9 and 4.10

are similar plots for $\overline{\gamma}$. The striking feature of these plots is the non–smooth variation of $\overline{x}$ and $\overline{\gamma}$ versus t_2. We also note the slight decrease of $\overline{\gamma}$ for increasing t_2. The profit $P = r\overline{\gamma} - c\overline{x}$ is plotted in Figures 4.11 and 4.12 as a function of the thresholds. These level curves were selected to include the maximum point. Figure 4.13 is a perspective plot of the profit versus t_1 and t_2 in the range $0 \leq t_2 \leq 30$, $t_2 \leq t_1 \leq t_2 + 30$. Of course $\overline{x}$, $\overline{\gamma}$, and P all are discrete functions of t_1 and t_2; continuous plots are used for graphical clarity. We point out that level curves in Figure 4.13 correspond to constant values of x_1 and $x_1 - x_2$.

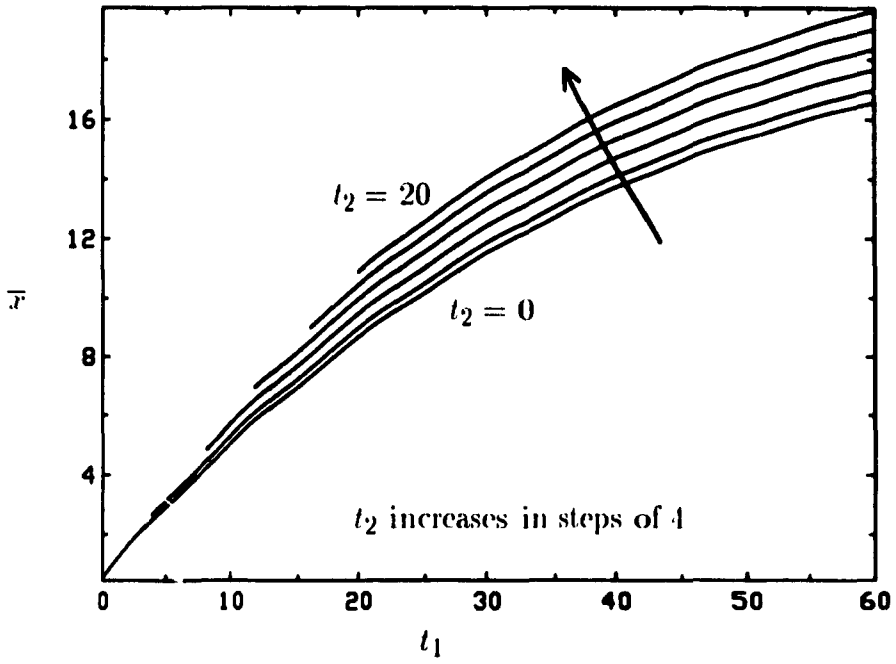

Fig. 4.7 Number of jobs as function of t_1.

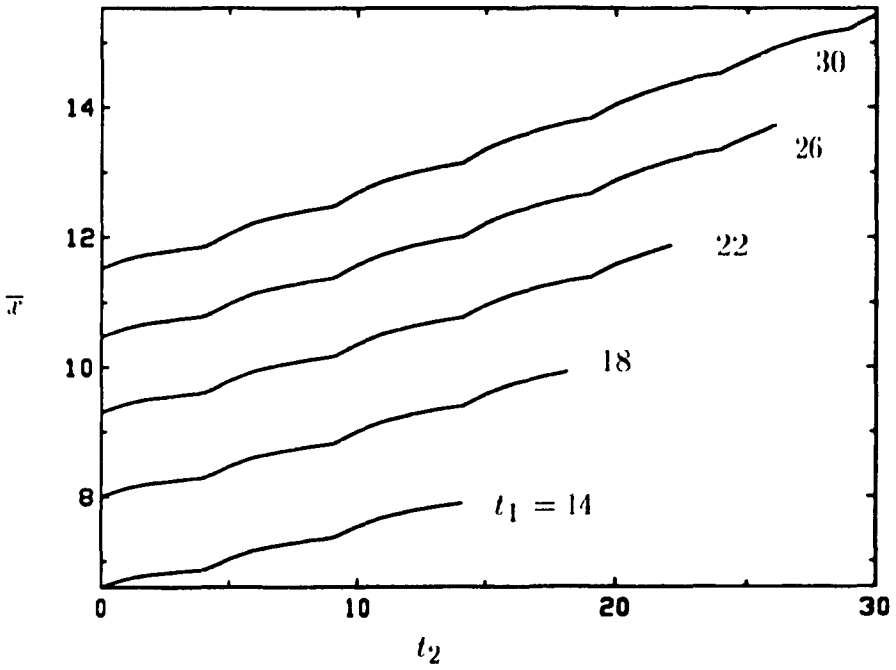

Fig. 4.8 Number of jobs as function of t_2.

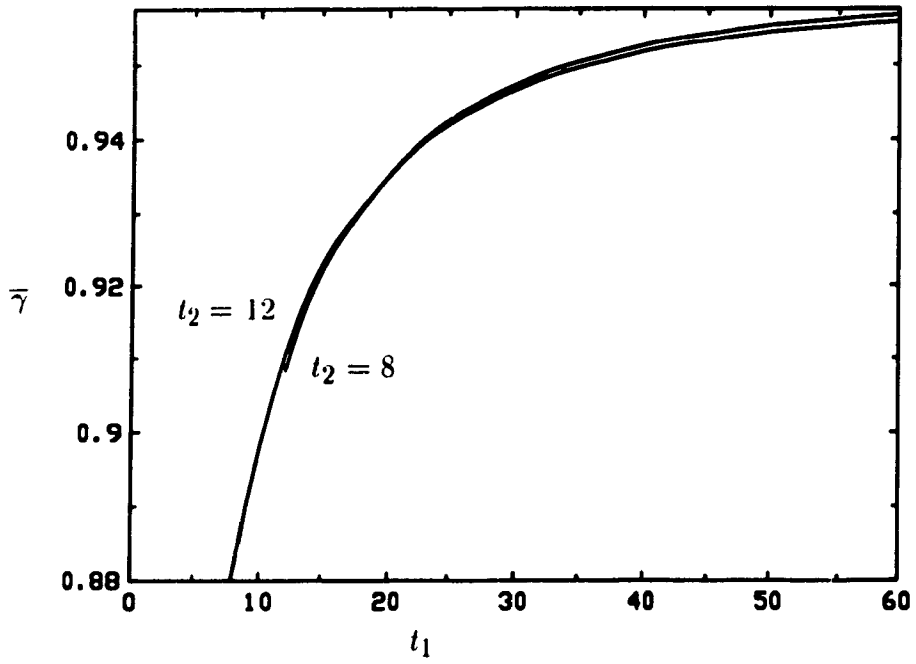

Fig. 4.9 Throughput as function of t_1.

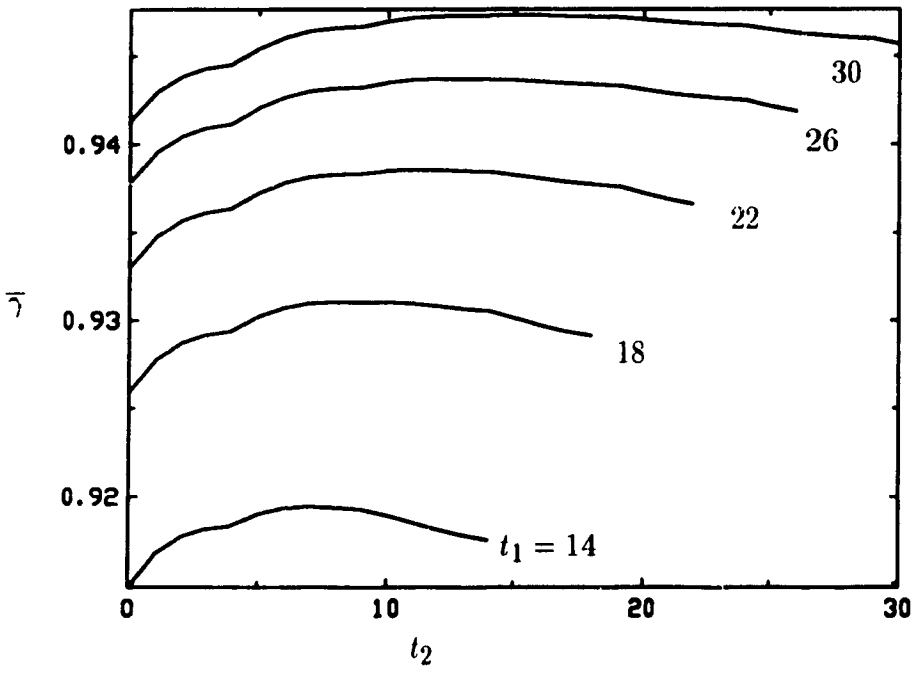

Fig. 4.10 Throughput as function of t_2.

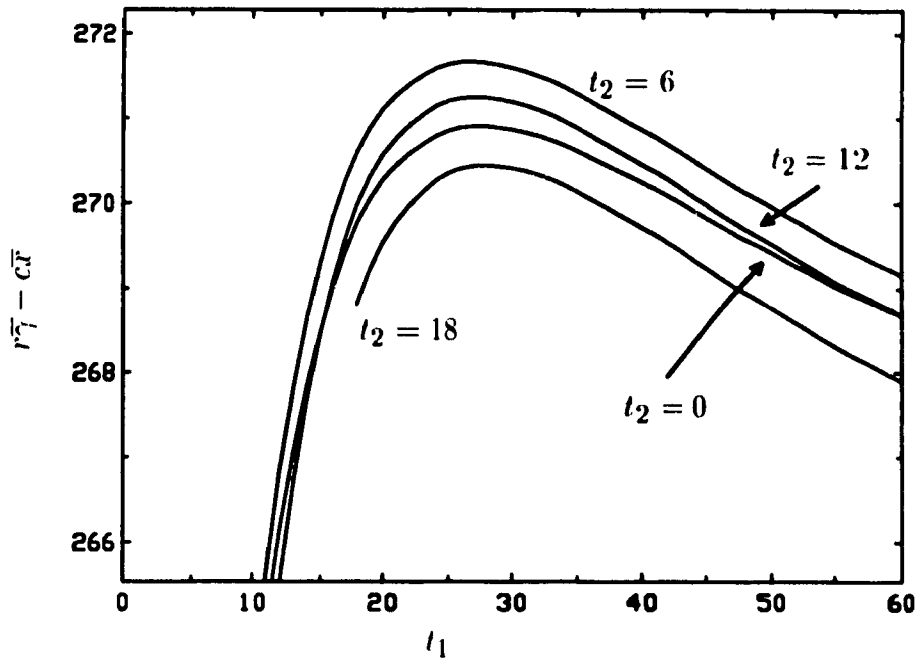

Fig. 4.11 Profit as function of t_1.

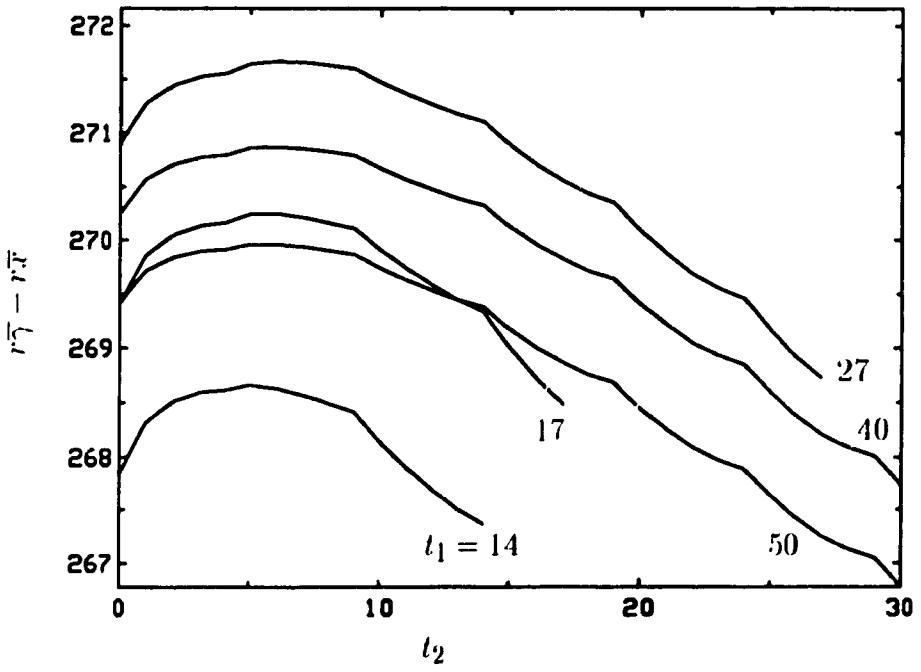

Fig. 4.12 Profit as function of t_2.

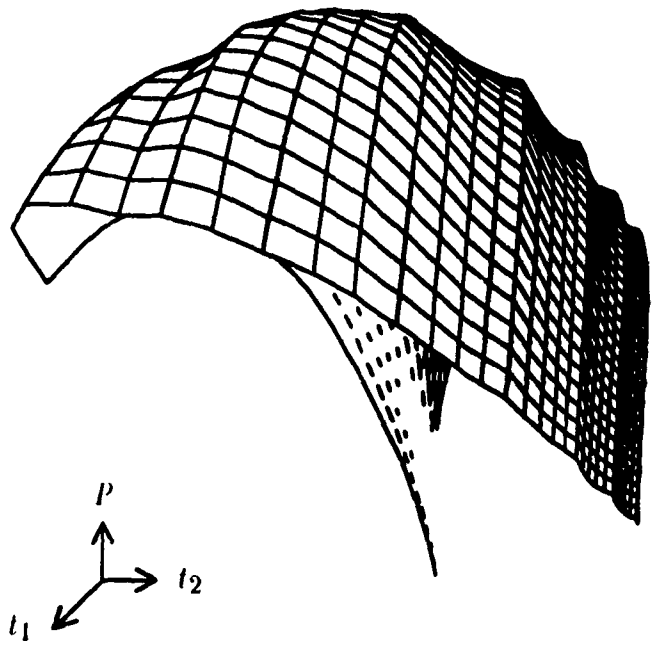

Fig. 4.13 Plot of profit versus thresholds t_1 and t_2.

4.4 Comparison between optimal and threshold–type flow control under SEJF scheduling

As mentioned in the introduction, in the particular case that costs and rewards are class–independent, the optimization problem amounts to determining the delay/throughput characteristic of the queue. Figures 4.14 and 4.15 are plots of the optimal delay/throughput tradeoff. These curves were obtained by varying the reward r with all others parameters kept fixed, (in particular, $c = 1.0$) and computing for each value of r the optimal joint controller and its performance.

If $\mu_1 = \mu_2$, since holding cost–rates and rewards are class–independent, customers form a single class; the optimal server only has to be conservative while the optimal flow controller admits according to a threshold on the total number of customers in queue. Lazar [42] derived an expression for the delay/throughput curve for this queue. The effect of $\beta = \mu_1/\mu_2$ is shown in Figure 4.15.

Tables 4.1, 4.2, and 4.3 compare the delay/throughput performance for optimal and best threshold type flow controls. In these tables, $\overline{D}$ refers to delay averaged over all customers. Superscript *opt* obviously refers to the truly optimal values; superscript *thr* refers to the best threshold type flow control under SEJF scheduling. All values were computed with parameters μ_1 and c set to 1.0. In these tables • indicates cases where the optimal tradeoff is achieved by a threshold–type flow control. Threshold t_1 can be restricted to integer values; generally $t_1 \geq 0$ unless it is optimal to close the system to type 1 customers, a

possibility described by $t_1 = -1$. Shutting down the system cannot be optimal if $r > c/\mu_1$, a condition that we assume throughout. If β is an integer, threshold t_2 can also be restricted to integer values; generally $t_2 \geq 0$ unless it is optimal to close the system to type 2 customers, a possibility described by $t_2 = -1$. Data is presented in the form of tables because corresponding delay/throughput curves would be hardly distinguishable.

It is observed from this comparison that the class of threshold-type flow controls is only slightly suboptimal. This fact will be exploited in Chapter 5 to approximate the controlled process; this approximate-process has easily computable optimal thresholds.

We point out that occurances of equality between optimal and best-threshold policies in terms of delay and throughput performance when the optimal policy is not of threshold type are due to roundoff.

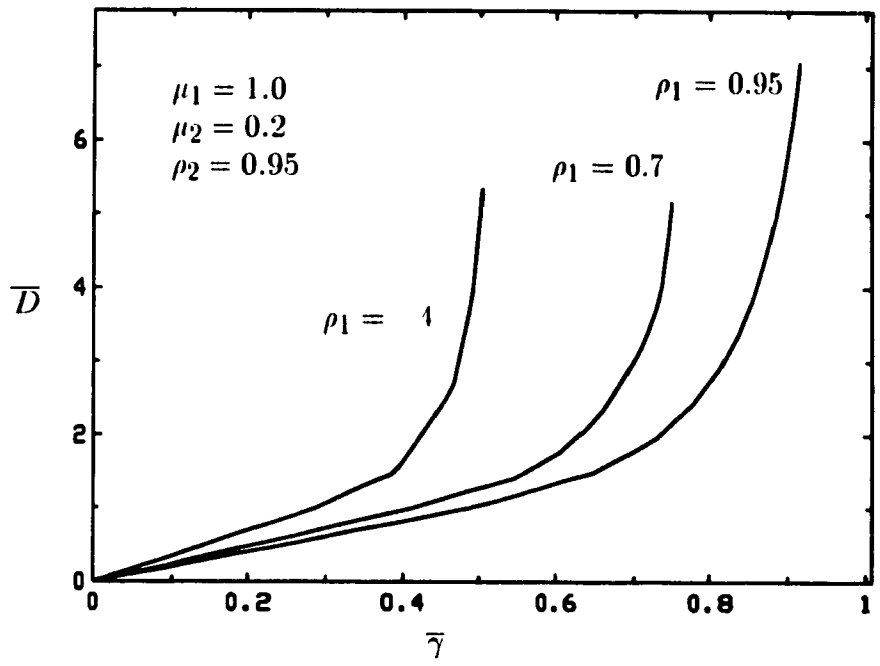

Fig. 4.14 Delay/throughput tradeoff curves.

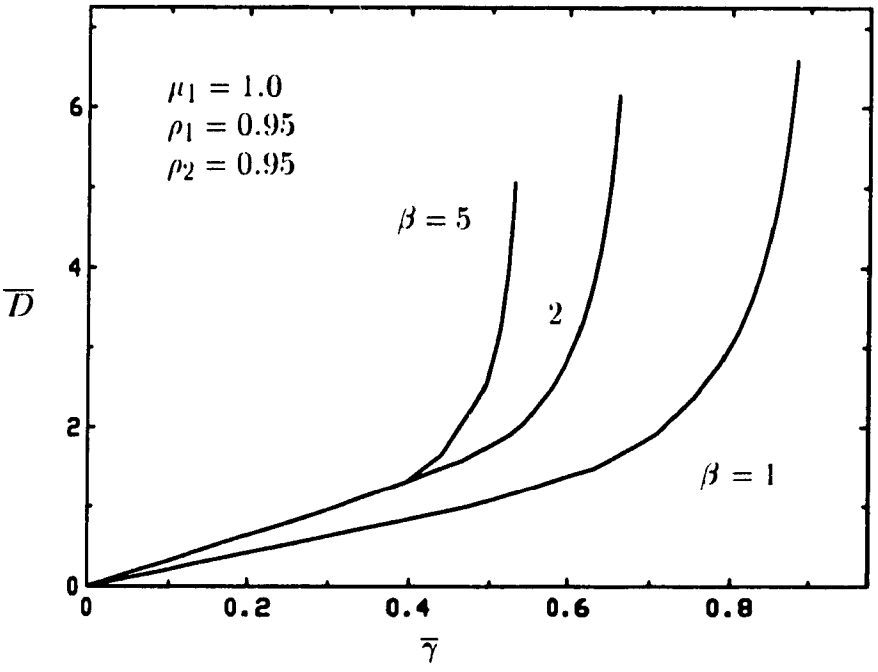

Fig. 4.15 Delay/throughput tradeoff curves.

