Università degli studi di Padova

Facoltà di Ingegneria

Tesi di Laurea in

Ingegneria dell'Automazione

Optimal control techniques for virtual memory management

Relatore

Prof. Gianfranco Bilardi

Candidato

Stefano Ermon

Anno Accademico 2007/2008

Life can only be understood backwards,
but it must be lived forwards.

Søren Kierkegaard

Abstract

Memory management performance is critical to any modern computing system. The page replacement problem has been extensively studied, but little attention has been given in literature to the *page I/O problem*, namely the problem of scheduling the I/O operations needed to manage page faults. The aim of the present work is to use optimal control techniques to find page I/O management policies that minimize the fault service time, that is the amount of time that passes from when a page fault arises to when the referenced page becomes available.

To this end, we make use of the simplified dynamical system model of a VMM system proposed in [3]. In this context, some novel statistical models for the disk access times are developed, witch special focus on the average bandwidth obtained by common disk scheduling algorithms as a function of the size of the I/O queue.

Given a formulation of the problem into the optimal control framework, firstly a relaxed problem is completely solved obtaining an useful lower bound to the optimal fault service time. Moreover some interesting properties regarding the optimal control policy are finally proved, the most important one regarding the optimal closed form control law for load operations.

Contents

Chapter 1

Introduction

A virtual memory system involves the separation of logical memory as perceived by users from physical memory. This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available, through a combination of hardware and software components that allow to map a large logical address space onto a smaller physical memory.

Virtual memory is commonly implemented by paging, that involves breaking physical memory into fixed-sized blocks called *frames* and dividing logical memory into blocks of the same size called *pages*.

Since virtual memory is usually larger than the available physical memory, only a subset of the virtual pages are mapped onto physical memory, while the others are stored on secondary memory (usually a disk).

If a process tries to use an unmapped page, it causes the CPU to trap the operating system and, in that case, a *page fault* is said to occur. To solve the problem, the Operating System has to fetch the page just referenced into a free page frame before restarting the trapped instruction. In order to maintain a pool of free frames (or to free one if the pool is empty), it also has to pick a page frame according to a *page replacement policy* to be evicted by writing its contents back to the disk. We define the time elapsing from when

a fault arises to when the referenced page becomes available in memory *fault service time*, that is obviously equal to the amount of time that the process has to wait before being restarted.

When the operating system decides to write a page to the disk or to fetch an unmapped page, it has to forward these I/O requests to the *disk subsystem*. Following [3], we call *page I/O manager* the component responsible of scheduling these operations, namely responsible of deciding when to submit the requests to the disk.

While the aim of *page replacement policies* is to minimize the number of occurring faults by properly choosing the pages to be evicted, the problem we are facing in this work is to find and analyze page I/O policies that minimize the *fault service time*.

The basic policy that issues load requests as soon as the faults arise and the store requests as soon as the pages to be evicted are chosen, is not generally optimal. In fact, *fault service time* is influenced directly only by the time needed to load the unmapped page into memory, but, at least if there are free frames left, it does not directly depend on the store operations needed for the evictions. Moreover, if the system keeps a pool of free frames, the *page I/O manager* can, at least to a certain degree, decide the timing of the evictions. Therefore, while on the long term the number of stores must correspond to that of loads, on the short term it is convenient to give some priorities to loads over stores.

As reported in [3], another key issue of the problem is the fact that all the most common scheduling algorithms used to manage disk devices can reduce average disk access times when the number of pending I/O requests increases. As a consequence, a candidate I/O policy that issues store requests only when there is no load requests waiting, is not generally optimal. In fact with this choice it is possible that the resultant average size of the pool of I/O requests visible to the disk becomes too small, and hence the disk throughput does

not suffice to keep up with the page fault generation rate.

The present work is organized in the following way: in chapter 2 we will present the promising model of a virtual memory management (VMM) system developed in [3], that among the others features, permits to decouple the page I/O problem from that of the page replacement, under weak assumptions typically satisfied by real systems. In chapter 3 we will develop some novel statistical models for the disk access times, that represent the bottleneck of every virtual memory system. In particular, we will mainly focus our attention on the response in terms of bandwidth of the disk scheduling algorithms to the size of the pool of I/O requests, obtaining some interesting analytical results for the most common algorithms. In chapter 4 we will briefly describe some useful topics of the Optimal Control Theory, with particular reference to [1] and [2]. Moreover, we will show how to make use of them by presenting the optimal control framework developed by the authors of [3] for the page I/O management problem. Finally chapter 5 is dedicated to obtain some insights about the structure of the optimal control policy. Firstly, an useful reformulation of the cost function associated with the problem is presented, that leads to a stronger understanding of the problem. This result is used to entirely solve the relaxed problem in the ideal case of unlimited free frames, obtaining both the optimal policy and its resultant average *fault service time*, that represents an useful lower bound for the original problem. Regarding the latter unrelaxed problem, although we won't be able to show a closed form solution for the control law, we'll be able to prove mathematically which is the structure of the optimal load policy. The proof also provides some interesting insights about the store policy, that confirm the heuristic conjectures proposed in [3]. Moreover, some regularity properties regarding the infinite control horizon will be proved.

Chapter 2

VMM as a dynamical system

2.1 Structure of a VMM system

With the purpose of designing an optimal page I/O policy, we consider the simplified model of a VMM system proposed in [3]. Although this model constitutes a greatly simplified picture of any real operating system, it captures the essential features needed to study the page I/O problem. The key aspects of this model are a *workload*, a *paged memory*, a *fault queue*, a *disk subsystem* and a *page I/O manager*; for the clearness of exposition we provide a brief description of each of these items.

The *workload* is viewed as a sequence of events called *page faults*, that arise when a program accesses a page that is mapped in the virtual address space, but not loaded into the physical memory. When a *page fault* occurs, the operating system tries to handle it by allocating physical memory space to the corresponding page and, if it is needed, by loading its content from the external storage.
In a real system, the number and the timings of such faults is determined by many factors such as the nature of the processes in execution, the CPU

scheduling policy adopted by the Operating System and, of course, by how virtual memory is managed. To be more precise, it depends heavily on the page replacement policy used, but also on the page I/O policy adopted, that is the object of our analysis. All these elements are interacting in a very complex manner, hence for the sake of mathematical tractability we will model the fault process as statistically independent of any other event in the system.

As we have seen so far, the model refers to a paged memory, that can be modeled as a set of M page frames, whose size is equal to that of a virtual page. If supported by the architecture, operating systems can change the page size; default values for operating systems like Windows Xp and Linux are 4096 bytes for the x86 platform. Frames are categorized into 4 types:

- *Active frames.* Frames to which a virtual page is assigned and the state of such page is correctly reflected in the frame.
- *Free frames.* Frames to which no virtual page is assigned.
- *Load frames.* Frames to which a virtual page has been assigned but the page state has yet to be loaded from the disk.
- *Store frames.* Frames to which a virtual page has been previously assigned, then evicted and its state has yet to be stored to the disk.

When a *page fault* arises, it has to wait a certain amount of time before a free frame is assigned to the desired page that was not loaded in physical memory. During this period, the fault is kept by the operating system in a *fault queue*. This queue is generally managed with a FIFO policy, but the policy adopted won't affect our analysis.

While studying the page I/O problem it is clearly of paramount importance to model the external storage used to store programs and data that

are currently inactive and not held in physical RAM. In fact external storage access times, that are much higher than those of main memory, represent the bottleneck for every Virtual Memory Management system. In the present work we will take into account the case of a disk drive, that is the most common option. We will model this part as a *disk subsystem*, whose aim is to model both the disk device and the data structures used by the operating system to interact with the disk, such as the queue of pending requests maintained by the OS. This queue can be managed with different policies, such as SSTF, SCAN, CSCAN, algorithms that try to find a compromise between optimizing disk access time and maintaining a certain fairness. Often requests of the same type (load or store operations) that operate on contiguous areas of the disk device are put together into one single request.

The duty of the *page I/O manager* is to decide

- when to assign a free frame to a waiting fault
- when to evict an active frame

When a free frame is assigned to a waiting fault, the *page I/O manager* issues a load operation to the *disk subsystem*, to transfer the corresponding page into the physical memory. On the contrary, when an active frame is evicted, it is responsibility of the *page I/O manager* to save its state to the disk with a proper store operation. It's important to stress the fact that the *page I/O manager* decides only when and how many pages to load or store but not which ones. In this way the page I/O problem is decoupled from that of the page replacement, achieving a great simplification. In fact the performance of the page replacement policy affects the VMM system described only in an indirect way, by influencing the *workload* and hence the fault process.

2.2 A dynamic system model

In this section we will give a brief description of the VMM dynamic model developed in [3]. In particular the authors model the VMM system as a discrete time dynamical system, where the state of the system describes the state of the *fault queue* and of the memory subsystem, by counting the number of the various types of frames described in section 2.1 at a certain time. The model proposed has no outputs and a certain number of inputs, some of which are adjustable by the *page I/O manager*, while the so called disturbances are not.
The time scale at which events related to the virtual memory management occur in real computing system is very small because it is imposed by the CPU cycle time. In fact faults can arise whenever a memory access is requested by a process, hence with an extremely high frequency. However from the page I/O management point of view it is unpractical to control the system with such a high sampling frequency because it would cause an extremely high management computation overhead. Therefore it is convenient to model an undersampled version of the VMM system with a sample time τ equal to the time interval between two consecutive calls of the control algorithm (namely the *page I/O manager*). Hence the value of τ should be considered a design parameter that represents a compromise between the readiness of the controller and the computational overhead it causes. Considered that the controller is usually invoked by Operating Systems either when a fault arise or a disk operation is completed, a convenient value of τ should be that of the average time required for a disk operation. Therefore, for modern disk drives, it should be of the order of milliseconds.

The nonlinear dynamic model proposed in [3] is described in space state

form by the following transition equation

$$x_{k+1} = f(x_k, u_k, w_k) \tag{2.1}$$

where x_k is the state of the system at step k (namely at time $k\tau$), u_k is the set of controllable inputs while w_k represents the disturbances or uncontrollable inputs. As we have already said, this is an undersampled version of the real VMM system, therefore many events and actions could happen in a time step. The transition function should take into account this fact and model the cumulative effect of all the events occurred.

The state x_k is a four dimensional vector of non-negative integers $x_k = (x_k^f, x_k^q, x_k^\ell, x_k^s) \in \mathbb{N}^4$. In particular x^f represents the number of free frames, x^q the number of faults in the fault queue, x^ℓ the number of load frames while x^s stands for the number of store frames. It is possible to define a set of admissible states S as follows:

$$S = \{(x^f, x^q, x^\ell, x^s) |\ 0 \leq x^f \leq x^F, 0 \leq x^q \leq x^Q, 0 \leq x^\ell \leq x^L, 0 \leq x^s \leq x^S, \\ x^f + x^s \leq x^F, x^s + x^\ell \leq x^D\}. \tag{2.2}$$

The upper bound set on the values of the state variables, that is imposed by noticing that computer systems work with a finite arithmetic, will sometimes be relaxed by setting it to ∞ for reasons of mathematical tractability.
The bound $x^\ell + x^s \leq x^D$ is set to model the fact that the number of pending requests in the disk queue can be limited in practical settings. The constraint $x^f + x^s \leq x^F$ means that at any time the number of frames that are free of will become free (because they have been chosen to be evicted) cannot exceed x^F. This bound is particularly important, because it is needed to properly decouple the page replacement problem from that of the page I/O. In fact in this model we are assuming a fault generating process independent from any

other event in the system. However in a real system increasing the number of free frames reduces the number of active frames and hence usually results in an increased fault rate. Therefore it is crucial that x^F is small (below 0.1 % of the total number of frames), otherwise the instantaneous value of x^f would be too correlated with the fault rate, resulting in an unrealistic independence assumption.
In order to properly decouple the Page Replacement problem from the Page I/O problem, we can assume that the fault rate ϕ depends on X_F in the terms of a function $g(.)$ as follows:

$$\phi = g(X_F),$$

where the exact form of $g()$ is hard to determine and heavily depends on the Page Replacement Policy adopted and on the workload being considered. Even if we don't know it precisely, it's reasonable that

$$\frac{\partial g}{\partial X_F} \geq 0,$$

that describes the fact that higher X_F means fewer usable pages and thus an increased fault rate. In any case, let us consider the average service time T_{fst} of a fault. We have

$$\begin{gathered} T_{fst} = f(\phi, X_F) \\ \tfrac{\partial f}{\partial X_F} \leq 0 \\ \tfrac{\partial f}{\partial \phi} \geq 0, \end{gathered}$$

where $f(\cdot)$ depends on the page I/O policy adopted. Therefore we can study the problem considering the parameters ϕ and X_F as independent, being aware that at least in principle it's possible to obtain the optimal X_F by minimizing $f(g(X_F), X_F)$.

The controllable inputs of the VMM system are those that can be decided by the *page I/O manager*, and in this model they are represented as

a two-dimensional vector of non-negative integers (u_k^ℓ, u_k^s). The value of u_k^ℓ represents the number of faults that are assigned a free frame by the *page I/O manager* at step k. The state of these pages that have been assigned a free frame has to be loaded from the disk, hence until this operation is done they are classified as load frames. On the other hand, u_k^s stands for the number of pages to be evicted at step k, whose state has to be stored in the secondary memory (the disk). In this case a certain number of active frames will become store frames, at least until the store operation is completed by the disk.
Given a certain state x, the controllable inputs that are admissible for x are defined as follows:

$$U(x) = \{(u^\ell, u^s) \in \mathbb{N}^2 \mid 0 \leq u^\ell \leq u^L(x), 0 \leq u^s \leq u^S(x, u^\ell)\},$$

where

$$u^L(x) = \min(x^f, x^q, x^D - x^\ell - x^s) \ ,$$
$$u^S(x, u^\ell) = \min(x^F - x^f - x^s, x^D - x^\ell - x^s - u^\ell).$$

The upper limit $u^L(x)$ models the fact that the number of faults that are assigned a free frame cannot exceed nor the number of faults in queue nor the number of free pages. Moreover it guarantees that the length of the disk queue never exceeds its maximum value x^D. The constraint imposed on $u^s(x)$ through u^S is needed again to limit the length of the disk queue and to make sure that the number of frames that are free of will become free cannot exceed x^F.

The values of a certain number of inputs cannot be controlled by the *page I/O manager*. The so called *disturbance vector* is a three dimensional vector of non-negative integers $(w_k^\phi, w_k^\ell, w_k^s) \in \mathbb{N}^3$. The value of w_k^ϕ stands for the number of faults that arise in the in the time interval $[k\tau, (k+1)\tau]$, where τ is the length of the time step. Furthermore, the values of w_k^ℓ and w_k^s represents respectively the number of loads and store requests completed by

the disk in the same time interval as before. The set of admissible values of the disturbance vector depends both on the current state of the system and on the controllable inputs applied by the *page I/O manager*. This situation can be explained by thinking that the *page I/O manager* issues its commands at the very beginning of a time interval, while the disturbances take place during all the interval and therefore depend on what the controller decided at the beginning.

The set of admissible disturbances is defined as follows:

$$D(x,u) = \{(w_k^\phi, w_k^\ell, w_k^s) \in \mathbb{N}^3 \mid 0 \leq w_k^\phi, 0 \leq w_k^\ell \leq x^\ell + u^\ell, 0 \leq w_k^s \leq x^s + u^s\} \tag{2.3}$$

While no upper bound is set on the maximum number of arrivals per step, the limits imposed on w_k^ℓ and w_k^s simply reflect the fact that the number of services completed by the disk cannot exceed the number of requests present in its queue. The transition function resulting from the considerations made is

$$\begin{aligned} x_{k+1}^f &= x_k^f - u_k^\ell + w_k^s \\ x_{k+1}^q &= x_k^q - u_k^\ell + w_k^\phi \\ x_{k+1}^\ell &= x_k^\ell + u_k^\ell - w_k^\ell \\ x_{k+1}^s &= x_k^s + u_k^s - w_k^s \end{aligned}$$

or in matrix form

$$x_{k+1} = f(x_k, u_k, w_k) = Ax_k + B \begin{bmatrix} u_k \\ w_k \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix} x_{k+1}^f \\ x_{k+1}^q \\ x_{k+1}^\ell \\ x_{k+1}^s \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x_k^f \\ x_k^q \\ x_k^\ell \\ x_k^s \end{bmatrix} + \underbrace{\begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix}}_{B} \begin{bmatrix} u_k^\ell \\ u_k^s \\ w_k^\ell \\ w_k^s \\ w_k^\phi \end{bmatrix}.$$

The first equation reflects the fact that the number of free frames is decremented by the load control u^ℓ (because they are assigned a faulting page), and so is the number of faults in queue x_k^q. The number of free frames x^f is incremented whenever the disk completes a store operation, while x^q is incremented when new faults arise and they are consequently enqueued.
As far as x^s and x^ℓ are concerned, they are incremented when the *page I/O manager* issues new operations (load or store, respectively through u^ℓ and u^s) and decremented as soon as the disk completes them (reflected by the values of w^ℓ and w^s).
If we assume that there is no limit on the number of faults in the fault queue ($x^Q = \infty$), then it's easy to see that given an admissible state x_k, if $u_k \in U(x)$ and $w_k \in D(x, u_k)$ then $x_{k+1} = f(x_k) = Ax_k + Bu_k$ is also admissible.