ρ_1	ρ_2	r	t_1^{opt}	t_1^{thr}	t_2^{thr}	$\overline{\gamma}^{opt}$	$\overline{\gamma}^{thr}$	$\overline{D}^{opt}$	$\overline{D}^{thr}$	
.40	.40	10	3	3	3.11	.6579	–	2.1859	–	•
		20	6	6	5.55	.7189	–	3.1352	–	•
		50	14	14	12.22	.7521	–	4.3630	–	•
		80	22	22	18.33	.7577	–	4.7947	–	•
		100	26	26	22.33	.7591	–	4.9468	–	•
		200	50	50	42.33	.7600	–	5.0841	–	•
.40	.95	10	2	2	2.11	.7631	–	2.0614	–	•
		20	4	4	3.33	.8572	–	3.3438	–	•
		50	8	8	5.22	.8895	–	4.3703	–	•
		80	11	11	6.44	.9053	–	5.2668	–	•
		100	13	13	7.00	.9147	–	6.0620	–	•
		200	20	20	9.00	.9268	–	7.8785	–	•
		400	34	34	11.22	.9330	–	9.7951	–	•
		600	48	48	12.66	.9349	–	10.7824	–	•
.95	.40	10	2	2	1.11	.8160	–	2.1139	–	•
		20	3	3	2.22	.8720	–	2.7766	–	•
		50	6	6	4.11	.9359	–	4.5283	–	•
		80	8	8	5.00	.9552	–	5.7356	–	•
		100	8	8	5.33	.9570	–	5.8993	–	•
		200	12	12	6.66	.9749	–	8.3389	–	•
		400	16	16	8.11	.9824	–	10.5233	–	•
		600	19	19	9.00	.9859	–	12.2484	–	•
		1000	24	24	10.11	.9890	–	14.6853	–	•
		2000	33	33	11.77	.9916	–	18.3373	–	•
.95	.95	10	2	2	1.11	.8729	–	2.2368	–	•
		20	3	3	2.00	.9058	–	2.7443	–	•
		50	5	5	2.22	.9502	–	4.0175	–	•
		80	6	6	3.33	.9640	–	4.9186	–	•
		100	7	7	3.33	.9693	–	5.3847	–	•
		200	9	9	4.22	.9763	–	6.3625	–	•
		400	13	13	4.44	.9847	–	8.5087	–	•
		600	16	16	5.33	.9873	–	9.8049	–	•
		1000	20	20	5.55	.9900	–	11.7976	–	•
		2000	29	29	6.66	.9925	–	15.4178	–	•

Table 4.1 Optimal versus best–threshold delay and throughput for $\mu_2 = 0.9$, that is $\beta = 1\frac{1}{9}$

ρ_1	ρ_2	r	t_1^{opt}	t_1^{thr}	t_2^{thr}	$\overline{\gamma}^{opt}$	$\overline{\gamma}^{thr}$	$\overline{D}^{opt}$	$\overline{D}^{thr}$	
.40	.40	10	5	5	0	.4330	–	2.1633	–	•
		20	11	11	3	.4456	–	2.4333	–	•
		30	16	16	7	.4606	.4606	3.0863	3.0868	
		50	27	27	12	.4682	.4682	3.6718	3.6718	
		80	43	43	20	.4741	.4741	4.4205	4.4250	
		100	53	52	25	.4761	.4761	4.7814	4.7853	
.40	.95	10	5	5	0	.4569	–	2.4847	–	•
		20	10	10	2	.4690	–	2.7390	–	•
		30	16	16	5	.4875	.4875	3.6606	3.6621	
		50	26	26	7	.4925	.4925	3.9775	3.9777	
		80	41	41	11	.5032	.5032	5.2488	5.2497	
		100	51	51	12	.5037	.5037	5.3378	5.3381	
.95	.40	10	3	3	−1	.7790	–	2.4359	–	•
		20	5	5	−1	.8343	–	3.3506	–	•
		50	9	9	0	.8885	–	5.2333	–	•
		100	14	14	2	.9177	–	7.3916	–	•
		200	21	21	5	.9357	.9358	10.0949	10.1208	
		400	32	32	7	.9477	.9477	13.5369	13.5491	
		600	42	42	9	.9528	.9525	16.0888	15.9048	
		800	51	51	11	.9553	.9553	17.8289	17.8667	
.95	.95	10	3	3	−1	.7790	–	2.4359	–	•
		20	5	5	−1	.8343	–	3.3506	–	•
		50	9	9	0	.8917	–	5.3451	–	•
		100	13	13	1	.9160	–	7.0623	–	•
		200	21	21	3	.9367	–	10.0016	–	•
		400	32	32	5	.9487	.9487	13.5003	13.5116	
		600	42	42	6	.9535	.9535	15.8838	15.8896	
		800	51	51	7	.9558	.9558	17.4987	17.5075	

Table 4.2 Optimal versus best–threshold delay and throughput for $\mu_2 = 0.2$, that is $\beta = 5$.

ρ_1	ρ_2	r	t_1^{opt}	t_1^{thr}	t_2^{thr}	$\overline{\gamma}^{opt}$	$\overline{\gamma}^{thr}$	$\overline{D}^{opt}$	$\overline{D}^{thr}$	
.40	.40	10	5	5	−1	.3990	–	1.6420	–	•
		20	11	11	1	.4214	–	2.4512	–	•
		30	17	17	4	.4233	–	2.5461	–	•
		40	23	23	7	.4234	–	2.5520	–	•
		60	34	34	14	.4310	.4310	3.3013	3.3013	
		80	45	45	20	.4333	.4333	3.6902	3.6904	
		100	57	57	25	.4347	.4347	3.9402	3.9402	
.40	.95	10	5	5	−1	.3990	–	1.6420	–	•
		20	11	11	1	.4339	–	2.8724	–	•
		30	17	17	3	.4355	–	2.9534	–	•
		40	22	22	4	.4356	–	2.9612	–	•
		60	34	34	10	.4444	.4444	4.0221	4.0223	
		80	45	45	12	.4469	.4470	4.3643	4.3643	
		100	56	56	14	.4472	.4472	4.4058	4.4058	
.95	.40	10	3	3	−1	.7790	–	2.4359	–	•
		20	5	5	−1	.8343	–	3.3506	–	•
		30	6	6	−1	.8514	–	3.7953	–	•
		50	9	9	−1	.8840	–	5.0787	–	•
		100	14	14	−1	.9107	–	7.0519	–	•
		200	22	22	2	.9322	–	10.1777	–	•
		400	34	34	5	.9439	–	13.5906	–	•
		600	45	45	7	.9484	–	15.8554	–	•
		800	55	55	12	.9512	.9512	17.8367	17.8649	
.95	.95	10	3	3	−1	.7790	–	2.4359	–	•
		20	5	5	−1	.8343	–	3.3506	–	•
		30	6	6	−1	.8514	–	3.7953	–	•
		50	9	9	−1	.8840	–	5.0787	–	•
		100	14	14	−1	.9107	–	7.0519	–	•
		200	21	21	1	.9314	–	9.9285	–	•
		400	31	31	3	.9445	–	13.6474	–	•
		600	41	44	5	.9487	–	15.7451	–	•
		800	51	54	5	.9509	–	17.2697	–	•

Table 4.3 Optimal versus best–threshold delay and throughput for $\mu_2 = 0.1$, that is $\beta = 10$.

4.5 The optimal window flow control under SEJF scheduling

Window flow control is widely used is practice and has been shown to be optimal in a variety of context, ([34],[59]). A window flow control assigns finite separate buffers to the arrival streams; that is

Window flow control

admit a type–1 job in state (x_1, x_2) **iff** $x_1 \leq b_1$,

admit a type–2 job in state (x_1, x_2) **iff** $x_2 \leq b_2$.

As for threshold-type flow control, the best window sizes b_1 and b_2 under SEJF scheduling are obtained by minimizing the expected net cost $c(\overline{x}_1 + \overline{x}_2) - r(\overline{\gamma}_1 + \overline{\gamma}_2)$ as a function of b_1 and b_2. This computation assumes that this net cost has a dependence on b_1 and b_2 that is *discretely unimodular*, meaning that a local minimum is a global minimum.

4.6 Comparison between optimal and window flow control under SEJF scheduling

Tables 4.4, 4.5, and 4.6 compare the delay/throughput performance for optimal and best window flow controls. In these tables, $\overline{D}$ refers to delay averaged over all customers. Superscript *opt* obviously refers to the truly optimal values; superscript *win* refers to the best window flow control under SEJF scheduling.

t_2^{bnd} is the minimum value of t_2 such that the overall optimal admission region for type 2 jobs is included in $\{(x_1, x_2)|x_1 + \beta x_2 \leq t_2\}$. All values were computed with parameters μ_1 and c set to 1.0.

This comparison shows that the overall optimal delay/throughput tradeoff can be achieved very closely by a combination of preemptive SEJF scheduling and a window flow control. This observation is an indication of the relative value of detailed state feedback in the control of the system. In the present context for example, the optimal scheduling is a *static* priority rule. Moreover, under this optimal scheduling, the improvement obtained by the use of state feedback to achieve an optimal tradeoff between delay and throughput is marginal.

ρ_1	ρ_2	r	t_1^{opt}	b_1	t_2^{bnd}	b_2	$\overline{\gamma}^{opt}$	$\overline{\gamma}^{win}$	$\overline{D}^{opt}$	$\overline{D}^{win}$
.10	.40	10	1	3	2	1	.5283	.5428	2.3495	2.6232
		20	9	6	5	2	.5633	.5661	3.0574	3.1682
		30	13	10	8	3	.5800	.5773	3.6828	3.5783
		10	17	10	13	4	.5861	.5842	4.0228	3.9224
		60	26	11	18	7	.5928	.5942	4.5310	4.6674
		100	42	42	23	11	.5982	.5983	5.1598	5.1891
.40	.95	10	4	1	3	0	.5601	.5627	2.1887	2.2379
		20	8	6	3	1	.6224	.6215	3.0975	3.1811
		10	15	12	5	2	.6519	.6532	4.0561	4.1570
		60	22	18	7	3	.6684	.6695	5.0698	5.1970
		80	28	23	8	4	.6769	.6796	5.9079	6.3028
		100	35	30	9	4	.6789	.6796	6 1683	6.3028
		120	41	10	10	4	.6838	.6796	6 9732	6.3028
		150	51	38	11	5	.6856	.6861	7.3265	7.4661
.95	.10	10	3	3	0	−1	.8003	.7790	2 5373	2.4359
		10	7	6	2	0	.8940	.8896	4.5972	4.5424
		80	10	9	3	1	.9201	.9257	6.0307	6.6924
		100	12	10	4	1	.9320	.9303	7 0764	7.1119
		200	18	15	5	2	.9492	.9197	9.5601	9.9753
		400	27	24	7	2	.9613	.9602	13.0439	12.9668
		600	31	30	8	3	.9658	.9657	15.2921	15.5677
		800	10	37	9	3	.9683	.9680	16.9990	17.1893
.95	.95	10	2	2	0	0	.7702	.8179	2.1547	2 7534
		20	1	3	1	0	.8530	.8509	3.1756	3.2275
		30	5	5	1	0	.8734	.8882	3.6251	4.1450
		10	7	6	1	0	.8994	.8998	4.5015	4.5894
		60	8	8	2	0	.9150	.9160	5.2263	5.4514
		80	10	8	2	1	.9276	.9287	6.0645	6.3863
		100	11	9	3	1	.9339	.9337	6.6339	6.8037
		200	17	15	4	1	.9521	.9509	9.2416	9.1314
		400	25	22	5	2	.9624	.9625	12.1830	12 4639
		800	39	36	6	2	.9695	.9690	16.2838	16.1017

Table 4.4 Optimal versus best–window delay and throughput for $\mu_2 = 0.5$, that is $\beta = 2$.

ρ_1	ρ_2	r	t_1^{opt}	b_1	t_2^{bnd}	b_2	$\overline{\gamma}^{opt}$	$\overline{\gamma}^{win}$	$\overline{D}^{opt}$	$\overline{D}^{win}$
.40	.40	10	5	4	0	0	.4330	.4437	2.1633	2.4015
		20	11	10	3	0	.4456	.4460	2.4333	2.4483
		30	16	14	7	1	.4606	.4612	3.0863	3.1330
		50	27	24	12	2	.4682	.4686	3.6718	3.7142
		100	53	39	25	4	.4761	.4752	4.7814	4.6028
.40	.95	10	5	1	0	0	.4569	.4675	2.4847	2.7252
		20	10	9	2	0	.4690	.4696	2.7390	2.7673
		30	16	11	5	1	.4875	.4929	3.6606	4.0180
		50	26	21	7	1	.4925	.4929	3.9775	4.0180
		80	11	11	11	2	.5032	.5039	5.2488	5.3836
		100	51	51	12	2	.5037	.5039	5.3378	5.3836
.95	.40	10	3	3	−1	−1	.7790	.7790	2.4359	2.4359
		20	5	5	−1	−1	.8343	.8343	3.3506	3.3506
		50	9	9	0	−1	.8885	.8840	5.2333	5.0787
		100	11	13	2	0	.9177	.9180	7.3916	7.5300
		200	21	20	5	0	.9357	.9349	10.0950	10.0194
		100	32	30	7	1	.9477	.9480	13.5369	13.8628
		600	42	10	10	1	.9528	.9526	16.0888	16.0979
		800	51	49	11	1	.9553	.9548	17.8289	17.6273
.95	.95	10	3	3	−1	−1	.7790	.7790	2.4359	2.4359
		50	9	9	0	−1	.8917	.8840	5.3451	5.0787
		100	13	12	1	0	.9160	.9164	7.0623	7.2015
		200	21	20	3	0	.9367	.9367	10.0016	10.0587
		400	32	31	5	0	.9487	.9475	13.5003	13.2039
		800	51	19	7	1	.9558	.9559	17.4987	17.6941

Table 4.5 Optimal versus best–window delay and throughput for $\mu_2 = 0.2$, that is $\beta = 5$.

ρ_1	ρ_2	r	t_1^{opt}	b_1	t_2^{bnd}	b_2	$\overline{\gamma}^{opt}$	$\overline{\gamma}^{win}$	$\overline{D}^{opt}$	$\overline{D}^{win}$
.40	.40	10	5	5	−1	−1	.3990	.3990	1.6420	1.6420
		20	11	10	1	0	.4214	.4234	2.4512	2.5520
		40	23	22	7	0	.4234	.4234	2.5520	2.5521
		60	34	32	14	1	.4310	.4311	3.3013	3.3388
		100	57	10	25	2	.4317	.4317	3.9402	3 9429
.10	.95	10	5	5	−1	−1	.3990	.3990	1.6420	1.6420
		20	11	10	1	0	.4339	.4356	2.8724	2.9653
		40	22	21	4	0	.4356	4356	2.9613	2.9657
		60	31	32	10	1	.4444	.4472	4.0221	4.4123
		80	15	15	12	1	.4469	.4472	4.3643	4.4123
		100	56	56	11	1	.4172	.4472	1.1058	4.4123
.95	.10	10	3	3	−1	−1	.7790	.7790	2.4359	2.4359
		10	8	8	−1	−1	.8754	.8751	4.6593	4.6593
		100	11	11	−1	−1	.9107	.9107	7.0519	7.0519
		200	22	21	2	0	.9322	.9326	10.1777	10.3739
		400	31	33	7	0	.9439	.9440	13.5906	13.6973
		600	15	11	7	0	.9184	.9184	15.8554	15.9314
		800	55	51	12	0	.9512	.9505	17.8367	17.4171
.95	.95	20	5	5	−1	−1	.8343	.8313	3.3506	3 3506
		100	11	11	−1	−1	.9107	.9107	7.0519	7.0519
		200	21	21	1	0	.9314	.9334	9.9285	10.4306
		400	31	31	3	0	.9445	.9446	13 6174	13 7415
		600	11	13	5	0	.9187	.9487	15.7451	15.7958
		800	51	53	5	0	.9509	.9509	17.2697	17.3254

Table 4.6 Optimal versus best–window delay and throughput for $\mu_2 = 0.1$, that is $\beta = 10$.

4.7 Sensitivity of performance to scheduling

In the previous sections we have examined the sensitivity of performance to flow control by comparing the performance under SEJF scheduling (the optimal rule) coupled to: 1) the overall optimal joint flow control, 2) the best threshold-type flow control, and 3) the best window-type flow control. Here we study the sensitivity to scheduling by comparing the optimal performance to the performance under a Random Order-of-Service scheduling rule (ROS) coupled to the optimal flow control for such a service policy.

In our model with two classes of jobs, ROS is specified by the probability α that the server, when it turns free at a service completion epoch, is assigned to a type 1 job. It is assumed that ROS is activated only when there are jobs of each type queued. Otherwise, the server operates on a first-come-first-served basis.

Preemption is not allowed. In particular, ROS with $\alpha = 1$ is not identical to preemptive SEJF (assuming $\mu_1 > \mu_2$ as usual) because a type-1 job arriving to a queue of type 2 jobs only, is not allowed to preempt the type-2 job in service, as it is under preemptive SEJF scheduling.

Flow control under ROS is subject to optimization. The goal is to determine the flow control minimizing long-run average cost under a cost/benefit structure identical to the overall joint optimization problem of flow control and scheduling, that is with holding cost rate c and admission reward r both independent of class.

Performing uniformization with total event rate $\Lambda = \lambda_1 + \lambda_2 + \mu_1 + \mu_2$, the average cost dynamic programming optimality equation is

$$\begin{aligned} v(x_1,x_2) =& \frac{c(x_1+x_2)-g}{\Lambda} + \frac{\mu_2}{\Lambda} v(x_1,x_2) \\ &+ \frac{\lambda_1}{\Lambda} \min\{v(x_1+1,x_2)-r, v(x_1,x_2)\} \\ &+ \frac{\lambda_2}{\Lambda} \min\{v(x_1,x_2+1)-r, v(x_1,x_2)\} \\ &+ \frac{\mu_1}{\Lambda}\{[\alpha v(x_1-1,x_2)+(1-\alpha)w(x_1-1,x_2)]I(x_1>1) \\ &\quad + w(0,x_2)I(x_1=1)\}, \end{aligned} \tag{4.2a}$$

$$\begin{aligned} w(x_1,x_2) =& \frac{c(x_1+x_2)-g}{\Lambda} + \frac{\mu_1}{\Lambda} w(x_1,x_2) \\ &+ \frac{\lambda_1}{\Lambda} \min\{w(x_1+1,x_2)-r, w(x_1,x_2)\} \\ &+ \frac{\lambda_2}{\Lambda} \min\{w(x_1,x_2+1)-r, w(x_1,x_2)\} \\ &+ \frac{\mu_2}{\Lambda}\{[\alpha v(x_1,x_2-1)+(1-\alpha)w(x_1,x_2-1)]I(x_2>1) \\ &\quad + v(x_1,0)I(x_2=1)\}, \end{aligned} \tag{4.2b}$$

where $v(x_1,x_2)$ and $w(x_1,x_2)$ are the relative values in state (x_1,x_2) when service is devoted to type 1 and type 2 respectively, and g is the average cost independent of starting state. Once the optimal flow control under ROS has been determined by (4.2), the corresponding set of recurrent states, their steady-state probabilities, the mean delay, and the average throughput are easily obtained.

The selection of 0.5 for the parameter α is motivated by the desire to have an *unbiased* ROS controller, that is, one which is blind to the type of job to be provided service next. This should accentuate the difference between the optimal scheduling (SEJF which is clearly biased towards type-1) and suboptimal ROS, and provide a measure of the value of feedback for scheduling.

The overall optimal delay/throughput tradeoff is compared to the optimal tradeoff under ROS in Figure 4.16. The solid curve corresponds to the overall optimal tradeoff while the dotted curve corresponds to the optimal tradeoff under ROS. These curves were computed by varying the reward r with the holding cost rate c set to 1.

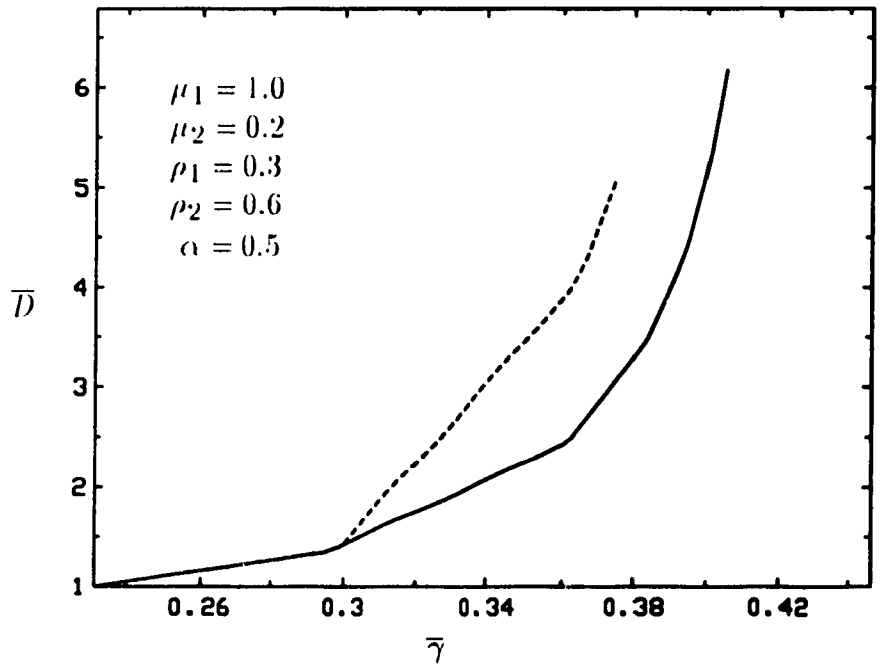

Fig. 4.16 Delay/throughput tradeoff curves.

Both curves coincide for values of r just slightly larger than $c/\mu_1 = 1$ because then the optimal flow control under both SEJF and ROS scheduling rules never admits any type 2 job and admits a type 1 job only when the queue is empty. This explains why the curves converge to the same point $(1, \lambda_1/(1+\rho_1))$ as r is reduced to $c/\mu_1 = 1$. Of course these curves could be extended to the point $(0,0)$ which corresponds to shutting down the system, the optimal policy when $0 \leq r < c/\mu_1$.

On the other hand, increasing r means increasing the optimal admission regions for both types of jobs. Since for the selected parameters the priority queue corresponding to SEJF scheduling and no flow control is stable, the optimal tradeoff curve converges to the point $(\overline{D}_\infty, \overline{\gamma}_\infty)$ given by

$$\overline{\gamma}_\infty = \lambda_1 + \lambda_2 = 0.12, \tag{4.3}$$

$$\overline{D}_\infty = \frac{\lambda_1 \overline{D}_1 + \lambda_2 \overline{D}_2}{\lambda_1 + \lambda_2} = 16.5, \tag{4.4}$$

where $\overline{D}_1$ and $\overline{D}_2$ refer to the average delay of type 1 and type 2 jobs respectively, in a single server queue with preemptive priority to type 1 tasks. Expressions for $\overline{D}_1$ and $\overline{D}_2$ can be found on page 125 of [38].

Figure 4.17 shows the gain achieved by exercizing flow control and scheduling. The lower curves are similar to the curves of Figure 4.16; they represent the delay/throughput tradeoff for SEJF (solid curve) and ROS (dotted curve)

scheduling with their respective optimal flow control. The upper curves represent the delay/throughput tradeoff for SEJF and ROS scheduling in the abcense of flow control. The mean delay for SEJF scheduling without flow control is calculated as in (4.3) and (4.4) from known expressions about preemptive priority queues. The mean delay for ROS scheduling of two job-classes differentiated by the mean rate of their exponential service times is obtained from the $M/G/1$ Pollaczek-Khinchin formula with an hyperexponential service distribution specified by the probability α that the service is exponential with rate μ_1, while the service is exponential with rate μ_2 with the complementary probability $1-\alpha$. The upper curves (no flow control) were obtained by keeping ρ_2 fixed and varying ρ_1 from 0 to the outset of instability. The lower curves (with flow control) correspond to $\rho_1 = 0.3$.

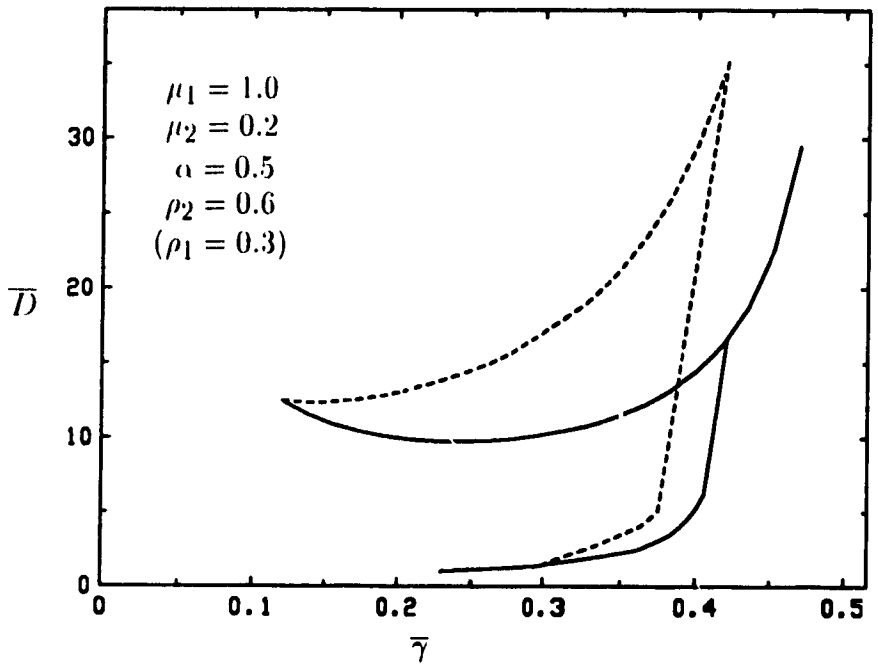

Fig. 4.17 The gain for exercizing control.

Figure 4.18 shows the sensitivity of the tradeoff to scheduling. It is a plot of the gain in decibels associated with the passage from ROS to optimal scheduling (with flow control). C_{opt} and C_{ros} refer to the minimum cost under SEJF scheduling (the optimal policy) and under ROS scheduling respectively. Consider the curve corresponding to $\rho_2 = \infty$. For ρ_1 smaller than the value where the maximum is achieved, there is a non-zero average throughput of type 2 tasks for both the optimal flow control under SEJF and the optimal flow control under ROS. For higher values of ρ_1, the optimal flow control under ROS closes the queue to the type-2 stream, while there is a non-zero throughput of type 2 jobs for the optimal flow control under SEJF. For even higher values of ρ_1, it becomes optimal to close the queue to type 2 jobs under SEJF too; for such values of ρ_1 there is no scheduling exercized anymore and the gain is 0. These comments apply to the curves corresponding to $\rho_2 < \infty$. The value of ρ_1 where the optimal flow control under ROS closes the queue to type-2 jobs is observed to be almost totally insensitive to the value of ρ_2.

It is apparent from these curves that the choice of scheduling can be of particular benefit to type 2, at least in terms of the volume of type 1 traffic that can be accomodated before type 2 is shut out. In the example to which the curves apply, the critical values of ρ_1 are $\rho_1 \approx 0.4$ when scheduling is ROS, and $\rho_1 \approx 0.8$ when scheduling is optimal - a two fold improvement.

When $\rho_2 = \infty$, we have the queue studied in Chapter 3. We recall that since

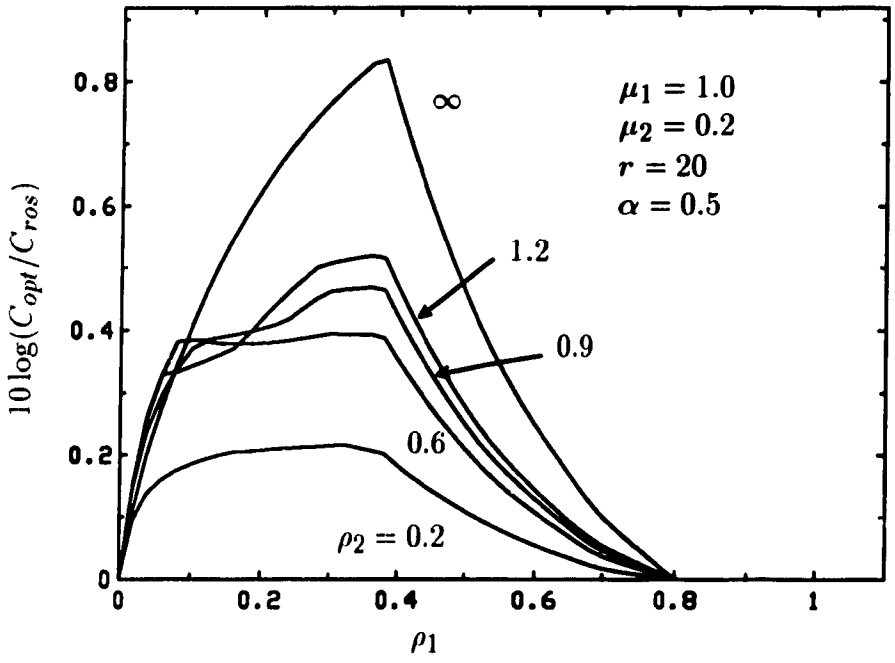

Fig. 4.18 The gain of SEJF over ROS.

type-2 tasks are always available on demand, a type-2 job is admitted only when the system is ready to start serving it. Under both SEJF and ROS, this occurs when the queue is empty. In this case, the gain of SEJF over ROS is really the gain of preemptive scheduling over non-preemptive scheduling. This gain is plotted in Figure 4.19. Note that as long as $r > c/\mu_2$, there is ρ_1 small enough for which it cannot be optimal to close the queue to type-2 jobs under either SEJF or ROS.

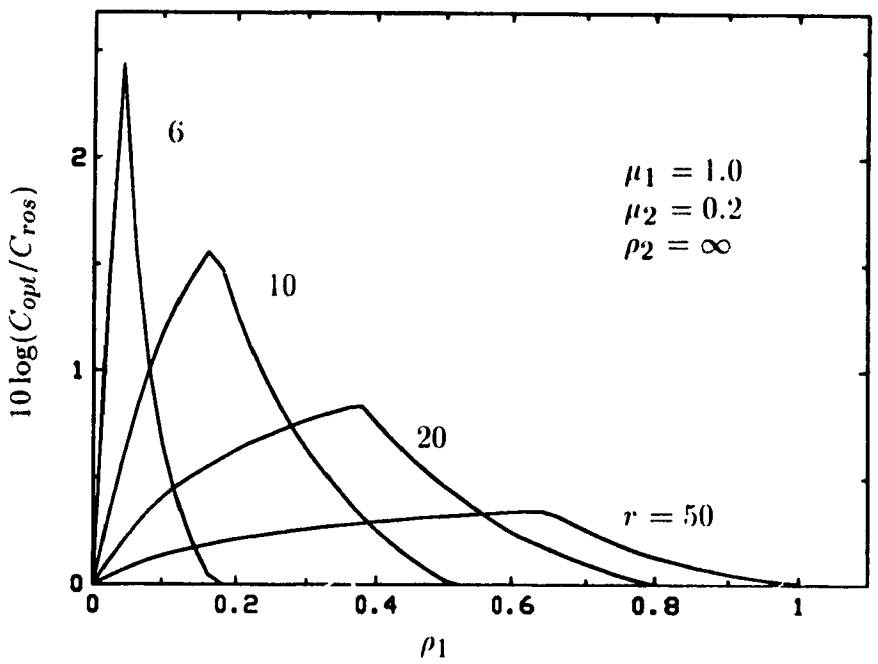

Fig. 4.19 The gain of SEJF over ROS.

Chapter 5

An approximation for the single–server queue with two job classes†

An approximation of the steady–state probabilities under SEJF scheduling and threshold type flow control is derived here. This approximation is based on the expectation that the queue process for high priority jobs (type–1) will reach steady state between any transition of the queue process for low priority jobs (type 2), especially when $\mu_1 >> \mu_2$.

Under this *steady–state* assumption, the computation of the system performance is a simple exercise. This procedure is used to determine the best threshold–type flow control for which the (approximate) average net cost is minimum.

A comparison between truly optimal and approximate delay/throughput tradeoffs reveals that the approximation is very robust and capable of remarkable accuracy.

† The analysis of this chapter has been published in [19]

5.1 The steady–state approximation

Figure 5.1 shows an optimal flow control. We recall that • indicates states where admitting type-2 jobs is the optimal action. The jointly optimal flow control for type-1 jobs is specified by threshold t_1^{opt} on the number of jobs in queue, $x_1 + x_2$. Of course, SEJF scheduling is the optimal scheduling.

The optimal type–2 flow control admits according to a monotonically decreasing function $f(x_2)$: admit a type–2 job in (x_1, x_2) iff $x_1 \leq f(x_2)$. The optimal policy of Figure 5.1 is not of threshold-type, that is, the optimal admission region for type 2 jobs is not of the form $x_1 + (\mu_1/\mu_2)x_2 \leq t_2$.

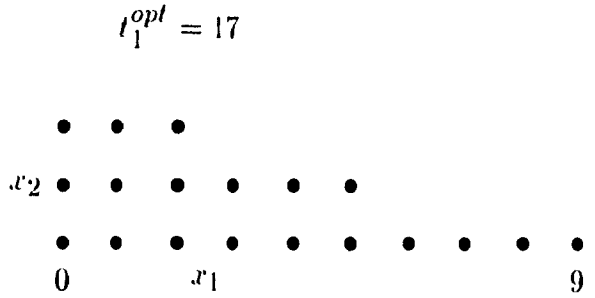

Fig. 5.1 An optimal policy.

The state-transition-rate diagram of the two-dimensional birth–death process $\mathbf{X}(t) = (X_1(t), X_2(t))$ corresponding to this optimal policy is shown in Figure 5.2. In this figure, unidirectional transitions, at rate λ_2, have been indicated with an arrow. Each horizontal segment between two states represents a pair of transitions,

one to the right at rate λ_1 and one to the left at rate μ_1. Similarly each vertical segment at the extreme left represents a pair of transitions, one up at rate λ_2 and one down at rate μ_2.

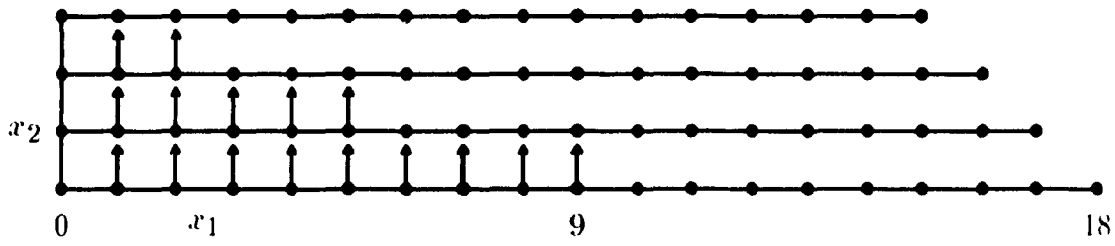

Fig. 5.2 State–transition–rate diagram for $\mathbf{X}(t)$.

Let k_2 denote the maximum value of x_2 for states in the admission region of type–2 jobs. For example, $k_2 = 2$ in Figure 5.1.

Let (X_1, X_2) be random variables representing the joint steady state distribution of process $\mathbf{X}(t)$ under a given combination of preemptive SEJF scheduling and monotonic flow control. Once the system of global balance equations has been solved for the steady–state probabilities $P(X_1 = i, X_2 = j)$, the performance

measures are easily obtained. The mean number of jobs in queue is

$$\overline{x}_1 = \sum_{j=0}^{k_2+1} \sum_{i=0}^{t_1+1-j} iP(X_1 = i, X_2 = j),$$

$$\overline{x}_2 = \sum_{j=0}^{k_2+1} \sum_{i=0}^{t_1+1-j} jP(X_1 = i, X_2 = j). \tag{5.1}$$

The mean throughputs are

$$\overline{\gamma}_1 = \lambda_1 \left\{ 1 - \sum_{j=0}^{k_2+1} P(X_1 = t_1 + 1 - j, X_2 = j) \right\},$$

$$\overline{\gamma}_2 = \lambda_2 \sum_{j=0}^{k_2} \sum_{i=0}^{f(j)} P(X_1 = i, X_2 = j). \tag{5.2}$$

The delay $\overline{D}$, averaged over all customers, is given by Little's theorem:

$$\overline{D} = \frac{\overline{x}_1 + \overline{x}_2}{\overline{\gamma}_1 + \overline{\gamma}_2}. \tag{5.3}$$

Finally, the expected cost corresponding to the (t_1, f) admission policy is

$$c(\overline{x}_1 + \overline{x}_2) - r(\overline{\gamma}_1 + \overline{\gamma}_2). \tag{5.4}$$

To approximate the steady-state probabilities, we assume that the X_1 process reaches steady-state between transitions of the X_2 process. This assumption is justified by the fact that $\mu_1 > \mu_2$, and the fact that SEJF service is applied. The distribution of the number of short jobs in queue, conditioned on the number of long jobs, is then given by

$$P(X_1 = i | X_2 = j) = \rho_1^i \left\{ \sum_{\ell=0}^{t_1-j+1} \rho_1^\ell \right\}^{-1}, \qquad \begin{array}{l} 0 \le i \le t_1 - j + 1, \\ 0 \le j \le k_2 + 1, \end{array} \tag{5.5}$$

which is the probability of state i in an $M/M/1/(k_1 - j + 1)$ queue.

Let p_j denote the conditional probability, given $X_2 = j$, of states where type 2 arrivals are admitted:

$$p_j = \sum_{\ell=0}^{f(j)} P(X_1 = \ell | X_2 = j), \qquad 0 \le j \le k_2. \tag{5.6}$$

Similarly, let q_j denote the probability of the single state where a down transition of the type-2 process is possible:

$$q_j = P(X_1 = 0 | X_2 = j), \qquad 0 \le j \le k_2 + 1. \tag{5.7}$$

To obtain the distribution of the number of long jobs in queue, we further assume that the time between transitions of the type-2 process is exponential The class-2 process is then approximated by the simple $M/M/1/(k_2 + 1)$ queue with parameters

$$\begin{aligned} \lambda(j) &= \lambda_2 p_j \ , \qquad & j = 0, 1, 2, ..., k_2 \ , \\ \mu(j) &= \mu_2 q_j \ , \qquad & j = 1, 2, ..., k_2 + 1 \ . \end{aligned} \tag{5.8}$$

It follows that

$$P(X_2 = j) = \prod_{\ell=0}^{j-1} \frac{\lambda_2 p_\ell}{\mu_2 q_{\ell+1}} P(X_2 = 0), \qquad 1 \le j \le k_2 + 1. \tag{5.9}$$

and the normalizing condition yields

$$P(X_2 = 0) = \left\{ 1 + \sum_{j=1}^{k_2+1} \rho_2^j \prod_{\ell=0}^{j-1} \frac{p_\ell}{q_{\ell+1}} \right\}^{-1} . \tag{5.10}$$

Finally the approximate steady-state probabilities are

$$P(X_1 = i, X_2 = j) = P(X_1 = i | X_2 = j) P(X_2 = j). \tag{5.11}$$

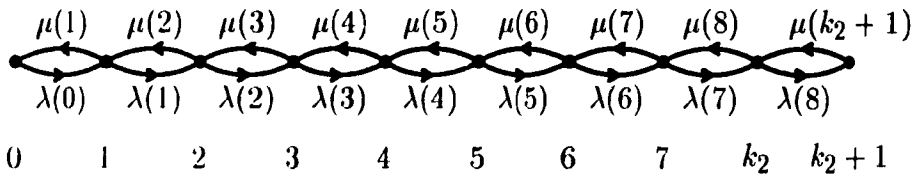

Fig. 5.3 The approximate type–2 process.

This computation is direct and avoids the solution of possibly large systems of linear equations necessary to obtain the exact performance. The function $f(x_2)$ does not have to be of the form $x_1 + (\mu_1/\mu_2)x_2 = t_2$ for some t_2. However, restricting flow control to the class of threshold–type admission policies, the above approximation is used to determine best thresholds minimizing the average net cost.

Best values of t_1 and t_2 can be computed under the assumption that the approximate cost has a dependence on t_1 and t_2 which is discretely unimodular. These best values of t_1 and t_2 approximate the truly optimal flow control obtained by dynamic programming in Chapter 4.

5.2 Comparison between approximate and best–threshold performances

We examine below the accuracy of the above approximation when it is used to evaluate the delay/throughput tradeoff. In view of the fact that the optimal joint flow control is very nearly of threshold type, rather than comparing best delay/throughput tradeoff under the steady state assumption to the overall optimal performance, we compare it to the optimal performance in the class of threshold–type flow controls. This allows us to compare policies parameterized by two numbers, t_1 and t_2.

In tables 5.1, 5.2, and 5.3, superscript *thr* refers to the optimal threshold type flow control coupled to SEJF scheduling; superscript *app* refers to the approximate flow control coupled to SEJF scheduling.

As is observed in these tables, the steady–state approximation is very robust; its accuracy is still very high for small values of β. We remark that in some cases, there is a wide gap between corresponding thresholds which does not translate in wide differences of delays and throughputs. This situation was observed to arise when the system parameters are such that the expected cost has a *flat* optimum as a function of the thresholds t_1 and t_2. As for the comparison of truly optimal and best threshold-type policies in Chapter 4, data is presented in the form of tables because corresponding plots result in curves hardly distinguishable.

ρ_1	ρ_2	r	t_1^{thr}	t_1^{app}	t_2^{thr}	t_2^{app}	$\overline{\gamma}^{thr}$	$\overline{\gamma}^{app}$	$\overline{D}^{thr}$	$\overline{D}^{app}$
.40	.10	10	4	4	2	3	.5283	.5465	2.3495	2.3737
		20	9	9	5	5	.5633	.5715	3.0574	2.8491
		30	13	13	8	9	.5803	.5891	3.6992	3.4804
		40	17	16	10	11	.5861	.5930	4.0228	3.7100
		60	26	20	14	15	.5928	.5970	4.5310	4.0316
		80	34	21	19	19	.5966	.5987	4.9403	4.2206
		100	42	21	23	21	.5982	.5991	5.1631	4.2814
.40	.95	10	4	4	1	1	.5605	.5680	2.1887	2.1056
		20	8	8	3	3	.6224	.6346	3.0975	2.9817
		40	15	15	5	5	.6519	.6636	4.0561	3.9600
		60	22	17	7	7	.6686	.6786	5.0893	5.0316
		80	28	20	8	7	.6769	.6786	5.9128	5.0316
		100	35	19	9	8	.6790	.6852	6.1904	5.9381
		150	51	20	10	9	.6843	.6871	7.0586	6.1904
.95	.10	10	3	2	0	1	.8003	.7658	2.5373	2.1345
		20	4	4	1	2	.8385	.8520	3.0896	3.2091
		40	7	7	2	3	.8940	.9048	4.5972	4.6635
		60	9	8	3	3	.9139	.9134	5.6130	5.0929
		80	10	10	3	4	.9201	.9294	6.0370	6.1652
		100	12	11	4	5	.9320	.9361	7.0764	6.7678
		200	18	17	5	5	.9492	.9524	9.5601	9.0674
		400	27	25	7	6	.9613	.9628	13.0439	12.0464
		600	31	32	8	7	.9658	.9674	15.2924	14.2813
.95	.95	10	2	2	0	1	.7702	.7858	2.1547	2.2234
		20	4	4	1	1	.8530	.8592	3.1756	3.1259
		40	7	6	1	2	.8991	.9008	4.5045	4.3274
		60	8	8	2	3	.9150	.9228	5.2263	5.3337
		80	10	10	2	3	.9276	.9345	6.0645	6.1698
		100	11	12	3	3	.9340	.9388	6.6518	6.5770
		200	17	16	4	3	.9525	.9522	9.3224	8.4981
		400	25	21	5	4	.9625	.9633	12.2179	11.6355
		600	32	32	5	5	.9665	.9665	14.1730	14.1760

Table 5.1 Best–threshold versus approximate delay and throughput for $\mu_2 = 0.5$, that is $\beta = 2$.