Regarding the modeling of the fault process w_k^ϕ, we have already seen that it depends in a complex way from the interaction between many components of the operating system and thus is extremely difficult to capture its true nature. Therefore, for reasons of mathematical tractability, w_k^ϕ is assumed as an independent identically distributed stochastic process. Moreover it is assumed independent from any other event of the VMM system. Clearly this is not true for real systems: as an example is it clear that it depends on the number of free frames, in fact an high number of free frames reduces that of active frames and hence it increases the faulting probability.
Furthermore page faults also depend on each other: if a memory access of a process raises a page fault, then the process must wait it to be served before continuing its execution, hence it cannot trigger any more faults.
A common choice in Queue Theory to model arrival processes like w_k^ϕ is to assume a Poisson distribution, with an average arrival rate per step ϕ. In our case, ϕ represents the number of faults that arise on average in a time step of length τ. The resulting probability distribution is

$$P[w_k^\phi = h] = \frac{e^{-\phi}\phi^h}{h!} \quad \forall \;\; h \geq 0, \; \forall \;\; k, \tag{2.5}$$

that represents the probability of exactly h faults arisen in the time interval $[k\tau, (k+1)\tau]$.

As far as w^ℓ and w^s are concerned, it is worth to point out that disk access times represents the bottleneck for every Virtual Memory Management system. In fact the secondary memory is by definition slower than primary memory, but in the case of a disk drive its average access times (of the order of milliseconds) are much higher than those of the primary memory (of the order of nanoseconds). Therefore it is of primary importance to understand its behavior in detail, also because the *disk subsystem* we are considering can be managed in very different ways from the operating system. In fact the I/O request queue is usually maintained with policies that rearrange the items in order to reduce the service times, while granting at the same time a certain fairness between them. In this way a more timely service can be delivered when there are many requests pending in queue. A separate chapter will be devoted to understand how much this increased throughput can be, considering some common queuing policies like SSTF and CSCAN. Moreover the analysis will capture how physical parameters of the disk device, such as the spinning velocity or the number of tracks, affects its bandwidth, according to the models we will develop.

While building a statistical model for disk access time, it's important to note that the time needed to service a certain I/O request depends both on the physical address (on the disk device) of such request and on the current state of the disk, namely the state of the disk head. Since in the model we are using we are not considering the individual identity of each page, we don't know their physical address either. Since adding this information to the model would be too complicated, we will construct an averaged statistical model for disk service times.

Although in chapter 3 some rather complex analytical models for the *disk*

subsystem will be derived, it is useful to describe the so called $b(y)$-DS model, introduced by the authors of [3], that will be often used in the following chapters, mostly for its simplicity. The in depth analysis developed in chapter 3 will also be useful to understand when the $b(y)$-DS model is appropriate and, in that case, how to set its parameters properly.

Whichever is the *disk subsystem* model used, we will always model w_k^ℓ and w_k^s as random variables affected solely by the numbers $y_k^\ell = x_k^\ell + u_k^\ell$ and $y_k^s = x_k^s + u_k^s$, that is the number of load and store requests visible to the disk during the time interval $[k\tau, (k+1)\tau]$. By further assuming stationarity, the *disk subsystem* model can be described by the probability distribution

$$P[w_k^\ell = \ell, w_k^s = s | y_k^\ell = i, y_k^s = j], \tag{2.6}$$

where for the definition of admissible disturbances 2.3,

$$P[w_k^\ell = \ell, w_k^s = s | y_k^\ell = i, y_k^s = j] = 0$$

if $\ell > i$ or $s > j$, because the disk cannot service more request than those effectively present in queue.
The $b(y)$-DS disk model proposed in [3] describes the probability 2.6 in terms of a bandwidth function $b(y) : \mathbb{N} \to [0, 1]$ with the following properties:

- B0: b(0)=0
- B1: $\lim_{y\to\infty} b(y) = 1$
- B2: if $z > y$ then $b(z) \geq b(y)$
- B3: if $z > y$ then $b(z)/z \leq b(y)/y$

In a $b(y)$ *disk subsystem*, the joint probability distribution 2.6 is such that

$$E[w^\ell | y^\ell = i, y^s = j] = b(i+j)i/(i+j),$$
$$E[w^s | y^\ell = i, y^s = j] = b(i+j)j/(i+j).$$

In particular in a Bernoulli $b(y)$-DS model the probability 2.6 is set in the following way:

$$P[w^\ell = 1, w^s = 0 | y^\ell = i, y^s = j] = b(i+j)i/(i+j),$$
$$P[w^\ell = 0, w^s = 1 | y^\ell = i, y^s = j] = b(i+j)j/(i+j),$$
$$P[w^\ell = 0, w^s = 0 | y^\ell = i, y^s = j] = 1 - b(i+j).$$

In a $b(y)$-DS model, if there are y pending request, $b(y)$ are completed on average in one step, hence $b(y)$ represents the average disk bandwidth, measured in requests per step. Property B0 implies that no requests are serviced if there is none, while it is assumed that load and store operations are equally likely to be served. Property B2 formalizes the fact that the disk bandwidth increases with the number of requests to be serviced, however property B3 ensures that this growth rate is always sub-linear.
In the particular Bernoulli case, it is assumed that no more than one request can be serviced in one step, hence the step time τ has to be properly chosen, otherwise the Bernoulli model is not appropriate. The Bernoulli model will be adopted for the analysis developed in chapter 5, mostly for its simplicity.

It is important to note that with a $b(y)$-DS *disk subsystem* model, the overall VMM model looses the linearity property because of the non-linear state feedback introduced by the disk-subsystem. Hence if we want to model it as a dynamical system with only three inputs (u_k^ℓ,u_k^s,w_k^ϕ), namely the ones that are really external to the system, then we have to cope with its inherent non-linearity.

Chapter 3

The disk I/O subsystem

As we have seen in the previous chapter, disk latencies plays a key role in determining the performance of any virtual memory system, therefore the *disk subsystem* model is likely to have a strong impact on the structure of the optimal *page I/O manager*.
If we look at the problem from the perspective of designing a *page I/O manager*, the key issue does not seem to give accurate estimates of disk access times, but rather to understand how much they are influenced by the number of requests visible to the disk. In fact disk access times have surely a great impact on the overall performance of the system, but the *page I/O manager* has no control over them. Instead it can choose the timings of the I/O operations to take advantage of the bandwidth increment, that consequently seems the most important aspect to be modeled.
Given a generic stationary *disk subsystem* model, that can be described by the joint probability distribution 2.6, it is always possible to define a bandwidth function

$$E[w^\ell + w^s | y^\ell = i, y^s = j] = b'(i, j).$$

In most cases there's no reason to consider asymmetries between load and store operations, hence $b'(i, j) = b(i + j)$.

Therefore the aim of this chapter is to understand in depth what are plausible growth rates of $b(\cdot)$ and, at the same time, how do physical properties of the disks such as the number of tracks and rotational speed affect it. In this way we will also be able to obtain some insights about the choice of $b(y)$-DS models described in chapter 2.