ρ_1	ρ_2	r	t_1^{thr}	t_1^{app}	t_2^{thr}	t_2^{app}	$\overline{\gamma}^{thr}$	$\overline{\gamma}^{app}$	$\overline{D}^{thr}$	$\overline{D}^{app}$
.10	.10	10	5	5	0	4	.1330	.4163	2.1633	2.3160
		20	11	11	3	4	.4456	.4477	2.4333	2.3767
		30	16	16	7	9	.1606	.1630	3.0868	3.0251
		40	21	19	10	14	.4657	.4701	3.4633	3 5727
		60	32	20	15	19	.1710	.4739	3.9870	4.0221
		80	43	20	20	19	.4711	.4739	4 1250	4.0221
		100	52	20	25	20	.4761	.4753	4.7853	4 2155
.10	.95	10	5	5	0	4	.4569	.4716	2.4817	2.6651
		20	10	10	2	4	.4690	.4732	2.7390	2.6979
		10	21	19	6	9	.4915	.4963	3.9003	3.9171
		60	31	19	8	9	.4928	.4963	4.0011	3 9171
		80	42	20	11	14	5032	.5067	5.2497	5.3381
		100	51	20	12	14	.5037	5067	5.3380	5 3381
.95	.10	10	3	3	−1	−1	.7790	.7790	2.4359	2 1359
		20	5	5	−1	−1	.8343	.8343	3 3506	3 3506
		40	8	8	0	4	.8798	.8857	1.8136	5.0150
		60	10	10	1	4	.8976	.9013	5.7488	5.8151
		80	12	12	2	4	.9095	.9121	6.6211	6.6451
		100	11	13	2	4	.9177	.9164	7.3916	7 0337
		200	21	21	5	4	.9358	.9362	10.1208	9 8601
		400	32	32	7	9	.9177	.9495	13.5191	13.6021
		600	12	11	9	9	.9525	.9535	15.9048	15.6008
.95	.95	10	3	3	−1	−1	.7790	.7790	2.4359	2.4359
		20	5	5	−1	−1	.8343	.8343	3.3506	3.3506
		40	8	8	0	4	.8831	.8885	4.9257	5.1122
		60	10	10	1	1	.9007	.9011	5.8727	5.9412
		80	12	11	1	4	.9117	.9099	6.6735	6 3418
		100	13	13	1	4	.9160	.9190	7.0623	7.1295
		200	21	20	3	4	.9367	.9368	10.0016	9.6333
		100	32	31	5	4	.9187	.9182	13.5116	12 8331
		600	42	41	6	9	.9535	.9512	15.8896	15 8335

Table 5.2 Best–threshold versus approximate delay and throughput for $\mu_2 = 0.2$, that is $\beta = 5$

ρ_1	ρ_2	r	t_1^{thr}	t_1^{app}	t_2^{thr}	t_2^{app}	$\overline{\gamma}^{thr}$	$\overline{\gamma}^{app}$	$\overline{D}^{thr}$	$\overline{D}^{app}$
.10	.10	10	5	5	−1	−1	.3990	.3990	1.6420	1.6420
		20	11	11	1	9	.4214	.4240	2.4512	2.5154
		30	17	18	4	9	.4233	.4240	2.5461	2.5157
		40	23	18	7	9	.4234	.4240	2.5520	2.5157
		60	34	19	14	19	.4310	.4316	3.3013	3.2520
		80	15	19	20	19	.4333	.4316	3.6904	3.2520
		100	57	19	25	19	.4347	.4316	3.9402	3.2520
.10	.95	10	5	5	−1	−1	.3990	.3990	1.6420	1.6420
		20	11	11	1	9	.4339	.4368	2.8724	2.9292
		30	17	16	3	9	.4355	.4368	2.9534	2.9295
		10	22	17	4	9	.4356	.4368	2.9613	2.9295
		60	34	19	10	18	.4444	.4482	4.0223	4.3788
		80	44	19	12	19	.4469	.4482	4.3643	4.3789
		100	56	19	14	19	.4472	.4482	4.4058	4.3789
.95	.10	10	3	3	−1	−1	.7790	.7790	2.4359	2.4359
		20	5	5	−1	−1	.8343	.8343	3.3506	3.3506
		30	6	6	−1	−1	.8514	.8514	3.7953	3.7953
		10	8	8	−1	−1	.8754	.8754	4.6593	4.6593
		80	12	12	−1	−1	.9024	.9024	6.2872	6.2872
		100	14	11	−1	−1	.9107	.9107	7.0519	7.0519
		200	22	21	2	9	.9322	.9321	10.1777	9.9910
		100	31	33	5	9	.9439	.9442	13.5906	13.3588
		600	15	11	7	9	.9484	.9488	15.8554	15.6339
		800	55	51	12	9	.9512	.9509	17.8649	17.1513
.95	.95	10	3	3	−1	−1	.7790	.7790	2.4359	2.4359
		20	5	5	−1	−1	.8343	.8343	3.3506	3.3506
		30	6	6	−1	−1	.8514	.8514	3.7953	3.7953
		10	8	8	−1	−1	.8754	.8754	4.6593	4.6593
		80	12	12	−1	−1	.9024	.9024	6.2872	6.2872
		100	11	11	−1	−1	.9107	.9107	7.0519	7.0519
		200	21	21	1	9	.9314	.9328	9.9285	10.0740
		400	34	33	3	9	.9145	.9448	13.6474	13.4488
		600	11	44	5	9	.9487	.9494	15.7451	15.7300
		800	51	54	5	9	.9509	.9516	17.2700	17.2521

Table 5.3 Best–threshold versus approximate delay and throughput for $\mu_2 = 0.1$, that is $\beta = 10$.

Chapter 6

Extension to more than two job classes

An application of the methods and approximations described in the preceeding chapters supports the conjecture that the tradeoff between delay and throughput in single server queues with multiple job classes is optimized by a combination of preemptive SEJF and monotonic flow control. The optimal flow control is very nearly characterized by hyperplanes in the multi dimensional state space. The steady state approximation extends to more than two job classes; the general formulation is sketched.

6.1 The optimal policy

Computation of the optimal combination of preemptive scheduling and flow control minimizing long run average net cost when there are more than two job classes suggests that SEJF remains optimal when coupled to simultaneously optimal flow control. As expected, the optimal flow control is monotonic, that is, described by surfaces C_i, $1 \leq i \leq m$, in the multi dimensional state space; a type i job is admitted only if the state upon arrival lies below surface C_i, $1 \leq i \leq m$

Moreover these surfaces are very nearly linear.

Let $\mathbf{x} = (x_1, x_2, ..., x_m)$ be the population and assume $\mu_1 > \mu_2 > ... > \mu_m$. The optimal flow control is almost a control-limit admission policy of the following type

Approximate threshold-type flow control

admit class-1 jobs **iff** $\dfrac{x_1 + x_2 + ... + x_m}{\mu_1} \leq t_1$

admit class-2 jobs **iff** $\dfrac{x_1}{\mu_1} + \dfrac{x_2 + x_3 + ... + x_m}{\mu_2} \leq t_2$

admit class-3 jobs **iff** $\dfrac{x_1}{\mu_1} + \dfrac{x_2}{\mu_2} + \dfrac{x_3 + x_4 + ... + x_m}{\mu_3} \leq t_3$

admit class-m jobs **iff** $\dfrac{x_1}{\mu_1} + \dfrac{x_2}{\mu_2} + ... + \dfrac{x_m}{\mu_m} \leq t_m$

or in compact form:

$$\textbf{admit a class-}k\textbf{ job in state x} \quad \textbf{iff} \quad \sum_{i=1}^{k-1} \frac{x_i}{\mu_i} + \frac{1}{\mu_k} \sum_{i=k}^{m} x_i \leq t_k \qquad (6.1)$$

In the *absence* of arrivals, the expression on the left hand side of (6.1) is easily related to the accrued cost to the system for having an extra type-k job. This extra cost is composed of $(c/\mu_k) + c\sum_{i=1}^{k}(x_i/\mu_i)$, which is the cost for queueing the

extra type k job while serving higher or equal priority jobs, and $(c/\mu_k)\sum_{i=k+1}^{m} c_i$, which is the cost for delaying lower priority jobs by $(1/\mu_k)$ the mean service time of the extra job.

When there is no arrival this extra cost is compared to the reward r in order to determine if having this extra job is beneficial to the system. In the case of independent Poisson processes, the thresholds l_i depend on the intensities λ_i, the higher the intensity, the smaller is the threshold.

Since holding cost rates and admission rewards are class independent, it is expected that if it is optimal to reject a class i job in state $\mathbf{x}$ it must be optimal also to reject lower priority jobs, namely those from classes $i+1, \ldots, n$ in state $\mathbf{x}$. It follows that we expect the optimal thresholds to satisfy

$$l_1 \geq l_2 \geq \ldots \geq l_m. \tag{6.2}$$

In the next section, we sketch how the steady state approximation of Chapter 5 can be extended to more than two job classes.

6.2 Steady-state approximation

We wish to compute the performance of a single server queue with m job classes, SEJF preemptive service, and $(t_1, t_2, ..., t_m)$ threshold-type flow control with $t_1 \geq t_2 \geq ... \geq t_m$. The computation of the joint steady-state probabilities of the process

$$\mathbf{X}(t) = (X_1(t), X_2(t), ..., X_m(t)), \tag{6.3}$$

under such control, is not an easy task, and requires the solution of a possibly large set of linear equations. We introduce a steady-state assumption allowing a simple recursive computation of approximate steady-state probabilities. The approximation can then be used to optimize the delay/throughput tradeoff within the class of $(t_1, t_2, ..., t_m)$ admission policies and to obtain the best values of $t_1, t_2, ..., t_m$. Such an optimization was carried out in Chapter 5 for the special case $m = 2$.

Let us fix the flow control thresholds $t_1 \geq t_2 \geq ... \geq t_m$. The **steady-state assumption** asserts that the process

$$\mathbf{X}_i^+(t) = (X_1(t), X_2(t), ..., X_i(t)) \tag{6.4}$$

reaches steady state between transitions of the process

$$\mathbf{X}_i^-(t) = (X_{i+1}(t), X_{i+2}(t), ..., X_m(t)) \tag{6.5}$$

for $i = 1, 2, ..., m-1$. Superscript $-$ $(+)$ refers to job classes with priority strictly smaller (higher or equal) than class i. This assumption is justified by the fact that $\mu_1 > \mu_2 > ... > \mu_m$, and the fact that SEJF service is applied.

Define $n_i = \lceil \mu_i t_i \rceil$. The set of recurrent states under our control limit admission policy is then

$$S_R = \left\{ \mathbf{x} \;\middle|\; \sum_{\ell=k}^{m} x_\ell \le n_k, 1 \le k \le m \right\}. \tag{6.6}$$

Define for $i = 2, 3, ..., m$,

$$T_i = \left\{ (x_i, x_{i+1}, ..., x_m) \;\middle|\; \sum_{\ell=k}^{m} x_\ell \le n_k, i \le k \le m \right\}. \tag{6.7}$$

T_i is the projection of the set of recurrent states onto

$$\{(x_1, x_2, ..., x_m) | x_1 = x_2 = ... = x_{i-1} = 0\}.$$

Also define for $i = 2, 3, ..., m$,

$$S_i(x_i, x_{i+1}, ..., x_m) = \left\{ (x_1, x_2, ..., x_{i-1}) \;\middle|\; \sum_{\ell=1}^{i-1} \frac{x_\ell}{\mu_\ell} + \frac{1}{\mu_i} \sum_{\ell=i}^{m} x_\ell \le t_i \right\}. \tag{6.8}$$

$S_i(x_i, ..., x_m)$ can be viewed as the support of class-i admission region above $(x_i, ..., x_m)$.

To obtain an approximation for the steady-state distribution $P(\mathbf{x})$ of $X(t)$, the conditional distributions

$$P(X_k = x_k | x_{k+1}, ..., x_m), \qquad 0 \le x_k \le n_k - x_{k+1} - ... - x_m, \tag{6.9}$$

for $(x_{k+1}, ..., x_m) \in T_{k+1}$, are computed recursively for $k = 1, 2, ..., m-1$. Finally, the marginal distribution $P(x_m)$ of $X_m(t)$ is obtained.

6.2.1 Conditional distributions of X_1

Under the current steady-state assumption and the current admission policy, the distribution of the number of shortest jobs in queue, conditioned on the number of longer jobs, is given by

$$P(X_1 = x_1 | X_2 = x_2, ..., X_m = x_m) = \rho_1^{x_1} \left\{ \sum_{\ell=0}^{n_1 - x_2 - ... - x_m} \rho_1^{\ell} \right\}^{-1}, \qquad (6.10)$$

for $0 \leq x_1 \leq n_1 - x_2 - ... - x_m$, and $(x_2, x_3, ..., x_m) \in T_2$, where $\rho_1 = \lambda_1/\mu_1$. This expression is recognized as the probability of state x_1 in an $M/M/1/(n_1 - x_2 - ... - x_m)$ queue.

6.2.2 Conditional distributions of X_k, $k = 2, ..., m-1$

Let us assume now that the conditional distributions $P(x_i | x_{i+1}, ..., x_m)$, $i = 1, 2, ..., k-1$, have been computed for some k, $2 \leq k \leq m-1$. For all $(x_{k+1}, ..., x_m) \in T_{k+1}$, define

$$\tilde{\lambda}_k(x_k | x_{k+1}, ..., x_m) = \sum_{(x_1, ..., x_{k-1}) \in S_k(x_k, ..., x_m)} \lambda_k P(x_1, ..., x_{k-1} | x_k, ..., x_m), \qquad (6.11)$$

and

$$\tilde{\mu}_k(x_k | x_{k+1}, ..., x_m) = \mu_k P(X_1 = 0, ..., X_{k-1} = 0 | x_k, ..., x_m). \qquad (6.12)$$

Of course $P(x_1, ..., x_{k-1} | x_k, ..., x_m)$ in (6.11) and (6.12) are given by

$$P(x_1, ..., x_{k-1} | x_k, ..., x_m) = \prod_{\ell=1}^{k-1} P(x_\ell | x_{\ell+1}, ..., x_m). \qquad (6.13)$$

The number of class k jobs in queue, conditioned on the number of lower priority jobs, is then approximated by the number of customers in an $M/M/1/(n_k - x_{k+1} - ... - x_m)$ queue with parameters

$$\begin{aligned} &\tilde{\lambda}_k(x_k|x_{k+1},...,x_m), \qquad 0 \le x_k \le n_k - x_{k+1} - ... - x_m - 1, \\ &\tilde{\mu}_k(x_k|x_{k+1},...,x_m), \qquad 1 \le x_k \le n_k - x_{k+1} - ... - x_m. \end{aligned} \tag{6.14}$$

It follows that

$$P(X_k = x_k|x_{k+1},...,x_m) = \prod_{\ell=0}^{x_k-1} \frac{\tilde{\lambda}_k(\ell|x_{k+1},...,x_m)}{\tilde{\mu}_k(\ell+1|x_{k+1},...,x_m)} P(X_k = 0|x_{k+1},...,x_m), \tag{6.15}$$

and the normalizing condition yields

$$P(X_k = 0|x_{k+1},...,x_m) = \left\{ 1 + \sum_{x_k=1}^{n_k - x_{k+1} - .. - x_m} \prod_{\ell=0}^{x_k-1} \frac{\tilde{\lambda}_k(\ell|x_{k+1},...,x_m)}{\tilde{\mu}_k(\ell+1|x_{k+1},...,x_m)} \right\}^{-1}. \tag{6.16}$$

6.2.3 Marginal distribution of X_m

Define

$$\tilde{\lambda}_m(x_m) = \sum_{(x_1,...,x_{m-1}) \in S_m(x_m)} \lambda_m P(x_1,...,x_{m-1}|x_m), \tag{6.17}$$

and

$$\tilde{\mu}_m(x_m) = \mu_m P(X_1 = 0,...,X_{m-1} = 0|x_m). \tag{6.18}$$

The marginal distribution of the number of longest jobs is approximated by the distribution of the number of customers in an $M/M/1/n_m$ queue with parameters

$$\begin{aligned} &\tilde{\lambda}_m(x_m), \qquad 0 \le x_m \le n_m - 1, \\ &\tilde{\mu}_m(x_m), \qquad 1 \le x_m \le n_m. \end{aligned} \tag{6.19}$$

It follows that

$$P(X_m = x_m) = \prod_{\ell=0}^{x_m-1} \frac{\tilde{\lambda}_m(\ell)}{\tilde{\mu}_m(\ell+1)} P(X_m = 0), \tag{6.20}$$

and the normalizing condition yields

$$P(X_m = 0) = \left\{ 1 + \sum_{x_m=1}^{n_m} \prod_{\ell=0}^{x_m-1} \frac{\tilde{\lambda}_m(\ell)}{\tilde{\mu}_m(\ell+1)} \right\}^{-1}. \tag{6.21}$$

6.2.4 Approximate performance

The approximate performance under the given $(t_1, t_2, ..., t_m)$ flow control coupled to SEJF scheduling is then obtained as follows. First the approximate steady-state distribution of the queue length process is

$$P(\mathbf{x}) = P(x_1|x_2, ..., x_m) \cdot P(x_2|x_3, ..., x_m) \cdots P(x_{m-1}|x_m) \cdot P(x_m). \tag{6.22}$$

The mean number of class i jobs in queue is

$$\overline{x}_i = \sum_{\mathbf{x} \in S_R} x_i P(\mathbf{x}), \qquad 1 \le i \le m. \tag{6.23}$$

The mean throughputs are

$$\overline{\gamma}_i = \sum_{(x_1,...,x_m) \in T_i} \quad \sum_{(x_1,...,x_{i-1}) \in S_i(x_1,...,x_m)} \lambda_i P(\mathbf{x}). \tag{6.24}$$

The mean delay $\overline{D}$, averaged over all customers, is given by Little's theorem:

$$\overline{D} = \frac{\overline{x}_1 + \overline{x}_2 + ... + \overline{x}_m}{\overline{\gamma}_1 + \overline{\gamma}_2 + ... + \overline{\gamma}_m}. \tag{6.25}$$

Finally, the expected cost corresponding to the $(t_1, t_2, ..., t_m)$ admission policy is

$$c(\overline{x}_1 + \overline{x}_2 + ... + \overline{x}_m) - r(\overline{\gamma}_1 + \overline{\gamma}_2 + ... + \overline{\gamma}_m). \tag{6.26}$$

Chapter 7 Methods of proof

The proof that the tradeoff between delay and throughput in our multi–class single server queue is optimized by a combination of preemptive SEJF scheduling and monotonic flow control is plagued with analytical difficulties. We show here how some techniques (in particular the Linear Programming Approach) fail in the present setting, while others (like the Induction Approach) present unexpected difficulties in their application. A proof of the above conjecture is still missing in the general case; in Chapter 2, we presented a proof for the special case of only two queues, one being saturated.

We consider here the model described in Chapter 2, but with only two job classes. We assume throughout that $\mu_1 > \mu_2$.

7.1 The Induction Approach

The Induction Approach is a well established method in the context of control of queues [72]. It has been widely used in a variety of situations to characterize

optimal policies. Notable and relevant applications of the Inductive Approach include: Johansen and Stidham [35] who applied it to the admission control for a stochastic input-output system; Hajek [26] who controls routing and service priority in a network of two queues with feedback; Ghoneim and Stidham [23] who control admission at each of two nodes in tandem; Weber and Stidham [81] who control service rates in a tandem network. See also [16], [27], [47], [48], [68], [70], [71], and [86].

In essence, the approach consists in studying finite-horizon α discounted costs for the system under investigation. The precise meaning of *finite horizon* will be given shortly. The finite horizon dynamic programming equation is then used to establish, *by induction on the horizon*, that finite-horizon optimal policies have certain structural properties (like switch-curve). Under general conditions ([65]), these properties extend to the infinite-horizon discounted-cost problem. The long-run average cost problem is then treated either as a limit of discounted problems with discount factor approaching zero as in [2], [45], [46], [64], or as the limiting case of finite time-horizon problems as in [6]-[9] and [61].

A continuous-time Markov decision process formulation of the problem was derived in Chapter 2. In the case of only two job-classes the state space is

$$S = \{\mathbf{x} = (x_1, x_2) | x_1, x_2 = 0, 1, 2, ...\}. \tag{7.1}$$

We recall that control $\mathbf{a} = (a_1, a_2, a_3)$ specifies whether a type i arrival ($i = 1, 2$) should be accepted ($a_i = 1$) or rejected ($a_i = 0$), while a_3 specifies whether

service is devoted to type 1 ($a_3 = 1$) or to type-2 ($a_3 = 0$). The state transition probability function is given by

$$\begin{aligned} p(\mathbf{x},\mathbf{a},\mathbf{y})\Lambda =& \lambda_1 a_1 I[\mathbf{y} = A_1(\mathbf{x})] + \lambda_2 a_2 I[\mathbf{y} = A_2(\mathbf{x})] \\ &+ \mu_1 a_3 I[\mathbf{y} = D_1(\mathbf{x})] + \mu_2(1 - a_3) I[\mathbf{y} = D_2(\mathbf{x})] \end{aligned} \tag{7.2}$$

where $\Lambda = \lambda_1 + \lambda_2 + \mu_1 + \mu_2$ is the total event rate, $I[\ ]$ the indicator function, and

$$\begin{aligned} A_1(\mathbf{x}) &= (x_1 + 1, x_2), \\ A_2(\mathbf{x}) &= (x_1, x_2 + 1), \\ D_1(\mathbf{x}) &= ((x_1 - 1)^+), x_2), \\ D_2(\mathbf{x}) &= (x_1, (x_2 - 1)^+), \end{aligned} \tag{7.3}$$

and $z^+ = \max(z, 0)$ for real z. A control policy π is a sequence $(\mathbf{u}_0, \mathbf{u}_1, ...)$ of functions

$$\mathbf{u}_k : S \mapsto \{0,1\}^3$$

Under a policy π, the state process starting in initial state $\mathbf{x}_0 = (x_{1,0}, x_{2,0})$ evolves as a uniformized Markov process

$$\mathbf{X}^{(\pi)}(t) = \mathbf{Y}^{(\pi)}_{N(t)},$$

where $N(t)$ is a Poisson process with rate Λ, independent of the Markov chain $\mathbf{Y}_k$ on S, with $P(\mathbf{Y}_0 = \mathbf{x}_0) = 1$, and transition probabilities

$$P(\mathbf{Y}_{k+1} = \mathbf{y} | \mathbf{Y}_k = \mathbf{x}) = p(\mathbf{x}, \mathbf{u}_k(\mathbf{x}), \mathbf{y}).$$

$\mathbf{Y}_k$ is the state embedded at transition epochs.