Once these properties are well understood, it will be much more easy to tune the *page I/O manager* for a certain system configuration: for example a system administrator could simply set a number of of important parameters (like the number of tracks, the disk's rotational speed) and the controller will be able to adjust itself to perform in a semi-optimal way in that setting. This goal also fits perfectly with the autonomic computing vision ([6]), whose aim is to split hugely complex systems into a number of interacting, autonomous and self-governing components.
Consequently, the aim of this chapter is to investigate how the bandwidth of the *disk subsystem* is related to the amount of workload it is subjected to.
In particular we suppose that the Operating System maintains a queue of requests (both load and store operations) for a certain disk, managed with a scheduling policy. Those requests, passed to the physical device in an order depending on the policy in use, are served by the disk within a certain amount of time. Our goal is to study how the length k of the queue affects the performance of the systems in terms of mean service time $E[T(k)]$. The bandwidth function $b(\cdot)$, that represents the average number of request serviced by the disk in an interval of length τ, can therefore be estimated as

$$b(k) = \tau \frac{1}{E[T(k)]}.$$

In particular we will consider both C-SCAN and SSTF policies in our models and we'll be able to solve them analytically proving some useful statistical results.

To estimate the average service time, we assume that while the disk is working, it spins with a fixed angular speed. To perform any kind of operation on a certain sector, the disk's head has to move onto the target sector in the track it belongs to. The time required for the head to reach the right track is usually called *seek time*.
Once the head has reached the track, it has to wait until the desired sector passes beneath it: only at this time the disk controller proceeds issuing the desired read or write operation. The time spent waiting for the disk to rotate is referred as *rotational delay*. Summing the rotational delay with the seek time we obtain a good estimate of the service time, usually called *access time*.
We begin defining the physical parameters that describe all of our *disk subsystem* models. It's important to notice that what we call a *disk subsystem* is not an existing physical device: in fact it models components that reside both in the physical disk and in the Operating system (such as the queue of pending requests). The following parameters will be common to all our models:

$$\begin{aligned} N &= \text{number of tracks;} \\ k &= \text{number of pending requests;} \\ \omega &= \text{rotational speed of the disk;} \\ N_{sec} &= \text{number of sectors per track.} \end{aligned}$$

Each one of the k requests is represented in our model as a couple (track, sector), stochastic and therefore described as a random vector (t, s). Moreover we assume that t is statically independent from s, which means that the desired sector is independent from the desired track and vice versa.
Furthermore we assume that each request is independent from the others and uniformly distributed on the entire disk, hence t and s are uniformly distributed random variables in $[1, N]$ and $[1, N_{sec}]$ respectively.

Considered the extremely high sector-per-track density in modern devices, we will in some cases assume a continue rather than discrete probability distribution for s. In that case, s represents the desired angular position onto a track t rather than a certain sector, hence it should be modeled as a uniformly distributed random variable in the interval $[0, 2\pi]$. For reasons of mathematical tractability, we assume that the various request are statistically independent from each other. This assumption is rather unrealistic but provides mathematical tractability to the problem. In practice, however, it is likely that requests issued at the same time (or in close succession) refer to logically related sectors of the disk (for example, where the Operating System keeps the various part of a certain file).
In the next sections of the present chapter a *disk subsystem* model will be developed with respect to the most common scheduling policies used by modern Operating Systems.

3.1 Modified C-SCAN scheduling policy

The first case considered is that of an Operating System maintaining the I/O request queue according to a variant of the CSCAN (Cyclic Scan) scheduling algorithm([12]), which is in turn a modified version of the elevator algorithm. The elevator algorithm, known by this name both in the disk world and the elevator world, requires the software to maintain one direction bit. When a request is serviced, the algorithm check the bit: if it is UP, the arm is moved to the next higher numbered track with a pending request. If no request are pending at higher positions, the direction bit is reversed. When the bit is set to DOWN, the move is to the next lower numbered track with a pending request, if any.
The Cyclic Scan variant works in a very similar way, with the difference that when the highest numbered track with a pending request has been serviced,

the arm goes to the lowest-numbered track with a pending request and then continues moving in an upward direction (sometimes this algorithm is also called CLOOK).

In the first model we develop, we consider a variant of the previously described scheme where the arm keeps scanning the tracks always in the same direction until the last track is visited, and then comes back to the first one before repeating the scanning (when the other is called CLOOK, this algorithm is usually referred as the proper CSCAN). By further assuming that moving from the last track to the first one takes no time, the motion of the disk head takes a modular form. In fact the resulting motion resembles that of the hand of a N hours clock, whose inherent symmetry greatly simplifies the analysis.

Regarding the *seek time*, we assume that the time needed by the head to move by exactly m tracks is directly proportional to m according to a constant A. This assumption is certainly realistic for m large enough, namely in the case in which the head is in motion at constant speed for the most part of its path. Unfortunately, when m is small the assumption we made does not hold, because the resulting motion is largely uniformly accelerated, hence the *seek time* results proportional to the square root of m. For even smaller m the seek time is dominated by the so called *settle time*, in which the drive controller sets the system to perform the desired task.

Therefore a more realistic function to model the time needed to move by m tracks, that takes into account also the accelerated motion phase, is

$$l(m) = \begin{cases} a\sqrt{m} + s & \text{if } 1 < m < d \\ Am + s & \text{otherwise} \end{cases} \tag{3.1}$$

In this case s represents a constant *settle time*, while d reflects the fact that for small distances the motion is largely uniformly accelerated, while on longer distances the constant speed phase prevails.

The values of a, A, d and s can usually be deducted from the drive data-sheet

or by experimental analysis with good precision. As an example, the value of A can be obtained by dividing the so called *full-stroke time* (time needed to move by N tracks) by N.

In this section we study the steady state behavior of the disk, where we suppose that whenever a request is serviced by the disk, another one immediately enters the queue, thus fixing the queue length at k elements. Moreover we suppose that the newly entered request has the same uniform probability distribution and is independent from the other ones. Our objective is to find an analytical expression for the average service time in this steady state condition. We define a random variable T_r that represents the time spent in queue by a request, counted from when it enters the queue till it is serviced by the disk.

When a new request enters the queue, it will be p tracks distant from the current position of the head, where $0 \leq p \leq N-1$. Moreover, considered the statistical independence between the requests in queue and the last one entered, we can affirm that p is uniformly distributed. Therefore on average the head must cover $\frac{N-1}{2}$ tracks to reach it, taking an average time of $A\,\frac{N-1}{2}$. If we disregard the rotational delays, this time equals the average time spent in queue by a request, that is

$$E[T_r] = A\,\frac{N-1}{2}\ .$$

According to the assumptions made, k elements are present in the queue at any time, each one with an average life length equal to $E[T_r] = A\,\frac{N-1}{2}$. Therefore the overall average frequency at which new requests are enqueued f_λ can be computed from the following formula:

$$f_\lambda = \frac{2k}{A(N-1)}\ .$$

In fact this also represents the overall average frequency at which requests are serviced, that is clearly equal to $\frac{k}{E[T_r]}$.

The average arrival frequency of new request f_λ is also equal to the reciprocal of the average time elapsed between two consecutive services performed by the disk, which we call T. Therefore

$$E[T(k)] = \frac{1}{f_\lambda} = \frac{A(N-1)}{2k} \ .$$

If we take into account the rotational delays, we have to specify in detail how the scheduling policy we are modeling handles requests that belong to the same track, and in particular in which order they are serviced. In our model we assume that, in case of multiple request on the same track, the system privileges the one that has been in the queue for the longest time (thus it is a FIFO like policy).

In this case the average time that elapses between two consecutive services can be easily computed by summing the time found in the previous case (when the rotational delay is set to zero) and the average rotational delay. Again, considered the independence between the requests in queue, the angular distance between the head and the sector to be read on the newly reached track can be considered a random variable uniformly distributed in $[0, 2\pi]$. Therefore the average distance is π radians, that implies an average rotational delay of $\frac{\pi}{\omega}$. With the preceding definition of $T(k)$,

$$E[T(k)] = \frac{1}{f_\lambda} = \frac{A(n-1)}{2k} + \frac{\pi}{\omega} \ .$$

Therefore the average frequency f_λ at which requests are serviced is

$$f_\lambda = \frac{1}{E[T(k)]} = \frac{2k\omega}{\pi 2k + A(n-1)\omega} \ .$$

It is easy to see that in this model the average service time $E[T(k)]$ is a decreasing function of k, but there is a lower bound imposed by the rotational delay that, with the assumptions made, cannot be reduced by increasing k.

3.2 C-SCAN scheduling policy

In the present section we consider a disk I/O queue maintained with the unmodified Cyclic Scan scheduling policy, as described in the previous section. Again we consider a set of k request, independent and uniformly distributed on the entire disk. This time we are interested in studying the average time needed to serve all of them, that divided by k provides a good estimate of the average service time.

First of all, we define N_t as the number of tracks with a pending request. The expected value of N_t can be computed with the following formula

$$E[N_t(N,k)] = NP[\text{track is requested}],$$

where $P[\text{track is requested}]$ is the probability that a certain track has a pending request. Obviously

$$P[\text{track is requested}] = 1 - P[\text{track is not requested}]$$

where

$$P[\text{track is not requested}] = \left(\frac{N-1}{N}\right)^k,$$

that means that all the k requests fall into the other $N-1$ tracks. Therefore

$$E[N_t(N,k)] = N\left(1 - \left(\frac{N-1}{N}\right)^k\right).$$

If there are N_t tracks with pending requests, then whichever is the initial position of the head, the maximum seek length (measured in tracks) is $N - N_t + 1$. In fact the maximum track gap between two consecutive tracks with pending request is $N - N_t$.

In this context, we are interested in determining the average seek time spent to serve all the k requests pending, that we call $c_s(k)$. To that end, we begin determining the expected number of seeks that occur at each distance. Then we multiply this number for the seek cost of that distance, that we call $l(j)$.

In the case of seek time linearly dependent from the number of tracks covered, we have

$$l(j) = Aj.$$

However a more realistic function, like 3.1, can be adopted without any problem.

Whichever is the definition $l(j)$, the value of $c_s(k)$ can be computed this way

$$c_s(k) = \sum_{j=1}^{N-N_t+1} s(j) \cdot l(j), \tag{3.2}$$

where $s(j)$ is the expected number of seeks of length j. If we assume that the head begins its job onto the lowest numbered track, then the expected number of seeks of length j can be deduced from the the probability p_j that a seek has length j, since we already know that exactly $N_t - 1$ seeks will be made, hence

$$s(j) = (N_t - 1)p_j.$$

Let's consider a pair of successive track with pending requests that are j tracks apart. This pair can be found in $N - j$ positions. For any pair, there are

$$\binom{N-j-1}{N_t-2}$$

positions valid for the other $N_t - 2$ selected tracks. Therefore the number of possible ways of picking two consecutive tracks with a pending request that are j tracks apart is

$$\binom{N-j-1}{N_t-2}(N-j).$$

Furthermore, the number of ways of picking two consecutive tracks with a pending request is

$$(N_t - 1)\binom{N}{N_t},$$

hence the probability p_j can be calculated this way

$$p_j = \frac{\binom{N-j-1}{N_t-2}(N-j)}{(N_t-1)\binom{N}{N_t}}.$$

Therefore

$$s(j) = \frac{\binom{N-j-1}{N_t-2}(N-j)}{\binom{N}{N_t}}$$

if $N_t \geq 2$, $N \geq N_t$ and $N - j - 1 \geq 0$.

Finally, by substitution into 3.2,

$$c_s(k) = \sum_{j=1}^{N-N_t+1} s(j) \cdot l(j) = \sum_{j=1}^{N-N_t+1} \frac{\binom{N-j-1}{N_t-2}(N-j)}{\binom{N}{N_t}} \cdot l(j).$$

Clearly, the formula just found does not take into account the time needed to go from highest numbered requested track to the lowest-numbered requested track, simply because we are supposing that the head begins its job onto the lowest-numbered track with a pending request.

If we want to consider the general situation, we need to estimate the average time needed to go from the highest numbered requested track to the lowest-numbered requested track, that we call $r(k)$.

To this end, we consider k discrete random variables $t_i,\ i = 1, \ldots, k$ uniformly distributed in $[1, N]$, that represent the numbers of the tracks requested. We need to evaluate

$$p_{mm}(k, j) = P[\ \max\{t_1, \ldots, t_k\} - \min\{t_1, \ldots, t_k\} = j].$$

To evaluate $p_{mm}(k, j)$, we note that there are

$$(N - j + 1)$$

possible intervals made of j consecutive tracks.

The probability that all the k requests fall into an interval of j tracks is

$$\left(\frac{j}{N}\right)^k.$$

Given that k requests fall into it, the probability that there is at least one request onto the first track of the interval and at least one request onto the last track of the interval, is

$$\left(1 + \left(\frac{j-2}{j}\right)^k - 2\left(\frac{j-1}{j}\right)^k\right).$$

Therefore

$$P[\max\{t_1,\ldots,t_k\} - \min\{t_1,\ldots,t_k\} = j-1] = (N-j+1)\left(\frac{j}{N}\right)^k \left(1 + \left(\frac{j-2}{j}\right)^k - 2\left(\frac{j-1}{j}\right)^k\right),$$

hence

$$r(k) = \sum_{j=2}^{N} l(j-1)(N-j+1)\left(\frac{j}{N}\right)^k \left(1 + \left(\frac{j-2}{j}\right)^k - 2\left(\frac{j-1}{j}\right)^k\right).$$

Therefore a better estimation of the average seek time spent to serve all the k requests pending is

$$c_s(k) = r(k) + \sum_{j=1}^{N-N_t+1} \frac{\binom{N-j-1}{N_t-2}(N-j)}{\binom{N}{N_t}} \cdot l(j).$$

As far as the rotational delay is concerned, we consider the overall average rotational delay spent to service all the k pending requests, that we call $c_r(k)$. Since there are exactly N tracks

$$c_r(k) = N\beta(k),$$

where $\beta(k)$ is the average rotational delay per track.
Modern disk drive controllers based on technology such as SCSI can operate on a queue of requests simultaneously. Therefore, requests for the same track can be enqueued in the controller so that it can determine the best way of retrieving the appropriate sectors in order to reduce the overall rotational delay. Since the controller has detailed information on the head's position, it can arrange the requests in a way consistent with the order with which the desired sectors will pass underneath the head.
If there are j pending request onto a single track, on average they divide the track (that can be seen as a circle) into j equal parts, each one of $\frac{2\pi}{j}$ radians. When the head reaches the track after the seek phase it will find itself in the

middle of one of these parts.

Therefore the overall time needed to service them all is

$$\frac{2\pi - \frac{\pi}{j}}{\omega}$$

if the disk spins at fixed angular speed ω.

The probability of having exactly j request on a track results

$$P[j \text{ req. present}] = (\frac{1}{N})^j (\frac{N-1}{N})^{k-j} \binom{k}{j}.$$

Therefore we can determine $c_r(k)$ in the following way

$$c_r(k) = N \sum_{j=1}^{k} \frac{2\pi - \frac{\pi}{j}}{\omega} (\frac{1}{N})^j (\frac{N-1}{N})^{k-j} \binom{k}{j}.$$

At this point the total average time needed to service k requests is obviously

$$c_t(k) = c_s(k) + c_r(k).$$

It is also possible to obtain an estimate of the average service time by dividing the total time $c_t(k)$ by the number of requests k,

$$E[T(k)] = \frac{c_s(k) + c_r(k)}{k}.$$

As a confirmation of the results found, it is possible to see that, assuming a constant speed motion of the head (that is $l(j) = Aj$),

$$\lim_{k \to \infty} c_r(k) + c_s(k) = 2AN + N\frac{2\pi}{\omega},$$

that implies

$$\lim_{k \to \infty} E[T(k)] = 0.$$

In fact a fixed and limited amount of time is needed to explore the entire disk, hence within that time the device is able to service any number of requests.

3.3 SSTF scheduling policy

In this section we consider an I/O request queue maintained with a Shortest Service Time First policy, that is essentially a form of shortest-job-first (SJF) scheduling. With this scheduling algorithm, the next request to be serviced is chosen as the pending request closest to the current head position. This policy provides a substantial improvement in performance, but may cause starvation of some requests because it does not ensure any kind of fairness. When an Operating System uses this kind of algorithm, the definition of distance used is essentially related to the seek time. In fact it usually does not have any information about the angular position of the head, hence the scheduling algorithm cannot take into consideration the rotational delays. However modern disk drive controllers can operate on a queue of requests simultaneously, hence they can use a definition of distance that considers both the seek time needed to reach a certain track with a pending request and the rotational delay to be spent once the track has been reached. Therefore in this section we will consider an SSTF operated queue, where the service time is the actual access time of a request and not only the seek time component.