Let $0 < \tau_1 < \tau_2 < \ldots < \tau_k < \ldots$ denote the sequence of jump times for process $N(t)$. The expected α-discounted net cost for operating the system under a policy π over the random interval $[0, \tau_n]$ when starting in state $\mathbf{x}_0$ is

$$V_n^{(\pi)}(\mathbf{x}_0) = E_{\mathbf{x}_0} \sum_{k=1}^{n} \left\{ c \int_{\tau_{k-1}}^{\tau_k} e^{-\alpha t} \|\mathbf{X}^{(\pi)}(\tau_{k-1})\| dt \right. \\ \left. - re^{-\alpha \tau_k} I \left[\|\mathbf{X}^{(\pi)}(\tau_k) - \mathbf{X}^{(\pi)}(\tau_{k-1})\| = 1 \right] \right\}, \tag{7.1}$$

where $\|\mathbf{X}(t)\| = X_1(t) + X_2(t)$. Evaluating we obtain

$$V_n^{(\pi)}(\mathbf{x}_0) = E_{\mathbf{x}_0} \sum_{k=1}^{n} \beta^{k-1} \left\{ c(\alpha + \Lambda)^{-1} \|\mathbf{X}^{(\pi)}(\tau_{k-1})\| \right. \\ \left. - \beta r I \left[\|\mathbf{X}^{(\pi)}(\tau_k) - \mathbf{X}^{(\pi)}(\tau_{k-1})\| = 1 \right] \right\}, \tag{7.5}$$

where

$$\beta = \frac{\Lambda}{\alpha + \Lambda}.$$

This cost is easily interpreted as the cost over the first n steps of a discrete time Markov decision process, with discount factor β (β was used in chapters 4 and 5 to denote a ratio of service rates; in this chapter, β denotes the discount factor).

For given initial state $\mathbf{x}_0$ define

$$V_n^{\beta}(\mathbf{x}_0) = \min_{\pi} V_n^{(\pi)}(\mathbf{x}_0), \tag{7.6}$$

the minimum β discounted cost over the first n transitions. By convention $V_0^\beta(\mathbf{x}_0)$ is set to 0. Then V_n^β is characterized by the dynamic programming optimality equation

$$\begin{aligned} V_{n+1}^\beta(x_1,x_2) =& c(x_1+x_2)(\alpha+\Lambda)^{-1} \\ &+ \beta(\lambda_1/\Lambda)\min\{V_n^\beta(x_1+1,x_2)-r, V_n^\beta(x_1,x_2)\} \\ &+ \beta(\lambda_2/\Lambda)\min\{V_n^\beta(x_1,x_2+1)-r, V_n\beta(x_1,x_2)\} \\ &+ \beta\min\Big\{(\mu_1/\Lambda)V_n^\beta((x_1-1)^+,x_2)+(\mu_2/\Lambda)V_n^\beta(x_1,x_2), \\ &\qquad (\mu_1/\Lambda)V_n^\beta(x_1,x_2)+(\mu_2/\Lambda)V_n^\beta(x_1,(x_2-1)^+)\Big\}, \end{aligned} \tag{7.7}$$

which is equivalent, after simplification, to

$$V_{n+1}^\beta = TV_n^\beta, \tag{7.8}$$

where T is the operator define on real-valued functions f on S by

$$\begin{aligned} Tf(x_1,x_2) = \beta\Lambda^{-1}\Big\{& c(x_1+x_2) \\ &+ \lambda_1\min\{f(x_1+1,x_2)-r, f(x_1,x_2)\} \\ &+ \lambda_2\min\{f(x_1,x_2+1)-r, f(x_1,x_2)\} \\ &+ \min\{\mu_1 f((x_1-1)^+,x_2)+\mu_2 f(x_1,x_2), \\ &\qquad \mu_1 f(x_1,x_2)+\mu_2 f(x_1,(x_2-1)^+)\}\Big\}. \end{aligned} \tag{7.9}$$

From (7.7) it follows that the optimal action $\mathbf{a} = (a_1, a_2, a_3)$ in state $\mathbf{x} = (x_1, x_2)$ when there are n steps remaining is determined by

$$\begin{aligned} a_1 = 1 &\Longleftrightarrow V_n^{r\beta}(x_1+1, x_2) - V_n^{r\beta}(x_1, x_2) \le r \\ a_2 = 1 &\Longleftrightarrow V_n^{r\beta}(x_1, x_2+1) - V_n^{r\beta}(x_1, x_2) \le r \\ a_3 = 1 &\Longleftrightarrow \mu_1 \left[V_n^{r\beta}(D_1(\mathbf{x}) - V_n^{r\beta}(\mathbf{x})\right] \le \mu_2 \left[V_n^{r\beta}(D_2(\mathbf{x}) - V_n^{r\beta}(\mathbf{x})\right] \end{aligned} \tag{7.10}$$

In view of (7.10) the *monotonicity* of the optimal flow control will be ensured if the optimal cost functions satisfy:

1) $V_n^{r\beta}(x_1+1, x_2) - V_n^{r\beta}(x_1, x_2)$ is non-decreasing in x_1 (convexity in x_1),

2) $V_n^{r\beta}(x_1+1, x_2) - V_n^{r\beta}(x_1, x_2)$ is non-decreasing in x_2 (supermodularity),

3) $V_n^{r\beta}(x_1, x_2+1) - V_n^{r\beta}(x_1, x_2)$ is non-decreasing in x_2 (convexity in x_2),

4) $V_n^{r\beta}(x_1, x_2+1) - V_n^{r\beta}(x_1, x_2)$ is non-decreasing in x_1 (supermodularity).

Indeed, these properties imply that if admission of a type i job is optimal in state (x_1, x_2), admission is also optimal in states $(x_1 - 1, x_2)$ and $(x_1, x_2 - 1)$. Thus the boundary between admission and rejection regions is characterized by a monotonic switching-curve. As for scheduling, optimality of SEJF requires that, for $x_1 \ge 1$ and $x_2 \ge 1$,

$$\mu_1 \left[V_n(x_1 - 1, x_2) - V_n(x_1, x_2)\right] \le \mu_2 \left[V_n(x_1, x_2 - 1) - V_n(x_1, x_2)\right]. \tag{7.11}$$

The objective is to use the dynamic programming equation (7.8) to show, by induction on the horizon n, that the minimum cost functions satisfy the required properties. This amounts to showing that these properties carry over from f to Tf. Usually, to show that Tf acquires a certain property from f, one obtains a much stronger result; namely, that each term of the summation defining Tf has itself the required property.

In some cases an even stronger result is obtained to establish the switch-curve structure, namely that diagonal-monotonicity (also called subconvexity), rather than convexity, carries over from f to Tf. Diagonal-monotonicity was used by Hajek in [26] for his inductive proof of switching-curve optimal policies for a network of two queues with feedback; it was also used in [23] to derive the monotonicity of optimal flow control at each of two nodes in tandem. See also [80].

For a real valued function $f(x_1, x_2)$ we define

Subconvexity of f in x_1:

$$f(x_1+1, x_2+1) - f(x_1, x_2+1) \leq f(x_1+2, x_2) - f(x_1+1, x_2).$$

Subconvexity of f in x_2:

$$f(x_1+1, x_2+1) - f(x_1+1, x_2) \leq f(x_1, x_2+2) - f(x_1, x_2+1).$$

A function f is **diagonally monotone** if it is both subconvex in x_1 and subconvex in x_2. The appeal of diagonal-monotonicity comes from the following

elementary results about the transformations

$$T_1 f(x_1, x_2) = \min\{f(x_1 + 1, x_2) - r, f(x_1, x_2)\}$$

$$T_2 f(x_1, x_2) = \min\{f(x_1, x_2 + 1) - r, f(x_1, x_2)\}.$$

where r is a real constant. The simple proofs are sketched in Appendix B.

Proposition 7.1

If f is nondecreasing in x_1 and in x_2,

then $T_1 f$ and $T_2 f$ are nondecreasing in x_1 and in x_2.

Proposition 7.2

If f is supermodular, subconvex in x_1, and subconvex in x_2,

then f is convex in x_1 and in x_2.

Proposition 7.3

If f is supermodular, convex in x_1 and convex in x_2,

then $T_1 f$ and $T_2 f$ are supermodular for all $r \geq 0$.

Proposition 7.4

If f is convex in x_1,

then $T_1 f$ is convex in x_1 for all $r \geq 0$.

Proposition 7.5

If f is convex in x_2,

then $T_2 f$ is convex in x_2 for all $r \geq 0$.

Proposition 7.6

For f nondecreasing and supermodular,

then $T_1 f$ is convex in x_2 for all $r \geq 0$ iff f is subconvex in x_2.

Proposition 7.7

For f nondecreasing and supermodular,

then $T_2 f$ is convex in x_1 for all $r \geq 0$ iff f is subconvex in x_1.

Combining these results we have

Proposition 7.8

If f is nondecreasing, supermodular, and diagonally monotone, then $T_1 f$ and $T_2 f$ are nondecreasing, supermodular, and diagonally monotone for all $r \geq 0$.

Should supermodularity and diagonal-monotonicity carry over from f to $T_3 f$

$$T_3 f(x_1, x_2) = \min\{\mu_1 f((x_1 - 1)^+, x_2) + \mu_2 f(x_1, x_2), \\ \mu_1 f(x_1, x_2) + \mu_2 f(x_1, (x_2 - 1)^+)\}, \tag{7.12}$$

then finite horizon minimum cost functions V_n^β would be supermodular and diagonally monotone, hence convex, implying switch-curve structure for finite-horizon optimal flow controls.

However, a numerical evaluation of the minimum costs V_n^β reveals that they are not necessarily diagonally monotone functions of the state (x_1, x_2). On the other hand they appear to be convex.

Thus, diagonal monotonicity is too strong in the context of (7.7). To perform the inductive step, one must then determine a set of weaker properties of the cost functions V_n^β that imply switch-curve flow control and SEJF scheduling, and that carry over either directly from f to Tf or from f to each term of Tf and thus to Tf itself. We were not able to carry out that program. Rosberg, Varaiya, and Walrand [61] encountered similar difficulties in their analysis of the optimal control

of service in tandem queues. Nevertheless, they prove that the optimal control is switch-curve by constructing an equivalent linear programming problem and using a duality argument to deduce convexity. The application of this approach to our problem is studied in the next section.

7.2 The Linear Programming Approach

The Linear Programming Approach, introduced in [61] and subsequently generalized in [80], fails when applied to our problem. In short, this approach consists in formulating the finite horizon optimization problem as an integer programming problem on the finite horizon controls. One considers then the corresponding linear programming problem obtained by allowing the variables (controls) to take values in $[0, 1]$ rather than $\{0, 1\}$. The success of the approach, hinges on the possibility of showing that, for any starting state, the relaxed LP has an integer solution, which must then be equal to the solution of the original integer program. Viniotis [80] pointed out that a simple way of establishing this property is to check that the matrix of constraints for LP is totally unimodular [66]. This is easily done for many controlled queues [41], [49], [80]. However, as shown below, the matrix of constraints corresponding to our problem **is not** totally unimodular.

7.2.1 Formulation of the finite–horizon problem as a integer program

As in section 7.1, we consider the special case of only two job classes. Once uniformization has been applied to the original continuous-time problem, the

sample space for the discrete-time Markov decision problem over n uniformized time slots consists of all sequences

$$\omega^n = (\omega_1, ..., \omega_n) \in \Omega^n,$$

where $\omega_i \in \Omega = \{A_1, A_2, D_1, D_2\}$ are i.i.d. random variables with $Pr\{\omega = A_i\} = \lambda_i$, and $Pr\{\omega = D_i\} = \mu_i$, for $i = 1, 2$, after the parameters have been normalized so that $\Lambda = \lambda_1 + \lambda_2 + \mu_1 + \mu_2 = 1$. Obviously, $\omega_k = A_i$ or D_i according to whether the transition at step k is a type–i arrival or a type–i departure (true or false).

Define

$$\xi(\omega) = \begin{cases} (1,0) & \text{if } \omega = A_1, \\ (0,1) & \text{if } \omega = A_2, \\ (-1,0) & \text{if } \omega = D_1, \\ (0,-1) & \text{if } \omega = D_2. \end{cases}$$

$\xi(\omega)$ represents the change in state incurred by transition ω.

A **process** f is a sequence of random variables $f_1, f_2, ..., f_n$, where f_k is a function on Ω^k. Let $\xi = (\xi_1, ..., \xi_n)$ be the process given by $\xi_k(\omega^k) = \xi(\omega_k)$.

A **policy** z is a process $(z_1 \ ..., z_n)$ of functions z_k on Ω^k with values in $\{0, 1\}$ and such that

$$z_k(\omega^{k-1}, D_1) + z_k(\omega^{k-1}, D_2) \in \{0, 1\}, \qquad \omega^{k-1} \in \Omega^{k-1}. \tag{7.13}$$

The control variable $z_k(\omega^k)$ specifies the action to be taken at time k if the system evolves as ω^k. The control variable $z_k(\omega^{k-1}, A_i)$ is equal to 1 or 0 according to

whether a type-i arrival is accepted or rejected at time k. On the other hand, $z_k(\omega^{k-1}, D_1) = 1$ indicates that service is devoted to a type-1 job over the k th slot. The condition (7.13) ensures that a single customer is being served at any given time.

The **trajectory** in state space S corresponding to policy z and initial state $\mathbf{x}_0 \in S$, is the process $(\mathbf{x}_1, ..., \mathbf{x}_n)$ defined by

$$\mathbf{x}_k(\omega^k) = \mathbf{x}_{k-1}(\omega^{k-1}) + z_k(\omega^k)\xi_k(\omega^k), \qquad 1 \le k \le n. \tag{7.14}$$

We assume further that a policy is conservative and maintains the corresponding trajectory within the state space S, that is

$$x_{k,1}(\omega^k) \ge 0, \quad x_{k,2}(\omega^k) \ge 0, \qquad \omega^k \in \Omega^k, 1 \le k \le n. \tag{7.15}$$

The expected costs incurred over n uniformized time slots when starting in state $\mathbf{x}_0$ and applying policy z is

$$\begin{aligned} V_n(z, \mathbf{x}_0) = & E \sum_{k=0}^{n-1} \beta^k c(\alpha + 1)^{-1} \|\mathbf{x}_k\| \\ & + \sum_{k=1}^{n} \sum_{\omega^{k-1} \in \Omega^{k-1}} \Big\{ P(\omega^{k-1}, A_1) z_k(\omega^{k-1}, A_1) \\ & \qquad + P(\omega^{k-1}, A_2) z_k(\omega^{k-1}, A_2) \Big\} \beta^k r, \end{aligned} \tag{7.16}$$

where $\|\mathbf{x}_k\| = x_{k,1} + x_{k,2}$ is the total number of jobs at epoch k. Substitution for

$\mathbf{x}_k$ yields after elementary manipulations

$$\begin{aligned} V_n(z,\mathbf{x}_0) =& c\delta_0\|\mathbf{x}_0\| - \beta^n r(\lambda_1+\lambda_2) \\ &+ \sum_{k=1}^{n-1} \sum_{\omega^{k-1}\in\Omega^{k-1}} P(\omega^{k-1})\Big\{\lambda_1 z_k(\omega^{k-1},A_1)(\delta_k c - \beta^k r) \\ &\qquad + \lambda_2 z_k(\omega^{k-1},A_2)(\delta_k c - \beta^k r) \\ &\qquad - \mu_1 z_k(\omega^{k-1},D_1)\delta_k c \\ &\qquad - \mu_2 z_k(\omega^{k-1},D_2)\delta_k c\Big\}, \end{aligned} \tag{7.17}$$

where $\delta_k = \beta^k + \beta^{k+1} + ... + \beta^{n-1}$. Equation (7.17) can be expressed as

$$V_n(z,\mathbf{x}_0) = c\delta_0\|\mathbf{x}_0\| + \sum_{k=1}^{n} \sum_{\omega^k\in\Omega^k} \sigma_k(\omega^k) z_k(\omega^k), \tag{7.18}$$

where the process σ is independent of the policy z and the starting state $\mathbf{x}_0$. Thus, the cost is a linear function of control variables $z_k(\omega^k)$.

The optimal policy over the first n transition epochs is then the solution of the integer program in the finite set of variables $\{z_k(\omega^k)|\omega^k \in \Omega^k, 1 \le k \le n\}$,

$$(IP): \qquad V_n(\mathbf{x}_0) = \min_z \sum_{k=1}^{n} \sum_{\omega^k\in\Omega^k} \sigma_k(\omega^k) z_k(\omega^k)$$

subject to the constraints

- $z_k(\omega^k) \in \{0,1\}, \qquad \omega^k \in \Omega^k, 1 \le k \le n,$ (7.19a)
- $z_k(\omega^{k-1},D_1) + z_k(\omega^{k-1},D_2) \in \{0,1\}, \quad \omega^{k-1} \in \Omega^{k-1}, 1 \le k \le n,$ (7.19b)
- $\mathbf{x}_0 + \sum_{j=1}^{k} z_j(\omega^j)\xi_j(\omega^j) \in S, \qquad \omega^k \in \Omega^k, 1 \le k \le n.$ (7.19c)

7.2.2 Relaxation and optimal policy

Our objective is to establish the convexity of finite-horizon cost $V_n(\mathbf{x}_0)$; this would imply switch-curve structure for the optimal finite-horizon flow controls. The simultaneous optimization of flow control and scheduling is embodied in the formulation; however, the structure of the jointly optimal scheduling is not addressed here.

With that in mind, we relax the constraints (7.19) by allowing the control variables to take values in $[0,1]$ rather than in $\{0,1\}$. The trajectory (7.14) can then be interpreted as the evolution of a queue with "fractional" jobs. The minimum finite-horizon cost for such a queue is then the solution of the linear program

$$(LP): \qquad W_n(\mathbf{x}_0) = \min_{z} \sum_{k=1}^{n} \sum_{\omega^k \in \Omega^k} \sigma_k(\omega^k) z_k(\omega^k)$$

subject to the constraints

- $0 \le z_k(\omega^k) \le 1, \qquad \omega^k \in \Omega^k, 1 \le k \le n,$ (7.20a)
- $0 \le z_k(\omega^{k-1}, D_1) + z_k(\omega^{k-1}, D_2) \le 1, \quad \omega^{k-1} \in \Omega^{k-1}, 1 \le k \le n,$ (7.20b)
- $\mathbf{x}_0 + \sum_{j=1}^{k} z_j(\omega^j)\xi_j(\omega^j) \ge (0,0), \qquad \omega^k \in \Omega^k, 1 \le k \le n.$ (7.20c)

The following proposition is a standard result in the theory of linear programming (see [61], [80])

Proposition 7.9

$W_n(\mathbf{x})$ is a convex function of $\mathbf{x}$.

In general (LP) and (IP) are quite different. However, if (LP) has an integer solution, then $W_n(\mathbf{x}) = V_n(\mathbf{x})$ for integer $\mathbf{x}$, and convexity of V_n follows from Proposition 7.9.

Viniotis [80] exploited the following fundamental theorem (see Schrijver [66], Chapter 19, Corollary 19.2a (Hoffman and Kruskal's theorem), page 268):

Proposition 7.10

Let A be an integral matrix. Then the vertices of the polyhedron $\{\mathbf{x}|A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$ all have integer components for any integral vector b if and only if the matrix A is *totally unimodular.*

Therefore, to determine whether (LP) has an integral solution, we need to verify if the matrix of constraints corresponding to (7.20) is totally unimodular. From (7.20) it is obvious that the matrix of constraints A for (LP) has entries in $\{-1, 0, 1\}$, while the vector b, in $Ax \leq b$, has entries in $\{1, x_{0,1}, x_{0,2}\}$. In fact A and b are the same for both (LP) and (IP).

Definition 7.11

The matrix A is **totally unimodular** if each of its subdeterminants is -1, 0, or $+1$. In particular, each entry of a totally unimodular matrix is -1, 0, or $+1$.

7.2.3 The matrix of constraints is not totally unimodular

There are many characterizations of totally unimodular matrices. Some of them are collected in Theorem 19.3 of [66]. Among these, a useful one in the present context is given by the next proposition.

Proposition 7.12

A matrix with entries 0, +1, −1 is totally unimodular if and only if each collection of columns can be split into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries only 0, +1, and −1.

To see that the matrix of constraints A defined by (7.20) is not totally unimodular in general, we set the horizon $n = 3$ and consider the collection of columns corresponding to variables $z(D_1)$, $z(D_1, D_1)$, $z(D_1, D_2)$, $z(D_1, D_2, D_1)$, and $z(D_1, D_2, D_2)$. Concentrating on the lines corresponding to constraints

a) $x_{3,1}(D_1, D_1, D_1) \geq 0$,

b) $x_{3,1}(D_1, D_2, D_1) \geq 0$,

c) $x_{3,2}(D_1, D_2, D_2) \geq 0$,

d) $z(D_1, D_1) + z(D_1, D_2) \leq 1$,

e) $z(D_1, D_2, D_1) + z(D_1, D_2, D_2) \leq 1$,

we obtain the following tableau:

	$z(D_1)$	$z(D_1, D_1)$	$z(D_1, D_2)$	$z(D_1, D_2, D_1)$	$z(D_1, D_2, D_2)$
a)	1	1	0	0	0
b)	1	0	0	1	0
c)	0	0	1	0	1
d)	0	1	1	0	0
e)	0	0	0	1	1

It should be obvious that this set of columns cannot be split into two parts such that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries only 0, +1, and −1. By proposition 7.12 we must conclude that the matrix of constraints is not totally unimodular. That (LP) has a non integer solution was also verified numerically.

Consequently, (LP) does not necessarily have an integer solution, optimal and relaxed cost-functions V_n and W_n do not coincide, and convexity of W_n cannot be used to obtain convexity of V_n. Thus, the method fails in this context.