We start our analysis by supposing that the head begins its duty of serving k requests onto a generic track $\bar{x}$. The rotational delay of each of the k requests, namely the amount of time the head has to wait until the desired sector will pass underneath the head (once it has reached the destination track) is uniformly distributed in $[0, \frac{2\pi}{\omega}]$. In fact all the I/O requests are supposed to be independent and uniformly distributed onto the entire disk. We call rotational distance of a certain request the amount of time the head has to wait until the desired sector will pass underneath the head (once it has reached the destination track).

We are interested in determining the average rotational distance of the h-th closest request, where $h = 0$ refers to the closest one and $h = k - 1$ to the farthest(we are only considering the rotational delay and not the seek time needed to reach the track it belongs to).
To that end, we consider the probability that a gap of j sectors is present between the head and the request (notice that this time we are considering a track divided into sectors), and that exactly h other requests belong to this gap:

$$P_{gap}(j,k,h,N_{sec}) = \binom{k}{k-h}(\frac{N_{sec}-j}{N_{sec}})^{k-h}(\frac{j}{N_{sec}})^{h}(1-(\frac{N_{sec}-j-1}{N_{sec}-j})^{k-h}).$$

The average rotational distance of the h-th closest request, $r_v(q,h,N_{sec})$ can be calculated in the following way

$$r_v(k,h,N_{sec}) = \frac{2\pi}{N_{sec}} \sum_{j=1}^{N_{sec}-1} j \; P_{gap}(j,k,h,N_{sec}).$$

For $N_{sec} \to \infty$, the k requests tend to divide the track into k parts each one of $\frac{2\pi}{k}$, hence

$$\lim_{N_{sec}\to\infty} r_v(k,h,N_{sec}) = \frac{\pi(2h+1)}{k}.$$

To obtain an estimate of the average overall access time (that takes into account both the rotational delay and the seek time) we can use the following formula:

$$c(k) = \sum_{i=0}^{k-1}\sum_{l=0}^{N-1} c(i,l)P(i,l),$$

where $c(i,l)$ is the overall access time of the i-th request if it is on the track numbered l, while $P(i,l)$ represents the probability that the i-th request is chosen by the SSTF algorithm, again if it is on track l. The ordering we are considering is the one introduced before that is relative to the rotational distance.
In the case of seek time linearly dependent from the number of tracks covered,

we have

$$c(k) = \sum_{i=0}^{k-1} \sum_{l=0}^{N-1} (\frac{r_v(k,i,N_{sec})}{\omega} + Al)P(i,l).$$

It remains to find an expression for $P(i,l)$, namely the probability that the i-th closes request (from a rotational distance point of view) is exactly l tracks apart from the head and, at the same time, it is the one with the shortest service time, hence the one chosen by the queuing algorithm. To that end, let's consider two random variables x_f and x_j, discrete and uniformly distributed in $[1, N]$. It's easy to see that

$$P[\ |x_f - x_j| = l\ ] = \begin{cases} \frac{1}{N} & , \quad \text{if } l = 0 \\ \frac{2(N-l)}{N^2} & , \quad \text{if } l \neq 0 \end{cases}$$

Moreover

$$P[|x_f - x_j| > l] = \begin{cases} 1 & , \quad \text{if } l < 0 \\ 0 & , \quad \text{if } l > N-1 \\ \frac{2(N(N-l-1)+\frac{(N+l)(1+l-N)}{2})}{N^2} & , \quad \text{otherwise} \end{cases}$$

In fact the following equation holds

$$\sum_{k=l+1}^{N-1} P[|x_f - x_j| = k] = \sum_{k=l+1}^{N-1} \frac{2(N-k)}{N^2} = \frac{2(N(N-l-1)+\frac{(N+l)(1+l-N)}{2})}{N^2}.$$

Furthermore

$$\begin{aligned} P(i,l) &= P[i\text{-th is } l \text{ tracks apart}]P[\text{the others have greater access times}] \\ &= P[|x_f - x_j| = l] \prod_{s=0, s\neq i}^{k-1} P[s\text{-th request has a greater access time}] \end{aligned}$$

To estimate the last unknown probability, we try to render in terms of seek time the advantage (or disadvantage) of the s-th request in comparison with the i-th from the rotational distance point of view.

We define

$$n(s,i,l) = \lfloor \frac{\frac{r_v(k,i,N_{sec})}{\omega} + Al - \frac{r_v(k,s,N_{sec})}{\omega}}{A} \rfloor.$$

When $n(s,i,l) - l > 0$, request s is closer to the head than request i. In that case, $n(s,i,l)$ represents the number of tracks needed to amortize the advantage from the rotational distance point of view. In other words, it means that request s has a greater access time than request i only if it is at least $n(s,i,l)$ tracks apart from the head. Therefore

$$P(i,l) = P[|x_f - x_j| = l] \prod_{s=0, s\neq i}^{k-1} P[|x_f - x_j| > n(i,s,l)].$$

By substitution we obtain the following closed form expression for $c(k)$:

$$c(k) = \sum_{i=0}^{k-1} \sum_{l=0}^{N-1} (\frac{r_v(k,i,N_{sec})}{\omega} + Al) P[|x_f - x_j| = l] \prod_{s=0, s\neq i}^{k-1} P[|x_f - x_j| > n(i,s,l)].$$

It is possible to see that $c(k)$ is well approximated by $\frac{1}{\sqrt{k}}$, a fact that can be interpreted heuristically in the following way. If we imagine to throw k points at random on a square area in a plane, the average distance between them decreases as $\frac{1}{\sqrt{k}}$. Now, if we represent the k requests as points in a plane with coordinates representing respectively the seek and rotational distance, the $\frac{1}{\sqrt{k}}$ approximation does certainly make sense.

As we have already pointed out before, the estimates of the average service time just found in this section can be used to find suitable growth rates for the bandwidth function $b(\cdot)$, starting with physical parameters of the disk known at design time (or at least set by system administrators and hence known to the *page I/O manager*).

Chapter 4

Optimal control problem formulation

In this chapter we will firstly describe in brief some topics of the optimal control theory and then we will present the optimal control framework developed by the authors of [3] for the page I/O management problem.

Optimal control deals with the problem of finding a control law for a given system such that a certain optimality criterion is achieved. In particular, given a dynamical system, the objective is to minimize a certain cost function of the system evolution during a given time interval, a mathematical expression of what is considered an undesirable outcome.
Following the definition of the problem given in [2], we consider a stationary discrete time dynamic system

$$x_{k+1} = f(x_k, u_k, w_k) \qquad k = 0, 1, \dots \quad ,$$

where for all k, the state x_k belongs to the state space S, the control u_k is an element of a space C and w_k is an element of a space D. The control u_k is constrained to take values in a given non-empty set $U(x_k)$, that depends on the current state x_k. The disturbance w_k is modeled as a stochastic

process, in which each random variable is characterized by a probability distribution $P(\cdot, x_k, u_k)$ that might explicitly depend on x_k and u_k but not on prior disturbances $w_0, \ldots, w_{k-1}$. Moreover the random disturbances w_k are supposed to have identical statistics.

Given a time interval T, that can be either finite or infinite, an admissible control policy $\pi = \{\mu_k, k \in T\}$ is a sequence of functions $S \to C$ such that $\mu_k(x_k) \in U(x_k)$ for every $x_k \in S$, for each $k \in T$. Given such a control policy π, we say that the system is operated under the control policy π if it evolves according to the following equation:

$$x_{k+1} = f(x_k, \mu_k(x_k), w_k) \qquad k = 0, 1, \ldots \quad . \tag{4.1}$$

In other words, the controllable inputs u_k are chosen for each $k \in T$ as $u_k = \mu_k(x_k)$.

4.1 Finite horizon

In the case of a finite time interval $T = [0, 1, \ldots, N]$, the minimization problem arises from the definition of an instantaneous cost function $g_k(x_k, u_k, w_k)$ and of a terminal cost $g_N(x_N)$ incurred at the end of the time interval depending on final state reached by the system.