7.3 The First Passage Cost Approach

As mentioned in Section 7.1, the optimality of a monotonic policy for minimization of long-run average costs is usually inferred from a similar result for the α discounted problem. Stidham and Weber [73] approached the problem from a different angle. They consider minimizing the expected (undiscounted) net cost until the system first returns to the empty state from each starting state. Their

method relies on the left skip-free property of the state process on its way to the empty state. The average cost problem is recovered from its equivalence to the problem of minimizing expected net cost until a return to the empty state when holding cost-rate in state i is $ci - g$, where g is the minimum average net cost.

Under SEJF scheduling (priority to class 1), the state process for our single server queue must visit state $(0, x_2)$ on its way to state $(0,0)$ from state (x_1, x_2). We investigate here whether this property can be exploited, as in [73], to establish the monotonicity of the optimal flow control under SEJF scheduling.

SEJF scheduling is a static priority discipline. We are then considering the optimal flow control problem in priority queues. That scheduling is not subject to optimization is a shift from our main theme. This shift is motivated by our conjecture that SEJF is optimal.

We consider our single-server model with two job-classes. We modify the holding cost-rate to $c(i+j) - h$ in state (i,j), where h is a real constant and, as usual, first and second coordinates refer to the number of type 1 and type 2 jobs respectively. We assume $h \leq 0$.

Let $v(i,j)$ denote the minimum expected first-passage cost from state (i,j) to state $(0,0)$ assuming that class 1 is assigned preemptive priority over class 2.

These costs satisfy the following optimality equation

$$v(0,0) = 0.$$

$$\begin{aligned}(\lambda_1 + \lambda_2 + \mu_2)v(0,j) &= (cj - h) + \mu_2 v(0,j-1) \\ &\quad + \lambda_1 \min\{v(1,j) - r, v(0,j)\} \\ &\quad + \lambda_2 \min\{v(0,j+1) - r, v(0,j)\}, \quad j \geq 1, \end{aligned} \tag{7.21}$$

$$\begin{aligned}(\lambda_1 + \lambda_2 + \mu_1)v(i,j) &= c(i+j) - h + \mu_1 v(i-1,j) \\ &\quad + \lambda_1 \min\{v(i+1,j) - r, v(i,j)\} \\ &\quad + \lambda_2 \min\{v(i,j+1) - r, v(i,j)\}, \quad i \geq 1, j \geq 0. \end{aligned}$$

With $w_k(i,j)$ be the minimum expected first-passage cost from state (i,j) to state $(0,k)$ for $j \geq k$,

$$w_k(i,j) = v(i,j) - v(0,k) \tag{7.22}$$

and

$$w_k(0,k) = 0,$$

$$\begin{aligned}(\lambda_1 + \lambda_2 + \mu_2)w_k(0,j) &= cj - h \\ &\quad + \lambda_1 \min\{w_k(1,j) - r, w_k(0,j)\} \\ &\quad + \lambda_2 \min\{w_k(0,j+1) - r, w_k(0,j)\} \\ &\quad + \mu_2 w_k(0,j-1), \qquad j \geq k+1, \end{aligned} \tag{7.23}$$

$$\begin{aligned}(\lambda_1 + \lambda_2 + \mu_1)w_k(i,j) &= c(i+j) - h \\ &\quad + \lambda_1 \min\{w_k(i+1,j) - r, w_k(i,j)\} \\ &\quad + \lambda_2 \min\{w_k(i,j+1) - r, w_k(i,j)\} \\ &\quad + \mu_1 w_k(i-1,j), \qquad i \geq 1, j \geq k. \end{aligned}$$

Defining $u_k(i,j) = w_k(i, j+k)$, for $i \geq 0$, $j \geq 0$, we obtain

$$
\begin{aligned}
u_k(0,0) &= 0, \\
(\lambda_1 + \lambda_2 + \mu_2)u_k(0,j) &= c(j+k) - h \\
&\quad + \lambda_1 \min\{u_k(1,j) - r, u_k(0,j)\} \\
&\quad + \lambda_2 \min\{u_k(0,j+1) - r, u_k(0,j)\} \\
&\quad + \mu_2 u_k(0,j-1), \qquad j \geq 1, \qquad (7.24) \\
(\lambda_1 + \lambda_2 + \mu_1)u_k(i,j) &= c(i+j+k) - h \\
&\quad + \lambda_1 \min\{u_k(i+1,j) - r, u_k(i,j)\} \\
&\quad + \lambda_2 \min\{u_k(i,j+1) - r, u_k(i,j)\} \\
&\quad + \mu_1 u_k(i-1,j), \qquad i \geq 1, j \geq 0,
\end{aligned}
$$

so that the u_k satisfy the same equation but with holding cost rate increased in proportion with k. Now

$$u_k(0,j) = u_k(0,1) + u_{k+1}(0,j-1), \qquad j \geq 1, \qquad (7.25a)$$

$$u_k(i,j) = u_{k+j}(i,0) + u_k(0,j), \qquad i \geq 1, \ j \geq 1. \qquad (7.25b)$$

It follows that u_k is completely determined by its value at $(0,1)$ and $(i,0)$ and the knowledge of u_{k+1}. Moreover, combining (7.24) and (7.25), we obtain

$$
\begin{aligned}
\mu_2 u_k(0,1) =& c(1+k) \\
&+ \lambda_1 \min\{u_{k+1}(1,0) - r, 0\} \\
&+ \lambda_2 \min\{u_{k+1}(0,1) - r, 0\}, \\
(\lambda_1 + \lambda_2 + \mu_1) u_k(i,0) =& c(i+k) \\
&+ \lambda_1 \min\{u_k(i+1,0) - r, u_k(i,0)\} \\
&+ \lambda_2 \min\{u_{k+1}(i,0) + u_k(0,1) - r, u_k(i,0)\} \\
&+ \mu_1 u_k(i-1,0), \qquad i \geq 1.
\end{aligned} \tag{7.26}
$$

The policy minimizing first passage cost to state $(0,0)$ from starting state (i,k) must first minimize the cost to state $(0,k)$. Let $\pi_k : S_k \mapsto \{0,1\}^2$, denote the optimal flow control minimizing first passage cost to state $(0,k)$, where $S_k = \{(i,j) | i \geq 0, j \geq k\}$. Then by (7.25), $\pi_k = \pi_{k+1}$ on S_{k+1}.

It can be argued that for k large enough, the first passage cost to state $(0,k)$ from all states $(i,j+k)$ is minimized by closing the system to all arrivals until state $(0,k)$ is reached. The optimal flow control π_k is monotonic by default for such k.

The objective is to show that monotonicity filters down through (7.26) from π_{k+1} to π_k, and thus to π_0, the optimal flow control for first passage cost to state $(0,0)$. We were not able to carry out this downward induction program. The difficulties encountered are very similar to those related to the induction

approach of Section 7.1.

7.4 Interchange Arguments

In the preceeding three sections, the emphasis was on the optimality of monotonic flow control; the jointly optimal scheduling was either left not identified, as in the Linear Programming Approach, or assumed, as in the First Passage Cost Approach. In this section we comment on the possibility of adapting *interchange arguments* used for pure scheduling problems to our scheduling and flow control problem.

The use of interchange arguments in scheduling problems goes back to Cox and Smith [17] who applied it to prove that the optimal priority assignment minimizing long-run average costs in $M/G/1$ queues is the μc-rule. The power of interchange arguments was revealed in the seminal contribution of Varaiya, Walrand, and Buyukkok [79] on bandit processes. Basic ideas from [79] were applied to dynamic scheduling of queues in [11], [33], [53], [54].

The basic procedure to establish the optimality of a policy π with a given property p (like the μc-rule say) consists in considering another policy $\tilde{\pi}$ which does not have property p, and showing that π outperforms $\tilde{\pi}$. The comparison between π and $\tilde{\pi}$ usually involves a comparison of sample paths under both policies. This comparison hinges on the independence between arrival processes and service processes in scheduling problem in queues. This assumption is obviously

violated when flow control is exercised together with scheduling. We were not able to circumvent the difficulties arising from the dependence between arrival and service processes, to obtain the optimality of SEJF scheduling and monotonic flow control by interchange arguments.

7.5 Optimal flow control in priority queues

The optimal flow control problem in priority queues has been alluded to in Section 7.3 in relation to the minimization of the expected undiscounted first passage cost until a return to an empty queue operated under SEJF scheduling. Another formulation is examined here, where the state evolution of a priority queue with flow control is observed at decision epochs, namely at arrival epochs.

The model is as in Chapter 2, but with only two classes of jobs, and SEJF scheduling applied; that is, priority is given to type-1 jobs over type-2 jobs.

The memoryless property of the exponential interarrival-time and service-time distribution guarantees that the system has the Markov property at arrival instants, that is, the future evolution of the system depends only on the current state and on the selected control. The resulting decision process is a continuous-time Markov decision process over S.

Let $V_n^{(k)}(i,j)$ denote the minimal expected α-discounted net cost, when a class k job has just arrived to a queue in state (i,j), and the horizon is n, that

is, the system will operate until n additional jobs have arrived and then admit no further jobs. A terminal cost $U_0(i,j)$ is incurred if the state at termination is (i,j). For example, the server might continue to operate until all $i+j$ jobs have been served, in which case $U_0(i,j)$ is the expected α-discounted holding cost until the queue is depleted of the $i+j$ jobs.

We denote by T the interarrival time; T is exponentially distributed with rate $\Lambda = \lambda_1 + \lambda_2$. Then, $V_n^{(k)}(i,j)$ satisfy the following dynamic programming optimality equation:

$$\begin{aligned} V_n^{(1)}(i,j) &= \min\{U_n(i+1,j) - r, U_n(i,j)\}, \\ V_n^{(2)}(i,j) &= \min\{U_n(i,j+1) - r, U_n(i,j)\}, \end{aligned} \tag{7.27}$$

for $n \geq 0$, where

$$\begin{aligned} U_n(i,j) = E\int_0^T e^{-\alpha t} c\|\mathbf{X}(i,j,t)\|dt \\ + (\lambda_1/\Lambda) E e^{-\alpha T} V_{n-1}^{(1)}(\mathbf{X}(i,j,T)) \\ + (\lambda_2/\Lambda) E e^{-\alpha T} V_{n-1}^{(2)}(\mathbf{X}(i,j,T)), \end{aligned} \tag{7.28}$$

where $\mathbf{X}(i,j,t)$ is the population vector-process between two successive arrivals, with coordinates given by

$$\begin{aligned} X_1(i,j,t) &= (i - N_1(t))^+, \\ X_2(i,j,t) &= (j - N_2(t-\tau_i))^+, \end{aligned} \tag{7.29}$$

where τ_i is the random time required to served the i type–1 jobs present immediately after a decision epoch, and $N_1(t)$, $N_2(t)$ are Poisson processes with rate μ_1, μ_2 respectively. $\|\mathbf{X}(i,j,t)\| = X_1(i,j,t) + X_2(i,j,t)$ is the number of jobs in queue t units of time after the last arrival epoch. As usual, $z^+ = \max(0,z)$.

The first term on the right–hand–side of the equation for U_n is the expected α–discounted cost incurred until the next arrival. The remaining sum is the expected present value of the optimal net cost over the remaining $n-1$ decision epochs. It follows that the optimal flow control when n decision epochs remain is:

admit a type–1 job in state (i,j) **iff** $U_n(i+1,j) - U_n(i,j) \leq r$,

admit a type–2 job in state (i,j) **iff** $U_n(i,j+1) - U_n(i,j) \leq r$.

We expect that the optimal admission policy is monotonic (switch–curve) for static priority queues. Monotonicity of the optimal flow control would follow if functions U_n are non–decreasing, supermodular, and convex. Our efforts to establish these properties by an inductive argument based on (7.27) and (7.28) remained unsuccessful. The difficulties encountered are similar to those discussed in Section 7.1.

It should be emphasized in this context that although we anticipate the monotonicity of the optimal flow control in static priority queues, the same is not true for any dynamic scheduling policy. Here is an example of a scheduling policy for which the corresponding optimal flow control is not monotonic; the parameters

are:

$$\mu_1 = 1.0 \qquad \rho_1 = 0.9 \qquad c = 1.0$$

$$\mu_2 = 0.5 \qquad \rho_2 = 0.1 \qquad r = 75.0$$

In Figure 7.3, it should be understood that service is devoted to class 2 in all states $(0, j)$.

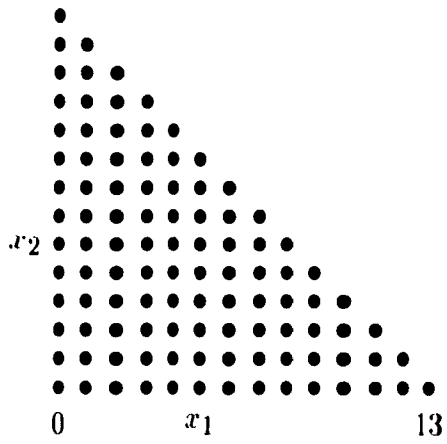

Fig. 7.1 Optimal flow control for type–1 jobs.

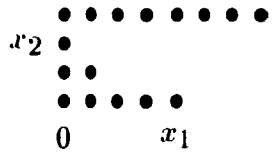

Fig. 7.2 Optimal flow control for type–2 jobs.

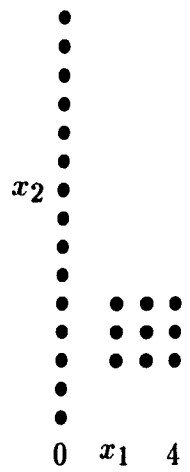

Fig. 7.3 Specified scheduling.

Chapter 8 Generalizations

Two generalizations of the basic model described in Chapter 2 are briefly considered here. First, the Poisson arrival processes are replaced by independent Markov-modulated Poisson arrival processes (MMPP). It appears that SEJF remains optimal, an indication that this optimality is insensitive to the exponential assumption for interarrival times. Second, we consider a network of two multi class single-server queues with feedback where jobs either leave the system or join the other queue upon service completion. Exogeneous traffic is subjected to dynamic flow control and dynamic scheduling is exercised at each node. For an appropriate selection of the routing probabilities, a tandem network is obtained. The optimality of SEJF persists at the *sink* node only. In both the MMPP model and the network model, the jointly optimal flow control is switch curve, but the boundary between acceptance and rejection regions is more complex than in the basic model. Again in this chapter we restrict to two classes of jobs, assuming throughout that $\mu_1 > \mu_2$.

8.1 MMPP arrivals

A Markov-modulated Poisson process (MMPP) is a doubly stochastic Poisson process; that is, a Poisson process whose instantaneous rate is itself a stationary random process which varies according to an irreducible Markov chain with m states. An MMPP is characterized by the infinitesimal generator Q of the underlying Markov process and by the vector of corresponding rates $(\lambda_1, \lambda_2, ..., \lambda_m)$. The MMPP is in *phase* i when the underlying Markov process is in state i. When the MMPP is in phase i, arrivals occur according to a Poisson process of rate λ_i.

We propose here to replace each Poisson arrival streams in the basic model by a two-phase MMPP. Thus, for $k = 1, 2$, type-k jobs arrive according to an MMPP with rates $(\lambda_1^{(k)}, \lambda_2^{(k)})$, and generator

$$Q^{(k)} = \begin{pmatrix} -\sigma_1^{(k)} & \sigma_1^{(k)} \\ \\ \sigma_2^{(k)} & -\sigma_2^{(k)} \end{pmatrix}. \tag{8.1}$$

The state of the system is now characterized by the number of jobs of each type in the system and the phase of each MMPP. Thus, the state space is

$$S = \{\mathbf{x} = (x_1, x_2, i_1, i_2) | x_1 \geq 0, x_2 \geq 0, i_1 \in \{1,2\}, i_2 \in \{1,2\}\}. \tag{8.2}$$

As in the case of Poisson arrivals, control $\mathbf{a} = (a_1, a_2, a_3)$ specifies whether a type k arrival ($k = 1, 2$) should be admitted ($a_k = 1$) or rejected ($a_k = 0$), while a_3 specifies whether service is devoted to type-1 ($a_3 = 1$) or to type-2 ($a_3 = 0$).

Performing uniformization, we obtain a continuous-time Markov decision process on S with state transition probability from $\mathbf{x} = (x_1, x_2, i_1, i_2)$ to $\mathbf{y} = (y_1, y_2, j_1, j_2)$ under action $\mathbf{a} = (a_1, a_2, a_3)$ given by

$$\begin{aligned} p(\mathbf{x}, \mathbf{a}, \mathbf{y})\Lambda = & \lambda_{i_1}^{(1)} a_1 I[\mathbf{y} = A_1\mathbf{x}] + \lambda_{i_2}^{(2)} a_2 I[\mathbf{y} = A_2\mathbf{x}] \\ & + \mu_1 a_3 I[\mathbf{y} = D_1\mathbf{x}] + \mu_2(1 - a_3) I[\mathbf{y} = D_2\mathbf{x}] \\ & + \sigma_{i_1}^{(1)} I[\mathbf{y} = PC_1\mathbf{x}] + \sigma_{i_2}^{(2)} I[\mathbf{y} = PC_2\mathbf{x}], \end{aligned} \tag{8.3}$$

where

$$\Lambda = \lambda_1^{(1)} + \lambda_2^{(1)} + \lambda_1^{(2)} + \lambda_2^{(2)} + \mu_1 + \mu_2 + \sigma_1^{(1)} + \sigma_2^{(1)} + \sigma_1^{(2)} + \sigma_2^{(2)}, \tag{8.4}$$

is the total event rate, $I[\ ]$ the indicator function, and

$$A_1\mathbf{x} = (x_1 + 1, x_2, i_1, i_2),$$

$$A_2\mathbf{x} = (x_1, x_2 + 1, i_1, i_2),$$

$$D_1\mathbf{x} = ((x_1 - 1)^+, x_2, i_1, i_2),$$

$$D_2\mathbf{x} = (x_1, (x_2 - 1)^+, i_1, i_2),$$

$$PC_1\mathbf{x} = (x_1, x_2, 1 + (i_1 mod(2)), i_2),$$

$$PC_2\mathbf{x} = (x_1, x_2, i_1, 1 + (i_2 mod(2))).$$

The transformation PC_1 and PC_2 correspond to phase changes in the underlying process for each class of jobs.

The average cost optimality equation is

$$
\begin{aligned}
\Lambda v(\mathbf{x}) + g =& c\|\mathbf{x}\| \\
&+ \lambda_{i_1}^{(1)} \min\{v(A_1\mathbf{x}) - r, v(\mathbf{x})\} \\
&+ \lambda_{i_2}^{(2)} \min\{v(A_2\mathbf{x}) - r, v(\mathbf{x})\} \\
&+ \sigma_{i_1}^{(1)} v(PC_1\mathbf{x}) + \sigma_{i_2}^{(2)} v(PC_2\mathbf{x}) \\
&+ \min\Big\{\mu_1 v(D_1\mathbf{x}) + \mu_2 v(\mathbf{x}), \mu_1 v(\mathbf{x}) + \mu_2 v(D_2\mathbf{x})\Big\}.
\end{aligned}
\tag{8.5}
$$

From general results in [6], the optimality equation has a unique solution v, g. The stationary policy determined by (8.5) is optimal, and the scalar g is the minimum average cost (independent of the starting state).

Policy iteration was applied to (8.5), as in section 2.2, to determine the optimal combination of flow control and scheduling minimizing long–run average net costs. The optimality of SEJF persists in the case of MMPP arrivals provided again that th flow control is simultaneously optimal. The boundary between acceptance and rejection regions is phase-dependent for each class of jobs. Although some of the linearity exhibited in the case of Poisson arrivals is also observed for MMPP arrivals, the admit/reject boundaries are usually slightly more complex as shown by the following example with parameters:

$$\mu_1 = 1.0 \qquad \lambda_1^{(1)} = 5.0 \qquad \lambda_2^{(1)} = 0$$

$$\mu_2 = 0.5 \qquad \lambda_1^{(2)} = 0.3 \qquad \lambda_2^{(2)} = 0$$

$$\sigma_1^{(1)} = 0.1 \qquad \sigma_2^{(1)} = 0.02 \qquad c = 1.0$$

$$\sigma_1^{(2)} = 0.01 \qquad \sigma_2^{(2)} = 0.01 \qquad r = 30.0$$

For these parameters, the optimal flow control for type-1 jobs in phases $(2,1)$ and $(2,2)$ are both of threshold-type on the number of jobs in queue, with thresholds 18 and 21 respectively. In phase $(1,1)$ it is optimal not to admit any type 2 jobs, while in phase $(1,2)$ it is optimal to admit type-2 jobs only in states $(0,0)$ and $(0,1)$. The remaining optimal acceptance regions are shown in Figures 8.1 to 8.4 below.

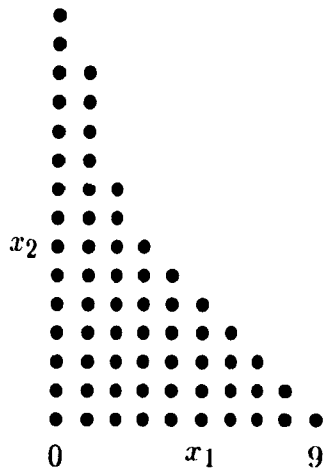

Fig. 8.1 Optimal flow control for type–1 jobs, in phase (1,1).