The cost function is additive, in the sense that $g_k(x_k, u_k, w_k)$, the cost incurred at time k, accumulates over time. Because of the presence of w_k, the state trajectory is in general stochastic, hence it is necessary to introduce an expectation for the minimization problem to be well-defined. Given an initial state x_0 and an admissible control policy π, equation 4.1 makes x_k and w_k random variables with well defined probability distributions. Therefore, given the instantaneous cost functions $g_k, k \in T$, the expected cost

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

is well defined.
For a given initial state x_0, an optimal control policy π^* is the one that minimizes the expected cost, or in other words

$$J_{\pi^*,N}(x_0) = \min_{\pi \in \Pi} J_{\pi,N}(x_0),$$

where Π is the set of all admissible policies and $J_{\pi^*,N}(\cdot)$ is the *optimal cost function*.

4.2 Infinite horizon

In the case of an infinite time interval $T = [0, 1, \dots,)$, the analysis is restricted to stationary systems, where the transition function, the instantaneous cost and the disturbance statistics are constant over the time.
The assumption of an infinite number of stages is never satisfied in practice, but it is a good approximation of situations in which the control horizon is extremely long. On the contrary, the assumption regarding the stationarity of the system is often satisfied in practice, or at least it represents a good approximation of systems described by with slowly changing parameters.
In the case of an infinite time interval, various classes of problems can be defined, based on the particular metric used to deal with the total cost over an infinite number of stages.
In this section we will focus our attention only to the optimization of the average cost per step starting from state i, which is defined by

$$J_\pi(i) = \limsup_{N \to \infty} \frac{1}{N} E[\sum_{k=0}^{N-1} g(x_k, \mu_k(x_k), w_k) | x_0 = i],$$

where π is an admissible control policy under which the system is operated. Again, the optimal control problem consists in finding the optimal policy π^*, that is the policy that minimizes the average cost per stage

$$\pi^* = \arg\min_{\pi \in \Pi} J_\pi(i).$$

The *optimal average cost per stage* is

$$J_{\pi^*}(i) = \min_{\pi \in \Pi} J_\pi(i),$$

that is in general dependent from the initial state i. However, in most problem of interest it results independent from the initial state. Moreover in most cases minimization of the average cost per stage can be achieved by a stationary policy, that is $\mu_k(x) = \mu(x)$ for each k, a fact that makes their implementation particularly simple.

4.3 Fault service time minimization

In [3] the authors show how the problem of minimizing *fault service time* in VMM can be cast as an optimal control problem. In particular, considered the dynamical description of the system given in chapter 2.2, they manage to find a suitable cost function. In this section we will briefly summarize their findings.

Even if the individual history of each fault is not modeled in the system, it is possible to see that, from the time it is generated until it is serviced, each fault is either in the fault queue (contributing to x^q) or in the *disk subsystem* queue (contributing to x^ℓ).

Hence a good definition for the *instantaneous cost* is

$$g(x, u, w) = g(x) = x^\ell + x^q,$$

which does not depend on time (hence it is stationary and can be consequently used in an average cost per step minimization problem), on the control and on the disturbance and is only affected by the state of the system. By setting the *terminal cost* to zero, the cost function for a finite control horizon becomes:

$$J_{\pi,N}(i) = E[\sum_{k=0}^{N-1} x_k^\ell + x_k^q \ | x_0 = i]. \tag{4.2}$$

The cost function represents the average total amount of time spent in queue by all the faults arisen in the interval $[0, N)$ (including the ones already present at the beginning of the interval), and is therefore equal to their average overall service time.

An estimate of the individual average *fault service time* achieved by a control policy π can be obtained by dividing $J_{\pi,N}(i)$ by the average number of faults arisen in $[0, N-1]$, at least if $i^q = i^\ell = 0$. In the case of Poisson arrival process described by equation 2.5, the latter is equal to $N\phi$. In fact

$$E[\sum_{k=0}^{N-1} w_k^\phi \ |x_0 = i] = NE[w_k^\phi] = N\phi,$$

because the probability distribution of w_k^ϕ is supposed to be independent from the state x_k and from the controllable input u_k.

The average *fault service time* $E[T^\phi]$ can therefore be estimated as follows:

$$E[T^\phi] \approx \frac{J_{\pi,N}(i)}{N\phi}.$$

Moreover it is possible to see that, under assumption typically satisfied by the dynamical system and by the policy,

$$E[T^\phi] = \frac{J_\pi(i)}{\phi},$$

a fact that is heuristically confirmed by noticing that

$$\lim_{N\to\infty} \frac{J_{\pi,N}(i)}{N\phi} = \frac{J_\pi(i)}{\phi}$$

by definition of average cost per stage.

The important consequence of these considerations is that minimizing the cost function is equivalent to minimize the *fault service time*, hence the optimal control problem formulation is complete. We may now concentrate our attention to the problem of analyzing the optimal control policy that minimizes such cost function, a topic covered in chapter 5.

Chapter 5

Properties of the optimal control policy

Considered the optimal control framework for the *fault service time* minimization, developed in [3] and described in section 4.3, the goal is to find an optimal policy. Dynamic programming techniques could be used to solve the problem, at least when the system spaces are finite. In that case through a dynamic programming approach it is possible to find both the optimal control policy π^* in tabular form and the optimal cost for each initial state. Even if some problems arise when system spaces are infinite like in our case (x^Q is set to $+\infty$), it is usually possible to approximate the VMM system with one described by a finite space state. If the state space of the approximate system is large enough, it is possible to apply dynamic programming techniques to obtain a solution in table form that, in most cases, is a good approximation of the optimal control policy for the original system.
Unfortunately the space state often results very large, hence in such cases a control policy in table form is not a good choice in terms of implementation efficiency. In practice, a closed form feedback law is often necessary to meet the tight efficiency requirements imposed by the setting in which the con-

trol algorithm operates. Therefore the role of dynamic programming should be limited to obtain insights about the the problem and as a benchmark to compare candidate control policies with the optimal choice.
Moreover the dynamic programming approach needs to know the statistical description of all the disturbances, including that of the fault process w_k^{ϕ}. Unfortunately, apart from the simplistic description given in chapter 3, a good statistical description is generally not known at design time and it is very difficult to obtain one. Moreover it is certainly not stationary, hence even if a statistical description is known at run time, an online dynamic programming approach would be certainly unpractical for its huge computational overhead. Therefore a stronger understanding of the problem is needed, to get the insights needed to develop a closed form control policy that achieves semi-optimal results under a wide variety of workloads.
In this chapter we will study the problem with reference to the model described in chapter 3, to obtain some insights about the structure of the optimal control policy. In our particular case there are two controllable inputs u^{ℓ} and u^{s}, hence the optimal control policy for the control horizon T is in the form:

$$\pi^* = \{\mu_k, k \in T\},$$

where

$$\mu_k(x) = (u^{\ell}_{opt}(x,k), u^{s}_{opt}(x,k)) \qquad k \in T.$$

Throughout this chapter we will always make use of the Bernoulli $b(y)$-DS model proposed in [3] and described in chapter 3, mostly for its simplicity.

5.1 Useful reformulations of the cost function

To effectively study the optimal control problem, it is useful to reformulate the cost function 4.2 in a way that conveys more clearly the effects of the controllable inputs on the cost.

For ease of reference, we rewrite the transition equations 2.4, that describe the evolution of the system in matrix form:

$$x_{k+1} = f(x_k, u_k, w_k) = Ax_k + B \begin{bmatrix} u_k \\ w_k \end{bmatrix}$$

$$\begin{bmatrix} x^f_{k+1} \\ x^q_{k+1} \\ x^\ell_{k+1} \\ x^s_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x^f_k \\ x^q_k \\ x^\ell_k \\ x^s_k \end{bmatrix} + \underbrace{\begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix}}_{B} \begin{bmatrix} u^\ell_k \\ u^s_k \\ w^\ell_k \\ w^s_k \\ w^\phi_k \end{bmatrix}.$$

For the known properties of linear systems

$$x_k = A^k x_0 + \sum_{i=0}^{k-1} A^{k-i-1} B u(i),$$

but A is the identity matrix, hence

$$x_k = x_0 + \sum_{i=0}^{k-1} Gu(i) = x_0 + G \sum_{i=0}^{k-1} u(i).$$

Therefore the state at time k depends only on the sum of the values of the total input signal (that includes both controllable inputs and disturbances), not on their particular order.