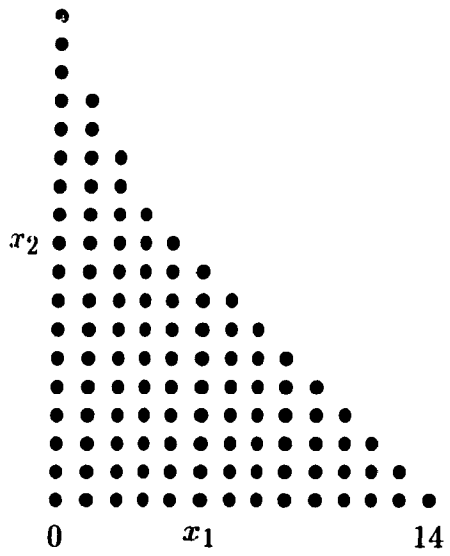

Fig. 8.2 Optimal flow control for type–1 jobs, in phase (1,2).

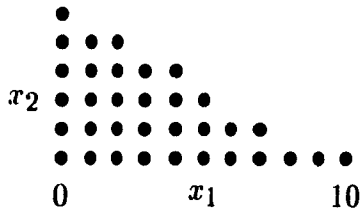

Fig. 8.3 Optimal flow control for type–2 jobs, in phase (2,1).

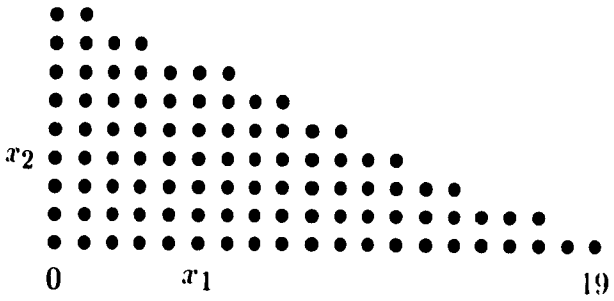

Fig. 8.4 Optimal flow control for type–2 jobs, in phase (2,2).

8.2 Tandem queues

We consider the network of Figure 8.5. Both stations are multi class single server queues with dynamic scheduling and dynamic flow control on the exogeneous traffic. Two classes of tasks use this network. Upon service completion, a job either leaves the network or joins the other queue. Interarrival and service times are exponential and class–dependent. Arrival intensities may be station dependent as well; however, the service rate for a given class of jobs is the same at both nodes. We follow our convention that holding cost rates and admission rewards are class–independent, denoted as usual by c and r respectively.

The state of the system is characterized by the population of jobs of each type at each station. Thus, the state space is

$$S = \{\mathbf{x} = (x_{11}, x_{21}, x_{12}, x_{22}) | x_{ik} \geq 0\}. \tag{8.6}$$

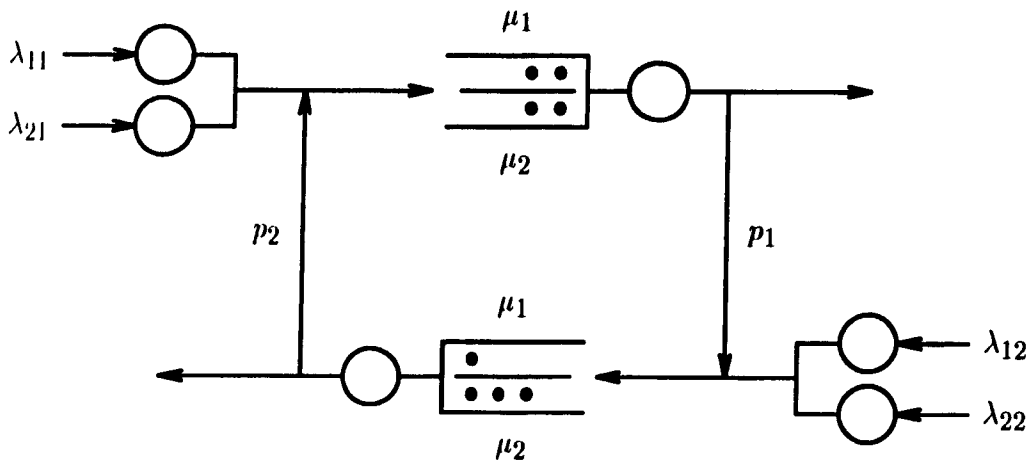

Fig. 8.5 A simple feedback network.

The coordinate x_{ik} refers to the population of class–i tasks at node k.

The control $\mathbf{a} = (a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{32})$ specifies whether a type–i arrival ($i = 1, 2$) should be admitted at node k, ($a_{ik} = 1$) for $k = 1, 2$, or rejected ($a_{ik} = 0$), while a_{3k} specifies whether service is devoted to type-1 ($a_{3k} = 1$) or to type 2 ($a_{3k} = 0$).

Performing uniformization, we obtain a continuous–time Markov decision pro-

cess on S with state transition probability from $\mathbf{x}$ to $\mathbf{y}$ under action $\mathbf{a}$ given by

$$\begin{aligned} p(\mathbf{x},\mathbf{a},\mathbf{y})\Lambda = \sum_{k=1}^{2} & \lambda_{1k}a_{1k}I[\mathbf{y} = A_{1k}\mathbf{x}] + \lambda_{2k}a_{2k}I[\mathbf{y} = A_{2k}\mathbf{x}] \\ & + \mu_1 a_{3k}\Big\{(1-p_k)I[\mathbf{y} = D_{1k}\mathbf{x}] + p_k I[\mathbf{y} = T_{1k}\mathbf{x}]\Big\} \\ & + \mu_2(1-a_{3k})\Big\{(1-p_k)I[\mathbf{y} = D_{2k}\mathbf{x}] + p_k I[\mathbf{y} = T_{2k}\mathbf{x}]\Big\} \end{aligned} \tag{8.7}$$

where

$$\Lambda = \sum_{k=1}^{2} (\lambda_{1k} + \lambda_{2k} + \mu_1 + \mu_2) \tag{8.8}$$

is the total event rate, $I[\ ]$ the indicator function, p_k is the probability of a transfer from node k to the other node (independent of class). The state transformations A_{ik} and D_{ik} correspond as usual to arrivals and departures of type i jobs at node k. The transformations T_{ik} correspond to the transfer of a type i job from node k to the other node; for example $T_{12}(\mathbf{x})$ is equal to $(x_{11}+1, x_{21}, x_{12}-1, x_{22})$ if $x_{12} > 0$, and to $\mathbf{x}$ if $x_{12} = 0$.

The average cost optimality equation is

$$\Lambda v(\mathbf{x}) + g = c\|\mathbf{x}\| + \sum_{k=1}^{2}\Big\{\lambda_{1k}\min\{v(A_{1k}\mathbf{x}) - r, v(\mathbf{x})\}$$

$$+ \lambda_{2k}\min\{v(A_{2k}\mathbf{x}) - r, v(\mathbf{x})\}$$

$$+ \min\Big\{\mu_1\Big[p_k v(T_{1k}\mathbf{x}) + (1-p_k)v(D_{1k}\mathbf{x})\Big] + \mu_2 v(\mathbf{x}),$$

$$\mu_1 v(\mathbf{x}) + \mu_2\Big[p_k v(T_{2k}\mathbf{x}) + (1-p_k)v(D_{2k}\mathbf{x})\Big]\Big\}\Big\}. \tag{8.9}$$

From general results in [6], the optimality equation has a unique solution v, g. The stationary policy determined by (8.9) is optimal, and the scalar g is the minimum average cost (independent of the starting state).

An application of policy iteration as in section 2.2 is possible only to a small size network. The number of states grows very rapidly with the maximum number of tasks accepted into the system. For example there are 135751 states when the network total population cannot exceed 40. Value iteration [77] was used to obtain c optimal flow control and scheduling for this network.

Setting $p_1 = 1$ and $p_2 = 0$, we obtain a multi-class tandem network with flow control and scheduling at each node. This is a multi-class version of the model studied in [23]. From an application of value iteration to (8.9) for numerous sets of parameters, it was observed that SEJF remains optimal at the *sink* node only,

provided that the flow control at each node and scheduling at the *source* node are simultaneously optimal. When flow control is not exercised (infinite queues), the optimality of SEJF (of the μc–rule in fact) at the *sink* node was established in [53] and [54].

The optimal acceptance regions are characterized by monotonic surface C_{ik} in the four-dimensional state-space, one such surface for each pair (job class i, node k). Some of the linearity found in the single-node model remains in the tandem network; however the boundary between acceptance and rejection regions in the tandem network is slightly more complex in general.

When feedback is allowed, $p_1 > 0$ and $p_2 > 0$, the model is a multi class version of the one considered in [14]. SEJF ceases to be optimal in general, but optimal flow controls are monotonic.

Chapter 9 Conclusion

Congestion control is an integral part of the design and analysis of telecommunication networks. The search for dynamic procedures optimizing some figure of merit has produced a substantial literature on the dynamic control of queueing systems. Flow control, scheduling, and routing are the main constituents of congestion control. It is only very recently that the combined optimization of these components has been undertaken ([41], [82], [87]).

9.1 This thesis

This thesis introduced the simultaneous optimization of scheduling and flow control in multi-class single-server queues. It addressed in particular the optimality of static priority rules in the presence of flow control.

A Markovian model was proposed, with a cost/benefit structure reflecting the fact that delay is bad and throughput is good. General results from the theory of Markov decision processes were invoked to characterize the jointly optimal

schedule and flow control.

Examples were provided to show that the pervasive μc rule is not optimal in general when flow control is exercised, even when flow control is jointly optimal. It was observed that for class-dependent waiting cost rates the optimal scheduling in the presence of flow control may be rather complex.

In the special case of class-independent user-fees and waiting cost rates, the problem amounts to characterizing the combination of preemptive scheduling and flow control optimizing the tradeoff between delay and throughput. In this context, we considered first the special case of two queues, one of which is saturated. In that setting, we provided a direct proof of the optimality of static priority rules when combined to simultaneously optimal flow control. The optimal scheduling is the SEJF rule (Shortest Expected Job First) while the optimal flow control is a control-limit admission rule.

Proving that a similar result persists in the general case of two independent Poisson arrival processes presented unexpected difficulties. The Linear Programming Approach introduced in [61] and generalized in [80] fails in this context. Several other techniques, among which the Inductive Approach, do not apply easily; the nature of the difficulties are pointed out.

In view of these analytical difficulties, an extensive computational investiga-

tion was undertaken. It led to the conjecture that SEJF remains optimal for independent Poisson arrival processes, when coupled to simultaneously optimal flow control. This computation also revealed that the jointly optimal monotonic flow control is characterized by switching curves that are very nearly linear. It was also observed that the class of window flow controls is only slightly suboptimal, in that the overall optimal tradeoff between delay and throughput can be achieved very closely by window policies. This observation illuminates the relative value of state feedback in the control of queueing systems, a topic that warrants further research.

An approximate performance analysis was derived. The approximation is based on the near- linearity of the optimal acceptance/rejection boundary and on a steady state assumption. The approximation is shown to be very robust and capable of remarkable accuracy.

Several extensions and generalizations were considered to examined the sensitivity of the above conjecture to the underlying assumptions. The simultaneous optimality of SEJF and monotonic flow control persists when there are more than two classes of tasks; moreover, the optimal acceptance/rejection boundaries are very nearly linear. The claim also seems to be insensitive to the exponential assumption about interarrival times, as its validity extends to MMPP arrivals.

9.2 Future research

The difficulties related to the proof of the optimality of SEJF scheduling coupled to monotonic flow control must be resolved. This requires either that the difficulties encountered in the application of well-established techniques be circumvented or that a new approach be used to clarify the dependencies between arrival and service processes.

We have studied preemptive scheduling to model the possibility of interweaving multi-packet messages. Preemption is not always possible or desirable in certain context. The optimization of nonpreemptive scheduling in the presence of dynamic flow control is an open research problem. Nonpreemptive static priority rules are known to be optimal in a wide variety of models. As for preemptive scheduling, this optimality rests on the independence between service and arrival processes, a condition that is violated when flow control is exercised. As in this thesis, the goal is to determine to what extent and in what context this condition can be relaxed.

The subject of optimal flow control in priority queues is closely related to our problem. In view of the importance of priority queues, and of the abundant literature on optimal flow control, it is somewhat surprising that the problem of dynamic flow control in priority queues has not been addressed yet. Much remains to be done in this area.

Optimal scheduling and optimal flow control of messages with real–time constraints is a research topic of current interest. So far however, each control mechanism has been studied in the absence of the other. A generalization to constrained simultaneous optimization of scheduling and flow control is an open problem.

Appendix A

The saturated queue: verification of the optimality of π_0^{opt}

This appendix supplements the analysis of Chapter 3. We sketch the verification that policy π_0^{opt} is overall optimal when $g_0 < g_1$; this requires that the relative values $v_0(i,j)$ corresponding to π_0^{opt} be shown to satisfy the optimality equation (3.1). That π_1^{opt} is overall optimal when $g_1 < g_0$ involves similar elementary manipulations that will not be presented here. For simplicity, we drop the superscript *opt* of π_0^{opt} in the following.

We rewrite the dynamic programming optimality equation as

$$v(i,j) = \min_{(a_1,a_2)\in\{0,1\}^2} Z(i,j,a_1,a_2), \tag{A.1}$$

where $Z(\cdot,\cdot,\cdot,\cdot)$ is defined by equation (3.4). By definition,

$$v_0(i,j) = Z(i,j,\pi_0(i,j)). \tag{A.2}$$

To establish the optimality of π_0, we need to verify that for all $(i,j) \in S$,

$$v_0(i,j) \le Z(i,j,a_1,a_2), \qquad (a_1,a_2) \neq \pi_0(i,j). \tag{A.3}$$

In view of the definition of π_0, the verification can be divided into four cases, two of which will be considered in greater details.

A.1 Optimality of π_0 in state $(i,0)$, $i \geq n_0$.

The action specified by π_0 in states $(i,0)$ for $i \geq n_0$ is $\pi_0(i,0) = (0,1)$, that is (reject type-1 arrivals, serve type-1 jobs). To show that this is the best choice, we compare to the alternatives.

A.1.1 Action $(0,1)$ is better than action $(1,1)$.

Action $(0,1)$ is preferred to action $(1,1)$ (accept type-1 arrivals, serve type-1) if

$$v_0(i,0) \leq \frac{ci - g_0}{\lambda_1 + \mu_1} + \frac{\lambda_1}{\lambda_1 + \mu_1}[v_0(i+1,0) - r] + \frac{\mu_1}{\lambda_1 + \mu_1} v_0(i-1,0), \qquad (A.4)$$

which is equivalent to

$$0 \leq ci - g_0 - \mu_1 [v_0(i,0) - v_0(i-1,0)] + \lambda_1 [v_0(i+1,0) - v_0(i,0) - r]. \qquad (A.5)$$

But

$$v_0(i,0) - v_0(i-1,0) = \frac{ci - g_0}{\mu_1}, \qquad i \geq n_0, \qquad (A.6)$$

which combined with

$$v_0(i+1,0) - v_0(i,0) = \frac{c(i+1) - g_0}{\mu_1} \geq \frac{c(n_0+1) - g_0}{\mu_1} > r, \qquad (A.7)$$

proves (A.1); the last inequality in (A.7) comes from (3.21).

A.1.2 Action $(0,1)$ is better than action $(0,0)$.

Action $(0,1)$ is preferred to action $(0,0)$ (reject type-1 arrivals, introduce a type-2 job in service) if

$$v_0(i,0) \leq \frac{c(i+1)-g_0}{\mu_2} - r + v_0(i,0), \tag{A.8}$$

that is

$$\mu_2 r \leq c(i+1) - g_0, \tag{A.9}$$

which follows from equation 3. and

$$\mu_2 r < \mu_1 r < c(n_0+1) - g_0 \leq c(i+1) - g_0. \tag{A.10}$$

A.1.3 Action $(0,1)$ is better than action $(1,0)$.

Action $(0,1)$ is preferred to action $(1,0)$ (accept type-1 arrivals, introduce a type-2 job in service) if

$$v_0(i,0) \leq \frac{c(i+1)-g_0}{\lambda_1+\mu_2} - r + \frac{\lambda_1}{\lambda_1+\mu_2}\left[v_0(i+1,1) - r\right] + \frac{\mu_2}{\lambda_1+\mu_2} v_0(i,0), \tag{A.11}$$

which is equivalent to

$$\begin{aligned} 0 \leq [c(i+1) - g_0 - \mu_2 r] &+ \lambda_1 \left[v(i+1,1) - v(i+1,0) - r\right] \\ &+ \lambda_1 \left[v(i+1,0) - v(i,0) - r\right]. \end{aligned} \tag{A.12}$$

Invoking (3.) again we have

$$c(i+1) - g_0 - \mu_2 r \geq c(n_0+1) - g_0 - \mu_2 r > (\mu_1 - \mu_2) r > 0. \tag{A.13}$$

After substitution and simplification we obtain, for $i \geq n_0 + 1$,

$$v_0(i,1) - v_0(i,0) = [c - g_0 + \phi_0(n_0 - 1)] \left(\frac{1}{\mu_2} - \frac{1}{\mu_1}\right) + \frac{c(i+1) - g_0}{\mu_1}. \tag{A.14}$$

By definition of n_0, $\phi(n_0 - 1) > \phi(n_0) = g_0$; combined with (A.6), this yields that second and third terms of the right-hand side of (A.12) are positive. Thus (A.12) verifies.

A.2 Optimality of π_0 in state $(i,0)$, $0 \leq i < n_0$.

The action specified by π_0 in states $(i,0)$ for $0 \leq i < n_0$ is $\pi_0(i,0) = (1,1)$, that is (accept type-1 arrivals, serve type-1 jobs). To show that this is the best choice, we compare to the alternatives.

A.2.1 Action $(1,1)$ is better than action $(0,1)$.

Action $(1,1)$ is preferred to action $(0,1)$ (reject type-1 arrivals, serve type-1 jobs) if

$$v_0(i,0) \leq \frac{ci - g_0}{\mu_1} + v_0(i-1,0) \qquad (A.15)$$

that is if

$$K(i,n_0,g_0) \leq \frac{ci - g_0}{\mu_1} + K(i-1,n_0,g_0) \qquad (A.16)$$

Substituting for $K(\cdot,\cdot,\cdot)$ and using (3.) which states that

$$\mu_1 r \geq c\left\{\frac{n_0}{1-\rho} - \frac{\rho - \rho^{n_0+1}}{(1-\rho)^2}\right\} \qquad (A.17)$$

inequality (A.16) reduces after simplification to

$$0 \leq (n_0 - i)(1-\rho) - (1 - \rho^{n_0 - i}) \qquad (A.18)$$

which verifies.

A.2.2 Action $(1,1)$ is better than action $(0,0)$

Action $(1,1)$ is preferred to action $(0,0)$ (reject type-1 arrivals, introduce a type-2 job in service) if

$$v_0(i,0) \leq \frac{c(i+1)-g_0}{\mu_2} - r + v_0(i,0) \qquad (A.19)$$

that is if

$$c(i+1) - \mu_2 r \geq g_0, \qquad 0 \leq i \leq n_0 - 1 \qquad (A.20)$$

But

$$c - \mu_2 r = \phi_1(0) \geq \phi_1(n_1) = g_1 > g_0. \qquad (A.21)$$

Thus (A.18) verifies.

A.2.3 Action $(1,1)$ is better than action $(1,0)$

Action $(1,1)$ is preferred to action $(1,0)$ (accept type-1 arrivals, introduce a type-2 job in service) if

$$v_0(i,0) \leq \frac{c(i+1)-g_0}{\lambda_1+\mu_2} - r + \frac{\lambda_1}{\lambda_1+\mu_2}\left[v_0(i+1,1) - r\right] + \frac{\mu_2}{\lambda_1+\mu_2} v_0(i,0) \quad (A.22)$$

This case needs to be further subdivided according to whether $i = 0$, or $1 \leq i \leq n_0 - 2$, or $i = n_0 - 1$. We will consider the case $i = 0$ only.

Since $v_0(0,0) = 0$, inequality (A.22) reduces to

$$0 \leq c - g_0 - (\lambda_1+\mu_2)r + \lambda_1\left[v_0(1,1) - r\right] \qquad (A.23)$$

After substitution and simplification, this becomes

$$0 \leq \left(1 + \frac{\lambda_1}{\mu_2}\right)\left(\frac{1-\rho^{n_0}}{1-\rho}\right)[\phi_1(n_0-1) - \phi_0(n_0)] \qquad (A.24)$$

which follows from

$$\phi_1(n_0-1) \geq \phi_1(n_1) = g_1 > g_0 = \phi_0(n_0) \qquad (A.25)$$

the first inequality resulting from the definition of n_1 (3.), the second inequality being the basic assumption.

The optimality of π_0 in state $(i,0)$, for $0 \leq i < n_0$, and in states $(i,1)$, $0 \leq i < n_0 - 1$ is established similarly.

Appendix B

Proofs of Propositions 7.1 to 7.7

In this appendix we sketch the elementary proofs of the propositions in section 7.1. We recall that for a real-valued function f and a real constant $r > 0$, the transformations $T_1 f$ and $T_2 f$ are defined by

$$T_1 f(x_1, x_2) = \min\{f(x_1 + 1, x_2) - r, f(x_1, x_2)\},$$

$$T_2 f(x_1, x_2) = \min\{f(x_1, x_2 + 1) - r, f(x_1, x_2)\}.$$

Proposition 7.1

If f is nondecreasing in x_1 and in x_2,

then $T_1 f$ and $T_2 f$ are nondecreasing in x_1 and in x_2.

Proof: Obvious.

□

Proposition 7.2

If f is supermodular, subconvex in x_1, and subconvex in x_2,

then f is convex in x_1 and in x_2.

Proof: The convexity of f in x_1 follows from the sequence of inequalities

$$f(x_1+1,x_2)-f(x_1,x_2) \le f(x_1+1,x_2+1)-f(x_1,x_2+1), \qquad (B.1)$$

$$\le f(x_1+2,x_2)-f(x_1+1,x_2), \qquad (B.2)$$

where (B.1) is due to supermodularity of f, and (B.2) to subconvexity of f in x_1. The proof that f is convex in x_2 is symmetric.

□

Proposition 7.3

If f is supermodular, convex in x_1 and convex in x_2,

then $T_1 f$ and $T_2 f$ are supermodular for all $r \ge 0$.

Proof: We want to show that

$$T_1(x_1+1,x_2)-T_1(x_1,x_2) \le T_1(x_1+1,x_2+1)-T_1(x_1,x_2+1). \qquad (B.3)$$

Label point (x_1,x_2) with A or R according to whether $f(x_1+1,x_2)-f(x_1,x_2)$ is $\le r$ or $> r$, with the obvious interpretation of admission or rejection of type–1 jobs in the context of finite–horizon discounted costs. Since f is convex in x_1 and in x_2, there are only six cases to consider in (B.3), namely

AA	AR	AR	RR	RR	RR
AA,	AA,	AR,	AA,	AR,	RR.

Inequality (B.3) follows directly from the supermodularity of f in the first and sixth cases. In the second case, (B.3) becomes

$$f(x_1+2,x_2)-f(x_1+1,x_2)\leq r, \qquad (B.4)$$

which verifies because of the lower right A. In the third case, both terms of (B.3) are equal to r. In the fourth case, (B.3) becomes

$$f(x_1+2,x_2)-f(x_1+1,x_2)\leq f(x_1+1,x_2+1)-f(x_1,x_2+1), \qquad (B.4)$$

which verifies because lower right A and upper left R means

$$f(x_1+2,x_2)-f(x_1+1,x_2)\leq r<f(x_1+1,x_2+1)-f(x_1,x_2+1). \qquad (B.5)$$

Finally, in the fifth case, (B.3) becomes

$$r\leq f(x_1+1,x_2+1)-f(x_1,x_2+1), \qquad (B.6)$$

which verifies because of upper left R. The proof that T_2f is supermodular is symmetric.

□

Proposition 7.4

If f is convex in x_1,

then T_1f is convex in x_1 for all $r\geq 0$.

Proof: We want to show that

$$T_1(x_1+1,x_2)-T_1(x_1,x_2)\leq T_1(x_1+2,x_2)-T_1(x_1+1,x_2). \qquad (B.7)$$

Using the labelling introduced above, the convexity of f reduces the number of cases to four, namely AAA, AAR, ARR, RRR. In the first and last cases, (B.7) follows directly from the convexity of f. In the second case, (B.7) becomes

$$f(x_1 + 2, x_2) - f(x_1 + 1, x_2) \leq r, \qquad (B.8)$$

which verifies by middle A. Finally, in the third case, (B.7) becomes

$$r \leq f(x_1 + 2, x_2) - f(x_1 + 1, x_2), \qquad (B.9)$$

which verifies by middle R.

□

Proposition 7.5

If f is convex in x_2,

then $T_2 f$ is convex in x_2 for all $r \geq 0$.

Proof: The proof is symmetric to the one for Proposition 7.4.

□

Proposition 7.6

For f nondecreasing and supermodular,

then $T_1 f$ is convex in x_2 for all $r \geq 0$ iff f is subconvex in x_2.

Proof: We first prove the "if" statement, that is we want to show that

$$T_1(x_1, x_2 + 1) - T_1(x_1, x_2) \leq T_1(x_1, x_2 + 2) - T_1(x_1, x_2 + 1). \qquad (B.10)$$

Because f is supermodular, only four cases need to be considered in (B.10), namely

A	R	R	R
A	A	R	R
A,	A,	A,	R.

In the first and last cases, (B.10) follows from the convexity of f in x_2 which itself results from the supermodularity and the subconvexity in x_2 of f. In the second case, (B.10) becomes

$$f(x_1+1,x_2+1)-f(x_1+1,x_2)\le f(x_1,x_2+2)-f(x_1+1,x_2+1)+r. \quad (B.11)$$

But

$$\begin{aligned} f(x_1+1,x_2+1)-f(x_1+1,x_2) &\le f(x_1,x_2+2)-f(x_1,x_2+1) && (B.12)\\ &\le [f(x_1,x_2+2)-f(x_1+1,x_2+1)] \\ &\quad +[f(x_1+1,x_2+1)-f(x_1,x_2+1)] && (B.13)\\ &\le f(x_1,x_2+2)-f(x_1+1,x_2+1)+r && (B.11) \end{aligned}$$

where (B.12) is subconvexity in x_2, (B.13) is (B.12) with a term added and subtracted, and (B.11) comes from the middle A. In the third case, (B.10) becomes

$$f(x_1,x_2+1)-f(x_1+1,x_2)+r\le f(x_1,x_2+2)-f(x_1,x_2+1) \quad (B.15)$$

But

$$\begin{aligned} f(x_1,x_2+1)-f(x_1+1,x_2)+r &\le [f(x_1,x_2+1)-f(x_1+1,x_2)] \\ &\quad +[f(x_1+1,x_2+1)-f(x_1,x_2+1)] && (B.15)\\ &\le f(x_1+1,x_2+1)-f(x_1+1,x_2) && (B.16)\\ &\le f(x_1,x_2+2)-f(x_1,x_2+1) && (B.17) \end{aligned}$$

where (B.15) comes from the middle R, (B.16) is just (B.15) after simplification, and (B.17) is subconvexity in x_2.

We now prove the "only if" part of the claim. We proceed by contradiction, assume that f is not subconvex in x_2, and show that a value of r can be found such that $T_1 f$ is not convex in x_2. Suppose there is a point (x_1, x_2) such that

$$f(x_1, x_2+2) - f(x_1, x_2+1) < f(x_1+1, x_2+1) - f(x_1+1, x_2). \qquad (B.18)$$

By supermodularity, we have that

$$\begin{aligned} f(x_1+1, x_2) - f(x_1, x_2) &\le f(x_1+1, x_2+1) - f(x_1, x_2+1) \\ &\le f(x_1+1, x_2+2) - f(x_1, x_2+2) \end{aligned} \qquad (B.19)$$

Select $r = f(x_1+1, x_2+1) - f(x_1, x_2+1)$. Then

$$\begin{aligned} T_1 f(x_1, x_2) &= f(x_1+1, x_2) - r, \\ T_1 f(x_1, x_2+1) &= f(x_1+1, x_2+1) - r \\ &= f(x_1, x_2+1), \\ T_1 f(x_1, x_2+2) &= f(x_1, x_2+2), \end{aligned} \qquad (B.20)$$

which implies that

$$T_1(x_1, x_2+2) - T_1(x_1, x_2+1) = f(x_1, x_2+2) - f(x_1, x_2+1) \qquad (B.21)$$

and

$$T_1(x_1, x_2+1) - T_1(x_1, x_2) = f(x_1+1, x_2+1) - f(x_1+1, x_2). \qquad (B.22)$$

Combining (B.18), (B.21), and (B.22), we obtain that

$$T_1(x_1, x_2+1) - T_1(x_1, x_2) > T_1(x_1, x_2+2) - T_1(x_1, x_2+1) \qquad (B.23)$$

a contradiction to the convexity of f in x_2.

□

Proposition 7.7

For f nondecreasing and supermodular,

then $T_2 f$ is convex in x_1 for all $r \geq 0$ iff f is subconvex in x_1.

Proof: The proof is symmetric to the one for Proposition 7.5.

□

References

[1] Baras J.S., Ma D.J., Makowsky A.M., "K competing queues with geometric service requirements and linear costs: the μc rule is always optimal", *Systems & Control Letters*, 6 (1985) 173–180.

[2] Bertsekas D.P., *Dynamic programming and stochastic control*, Academic Press, New York (1976).

[3] Bhattacharya P.P., Ephremides A., "Optimal scheduling with strict deadlines", *IEEE Trans. on Automatic Control*, 34 (1989) 721-728.

[4] Bolotin V.A., Kappel J.G., Kuehn P.J., (Editors), "Teletraffic analysis of communications systems", *IEEE J. on Selected Areas in Communications*, 9 (1991).

[5] Bolotin V.A., Kappel J.G., Kuehn P.J., (Editors), "Teletraffic analysis of ATM systems", *IEEE J. on Selected Areas in Communications*, 9 (1991).

[6] Borkar V.S., "Controlled Markov chains and stochastic networks", *SIAM J. Control and Optimization*, 21 (1983) 652-666.

[7] Borkar V.S., "On minimum cost per unit time control of Markov chains", *SIAM J. Control and Optimization*, 22 (1984) 965–978.

[8] Borkar V.S., "Control of Markov chains with long-run average cost criterion", *in Stochastic Differential Systems, Stochastic Control Theory and Applications*, W. Fleming and P.L. Lions (Editors), IMA Vol. 10, Springer–Verlag, New York, (1988) 57–77.

[9] Borkar V.S., "Control of Markov chains with long-run average cost criterion: the dynamic programming equations", *SIAM J. Control and Optimization*, 27 (1989) 612-657.

[10] Browne S., Yechiali U., "Dynamic priority rules for cyclic-type queues", *Advances in Applied Probability*, 21 (1989) 432–450.

[11] Buyukkok C., Varaiya P.P., Walrand J., "The $c\mu$ rule revisited", *Advances in Applied Probability*, 17 (1985) 237–238.

[12] Chen H., "Optimal intensity control of a multi class queue", *Queueing Systems*, 5 (1989) 281-294.

[13] Chen T.M., Walrand J., Messerschmitt D.G., "Dynamic priority protocols for packet voice", *IEEE J. on Selected Areas in Communications*, 7 (1989) 632-643.

[14] Christidou I., Lambadaris I., Mazumdar R., "Optimal control of arrivals to a feedback queueing system", *Proc. of the 27th Conf. on Decision and Control*, Austin, Texas, (1988) 663-667.

[15] Coffman E.G., Mitrani I., "A characterization of waiting time performance realizable by single server queues", *Operations Research*, 28 (1980) 810-821.

[16] Courcoubetis C.A., Reiman M.I., "Optimal control of a queueing system with simultaneous service requirements", *IEEE Trans. on Automatic Control*, 32 (1987) 717-727.

[17] Cox D., Smith W., *Queues*, Methuen, London (1961).

[18] De Serres Y., "Simultaneous optimization of flow control and scheduling in a single server queue with two job classes", *Operations Research Letters*, 10 (1991) 103-112.

[19] De Serres Y., "Simultaneous optimization of flow control and scheduling in a single-server queue with two job classes: numerical results and approximation", *Computers & Operations Research*, 18 (1991) 361-378.

[20] Ephremides A., Verdú S., "Control and optimization methods in communication network problems", *IEEE Trans. on Automatic Control*, 34 (1989) 930-942.

[21] Federgruen A., Groenevelt H., "*M/G/c* queueing systems with multiple customer classes: characterization and control of achievable performance under nonpreemptive priority rules", *Management Science*, 34 (1988) 1121-1138.

[22] George A., Liu J., Ng E., "User guide for SPARSPAK", *Department of Computer Science*, University of Waterloo, Waterloo, Ontario, Canada (1980).

[23] Ghoneim H.A., Stidham S., "Control of arrivals to two queues in series", *European J. of Operational Research*, 21 (1985) 399–409.

[24] Gittins J.C., "Bandit processes and dynamic allocation indices", *J. Royal Statistical Soc. B*, 11 (1979) 148–177.

[25] Gittins J.C., Nash P., "Scheduling, queues, and dynamic allocation indices", *Proc. EMS, Prague 1974*, Prague: Czechoslovak Academy of Sciences, (1977) 191–202.

[26] Hajek B., "Optimal control of two interacting service stations", *IEEE Trans. on Automatic Control*, 29 (1984) 491–499.

[27] Hariharan R., Kulkarni V.G., Stidham S., "Optimal control of two parallel infinite server queues", *Proc. of the 29th Conf. on Decision and Control*, Honolulu, Hawaii, (1990) 1329–1335.

[28] Harrison J.M., "Dynamic scheduling of a multiclass queue: discount optimality", *Operations Research*, 23 (1975) 270–282.

[29] Harrison J.M., "Brownian models of queueing networks with heterogeneous customer populations", *in Stochastic Differential Systems, Stochastic Control Theory and Applications*, W. Fleming and P.L. Lions (Editors), IMA Vol. 10, Springer Verlag, New York, (1988) 147–186.

[30] Harrison J.M., Wein L.M., "Scheduling networks of queues: heavy traffic analysis of a simple open network", *Queueing Systems*, 5 (1989) 265–280.

[31] Harrison J.M., Wein L.M., "Scheduling networks of queues: heavy traffic analysis of a two-station closed network", *Operations Research*, 38 (1990) 1052–1064.

[32] Hirayama T., "Optimal service assignment in a finite-source queue", *IEEE Trans. on Automatic Control*, 34 (1989) 67–75.

[33] Hirayama T., Kijima M., Nishimura S., "Further results for dynamic scheduling of multiclass $G/G/1$ queues", *J. of Applied Probability*, 26 (1989) 595–603.

[31] Hsiao M.T., Lazar A.A., "Optimal flow control of multiclass queueing networks with partial information", *IEEE Trans. on Automatic Control*, 35 (1990) 855-860.

[35] Johansen S.G., Stidham S., "Control of arrivals to a stochastic input output system", *Advances in Applied Probability*, 12 (1980) 972-999.

[36] Kakumanu P., "Relation between continuous and discrete time Markovian decision problems", *Naval Research Logistics Quarterly*, 24 (1977) 131-139.

[37] Kim B.G., Towsley D., "Dynamic flow control protocols for packet switching multiplexers serving real-time multipacket messages", *IEEE Trans. on Communications*, 31 (1986) 318-356.

[38] Kleinrock L., *Queueing Systems*, Vol. 2, Wiley, New York (1976).

[39] Klimov G.P., "Time sharing service systems", *Theory of Probability and its Applications*, Part I, 19 (1974) 532-551; Part II, 23 (1978) 314-321.

[40] Kuang L., Tsoucas P., "Scheduling a server in a network: from discounted to average cost", *Proc. of the 28th Conf. on Decision and Control*, Tampa, Florida, (1989) 1552-1554.

[41] Lambadaris I., Narayan P., "Optimal routing and admission control at a simple data network node", *Preprint* (1990).

[42] Lazar A.A., "The throughput time delay function of an $M/M/1$ queue", *IEEE Trans. on Information Theory*, 29 (1983) 914-918.

[43] Lazar A.A., "Optimal flow control of an $M/M/m$ queue", *J. of the Association for Computing Machinery*, 31 (1984) 86-98.

[44] Lazar A.A., "Optimal flow control of a class of queueing networks in equilibrium", *IEEE Trans. on Automatic Control*, 28 (1983) 1001-1007.

[45] Lippman S.A., "Semi Markov decision processes with unbounded rewards", *Management Science*, 19 (1973) 717-731.

[46] Lippman S.A., "On dynamic programming with unbounded rewards", *Management Science*, 21 (1975) 1225-1223.

[17] Lippman S.A., "Applying a new device in the optimization of exponential queueing systems", *Operations Research*, 23 (1975) 687–710.

[18] Lippman S.A., Stidham S., "Individual versus social optimization in exponential congestion systems", *Operations Research*, 25 (1977) 233–247.

[49] Luh P., Viniotis I., "Optimality of threshold policies for heterogeneous server systems", *Proc. of the 29th Conf. on Decision and Control*, Honolulu, Hawaii, (1990) 870–875.

[50] Maxemchuk N.F., Zarki M.E., "Routing and flow control in high-speed wide area networks", *Proc. of the IEEE*, 78 (1990) 204–221.

[51] Meilijson I., Yechiali U., "On the optimal right-of-way policies at a single-server station when insertion of idle times is permitted", *Stochastic Processes and their Applications*, 6 (1977) 25–32.

[52] Milito R.A., Levy H., "Modeling and dynamic scheduling of a queueing system with blocking and starvation", *IEEE Trans. on Communications*, 37 (1989) 1318–1329.

[53] Nain P., "Interchange arguments for classical scheduling problems in queues", *Systems & Control Letters*, 12 (1989) 177–184.

[54] Nain P., Tsoucas P., Walrand J., "Interchange arguments in stochastic scheduling", *J. of Applied Probability*, 27 (1989) 815–826.

[55] Nain P., Ross K.W., "Optimal priority assignment with hard contraint", *IEEE Trans. on Automatic Control*, 31 (1986) 883–888.

[56] Naor P., "The regulation of queue size by levying tolls", *Econometrica*, 37 (1969) 15–24.

[57] Panwar S.S., Towsley D., Wolf J.K., "Optimal scheduling policies for a class of queues with customer deadlines to the beginning of service", *J. of the Association for Computing Machinery*, 35 (1988) 832–844.

[58] Régnier J., Humblet P.A., "Average waiting time assignment", *IEEE Trans. on Communications*, 38 (1990) 2049–2072.

[59] Robertazzi T.G., Lazar A.A., "On the modeling and optimal flow control of the Jacksonian network", *Performance Evaluation*, 5 (1985) 29 43.

[60] Rosberg Z., "Process scheduling in a computer system", *IEEE Trans. on Computers*, 34 (1985) 633 645.

[61] Rosberg Z., Varaiya P.P., Walrand J., "Optimal control of service in tandem queues", *IEEE Trans. on Automatic Control*, 27 (1982) 600 610.

[62] Ross K.W., Yao D.D., "Optimal dynamic scheduling in Jackson networks", *IEEE Trans. on Automatic Control*, 31 (1989) 17 53.

[63] Ross K.W., Chen B., "Optimal scheduling of interactive and noninteractive traffic in telecommunication systems", *IEEE Trans. on Automatic Control*, 33 (1988) 261 267.

[64] Ross S.M., *Introduction to stochastic dynamic programming*, Academic Press, New York (1983).

[65] Schäl M., "Conditions for optimality in dynamic programming and for the limit of n stage optimal policies to be optimal", *Z. Wahrscheinlichkeitstheorie verw. Gebiete*, 32 (1975) 179 196.

[66] Schrijver A., *Theory of Linear and Integer Programming*, John Wiley & Sons, New York (1986).

[67] Serfozo R.F., "An equivalence between continuous and discrete time Markov decision processes", *Operations Research*, 27 (1979) 616 620.

[68] Shimkin N., Shwartz, A., "Control of admission and routing in parallel queues operating in a random environment", *Proc. of the 28th Conf. on Decision and Control*, Tampa, Florida, (1989) 1064 1065

[69] Stidham S., "Scheduling, routing and flow control in stochastic networks", in *Stochastic differential systems, stochastic control theory and applications*, IMA Vol. 10, W. Fleming and P.L. Lions (Editors), Springer Verlag, Berlin, (1988) 529 561.

[70] Stidham S., "Optimal control of admission to a queueing system", *IEEE Trans. on Automatic control*, 30 (1985) 705 713.

[71] Stidham S., "Socially and individually optimal control of arrivals to a $GI/M/1$ queue", *Management Science*, 24 (1978) 1598–1610.

[72] Stidham S., Prabhu N.U., "Optimal control of queueing systems", *in Mathematical Methods in Queueing Theory*, A.B. Clarke (Editor), Springer Verlag, Berlin, (1974) 263–294.

[73] Stidham S., Weber R.R., "Monotonic and insensitive optimal policies for control of queues with undiscounted costs", *Operations Research*, 37 (1989) 611–625.

[74] Suk J.B., Cassandras C.G., "Optimal control of a storage–retrieval queueing system", *Proc. of the 28th Conf. on Decision and Control*, Tampa, Florida, (1989) 1093–1098.

[75] Suk J.B., Cassandras C.G., "Optimal scheduling of two competing queues with blocking", *Proc. of the 27th Conf. on Decision and Control*, Austin, Texas, (1988) 1102–1107.

[76] Teha D.W., Pliska S.R., "Optimal control of single–server queueing networks and multi-class $M/G/1$ queues with feedback", *Operations Research*, 25 (1977) 248–258.

[77] Tijms H.C., *Stochastic modelling and analysis: a computational approach*, John Wiley & Sons, New York (1986).

[78] Turner L., Aoyama T., Pearson D., Anastasiou D., Minami T., (Editors), "Packet speech and video", *IEEE J. on Selected Areas in Communications*, 7 (1989).

[79] Varaiya P.P., Walrand J., Buyukkok C., "Extensions of the multiarmed Bandit problem: the discounted case", *IEEE Trans. on Automatic Control*, 30 (1985) 426–439.

[80] Viniotis I., "Optimal control of integrated communication systems via linear programming techniques", *PhD Thesis*, University of Maryland, College Park, MD (1988).

[81] Weber R.R., Stidham S., "Optimal control of service rates in networks of queues", *Advances in Applied Probability*, 19 (1987) 202–218.

[82] Wein L.M., "Scheduling networks of queues: heavy traffic analysis of a two station network with controllable inputs", *Operations Research*, 38 (1990) 1065-1078.

[83] Whittle P., "Multi armed bandits and Gittins index", *J. Royal Statistical Soc. B*, 42 (1980) 143-149.

[84] Whittle P., "Arm-acquiring bandits", *Annals of Probability*, 9 (1981) 284-292.

[85] Whittle P., *Optimization over time*, Vol. 1, John Wiley & Sons, New York (1982).

[86] Wu Z.J., Luh P.B., Chang S.C., Castanon D.A., "Optimal control of a queueing system with two interacting service stations and three classes of impatient tasks", *IEEE Trans. on Automatic Control*, 33 (1988) 42-49

[87] Yang P., Chen H., Yao D.D., "Optimal control and scheduling in a multi class queueing network: results and conjectures", *Proc. of the 29th Conf. on Decision and Control*, Honolulu, Hawaii, (1990) 582-586.

[88] Yao D.D., Schechner Z., "Decentralized control of service rates in a closed Jackson network", *IEEE Trans. on Automatic Control*, 34 (1989) 236-240.

[89] Yechiali U., "On optimal balking rules and toll charges in the *GI/M/1* queueing process", *Operations Research*, 19 (1971) 349-370

[90] Yechiali U., "Customers optimal joining rules for the *GI/M/s* queue", *Management Science*, 18 (1972) 434-443.