As a consequence,

$$x^q_k = x^q_0 + \sum_{i=0}^{k-1} w^\phi_i - \sum_{i=0}^{k-1} u^\ell_i, \tag{5.1}$$

$$x^\ell_k = x^\ell_0 + \sum_{i=0}^{k-1} u^\ell_i - \sum_{i=0}^{k-1} w^\ell_i; \tag{5.2}$$

by summing equations 5.1 and 5.2 we obtain

$$x^\ell_k + x^q_k = x^\ell_0 - \sum_{i=0}^{k-1} w^\ell_i + x^q_0 + \sum_{i=0}^{k-1} w^\phi_i. \tag{5.3}$$

We recall the definition 4.2 of the cost for an initial state i over a finite horizon of N steps with control policy π:

$$J_{\pi,\ N}(i) = E[\sum_{k=0}^{N-1} x_k^q + x_k^\ell \mid x_0 = i].$$

By substituting equation 5.3 into 4.2 we obtain

$$J_{\pi,\ N}(i) = E[\sum_{k=0}^{N-1}(x_0^\ell - \sum_{n=0}^{k-1} w_n^\ell + x_0^q + \sum_{n=0}^{k-1} w_n^\phi) \mid x_0 = i].$$

Therefore it easily follows that

$$J_{\pi,\ N}(i) = Nx_0^\ell + Nx_0^q + E[\sum_{k=0}^{N-1}(-\sum_{n=0}^{k-1} w_n^\ell + \sum_{n=0}^{k-1} w_n^\phi) \mid x_0 = i]$$

and also

$$J_{\pi,\ N}(i) = Nx_0^\ell + Nx_0^q + E[\sum_{k=0}^{N-1}(-\sum_{n=0}^{k-1} w_n^\ell + k\phi) \mid x_0 = i].$$

Moreover

$$J_{\pi,\ N}(i) = Nx_0^\ell + Nx_0^q + \frac{N(N-1)}{2}\phi - E[\sum_{k=0}^{N-1}(\sum_{n=0}^{k-1} w_n^\ell) \mid x_0 = i],$$

or equivalently

$$J_{\pi,\ N}(i) = Nx_0^\ell + Nx_0^q + \frac{N(N-1)}{2}\phi - E[\sum_{k=1}^{N-1}(\sum_{n=0}^{k-1} w_n^\ell) \mid x_0 = i],$$

that yields the following reformulation of the cost function

$$J_{\pi,\ N}(i) = Nx_0^\ell + Nx_0^q + \frac{N(N-1)}{2}\phi - E[\sum_{k=0}^{N-2} w_n^\ell(N-1-k) \mid x_0 = i], \quad (5.4)$$

that will be often used while analyzing the optimal control policy. Equation 5.4 also points out that *fault service time* is directly influenced only by load operations, stressing the fact that some priority should be given to loads over

stores on the short term.

The cost function can also be rewritten in the following equivalent forms:

$$J_{\pi,\ N}(i) = Nx_0^\ell + Nx_0^q + E[\sum_{k=0}^{N-1}(-\sum_{n=0}^{k-1} w_n^\ell + \sum_{n=0}^{k-1} w_n^\phi) \mid x_0 = i]$$

$$J_{\pi,\ N}(i) = Nx_0^\ell + Nx_0^q + E[\sum_{k=0}^{N-1}\sum_{n=0}^{k-1} (\phi - w_n^\ell) \mid x_0 = i]$$

$$J_{\pi,\ N}(i) = Nx_0^\ell + Nx_0^q + E[\sum_{k=0}^{N-2} (\phi - w_k^\ell)(N-1-k) \mid x_0 = i].$$

Considering the latter equation, the cost function can be interpreted as the cumulative length of a queue with w_k^ϕ as arrival process (with an average arrival rate of ϕ) and w_k^ℓ as departure process.

5.2 The ideal case of unlimited free frames

In this section, we will firstly consider the problem of finding the optimal control policy in the ideal case of unlimited free frames ($x_0^f = \infty$), an highly advantageous setting in terms of mathematical tractability. In this way we'll be able to obtain some useful insights about the way to tackle the original unrelaxed problem, namely the case in which the number of free frames is limited. Moreover we will estimate the *fault service time* associated to the optimal control policy, that clearly represents a lower bound for the one relative to the original problem.

5.2.1 The optimal control policy

It is clear by intuition that in this context there is no reason to issue store operations because with an unlimited pool of free frames there's never the need to free more pages. Moreover issuing store operations on average delays load operations and therefore increases the expected *fault service time*.

As far as the load operations are concerned, intuitively there's no reason to delay the forwarding of a load operation to the disk (of course as long as it is possible to do so, which means that the disk-subsystem queue is not full). On the contrary, increasing the number of load operations has positive effects on the bandwidth, therefore we will prove that in this basic case the optimal control policy is:

$$\mu^{\ell}_{opt}(i,k) = \mu^{L}(i) \ \forall \ i, \ \forall k, \tag{5.5}$$

$$\mu^{s}_{opt}(i,k) = 0 \ \forall \ i, \ \forall k, \tag{5.6}$$

where $u^L(i)$ is the largest admissible value for u^ℓ in the state i, that is $u^L(i) = \min(i^q, x^D - i^\ell - i^s + 1)$.

To prove this statement, let's recall the alternative formulation 5.4 of the expected cost $J_{\pi,\ N}(i)$ over a finite horizon $T = [0, N-1]$ of N steps with control policy π and initial state i:

$$J_{\pi,\ N}(i) = Nx^{\ell}_0 + Nx^{q}_0 + \frac{N(N-1)}{2}\phi - E[\sum_{k=0}^{N-2} w^{\ell}_k(N-1-k) \mid x_0 = i].$$

It's clear that for $N = 1$, $J_{\pi,\ N}(i)$ is not influenced by the control policy and depends only on the initial state i.
Since the controllable inputs have effects only on the sum involving w^ℓ_k, trying to minimize $J_{\pi,\ N}$ over the possible control policies π is equivalent to maximize the following non-negative index

$$H_{\pi,\ N}(i) = E[\sum_{k=0}^{N-2} w^{\ell}_k(N-1-k) \mid x_0 = i].$$

We will also call

$$H^*_N(i) = \max_{\pi} H_{\pi,N}(i)$$

the maximum achievable value of $H_{\pi,N}(i)$ with an admissible control policy over an horizon of N steps with initial state i. We will prove by induction the following properties:

- $H^*_N(i)$ does not depend on i^f
- $H^*_N(i)$ is monotonically decreasing with i^s
- $H^*_N(i)$ is monotonically increasing with i^ℓ
- $u^\ell_{opt}(i,k) = u^L(i)\ \forall\ i,\ \forall\ k = 0\ ,\cdots,\ N-2$
- $u^s_{opt}(i,k) = 0\ \forall\ i,\ \forall\ k = 0\ ,\cdots,\ N-2$
- $H^*_N(x^q, x^\ell - 1, x^s) + N \geq H^*_N(x^q, x^\ell, x^s - 1) \geq H^*_N(x^q, x^\ell, x^s)$

We choose $N = 2$ as the base case of induction because, as we have already seen, $H_{\pi,\ 1}(i)$ is not well defined, in the sense that $H_{\pi,\ 1}(i) = 0$ whichever is the control policy π. With $N = 2$ we obtain

$$H_{\pi,\ 2}(i) = E[w^\ell_0 \mid x_0 = i] = \frac{b(x^s_0 + x^\ell_0 + u^\ell_0 + u^s_0)}{x^s_0 + x^\ell_0 + u^\ell_0 + u^s_0}(x^\ell_0 + u^\ell_0),$$

where $i = (x^f_0, x^q_0, x^\ell_0, x^s_0) = (i^f, i^q, i^\ell, i^s)$. Considered that as an hypothesis we are considering a disk subsystem whose bandwidth is such that if $z \geq y$, $b(z)/z \leq b(y)/y$ (which means that increasing the number of request can never increase the bandwidth per request), it is clear that

$$u^s_{opt}(i,0) = 0\ \forall\ i.$$

Similarly it is easy to prove that

$$u^\ell_{opt}(i,0) = u^L(i)\ \forall\ i,$$

where $u^L(i)$ is the largest admissible value for u^ℓ in the state i. Therefore

$$H^*_2(i) = \frac{b(x^s_0 + x^\ell_0 + u^L(i))}{x^s_0 + x^\ell_0 + u^L(i)}(x^\ell_0 + u^L(i)).$$

It's important to note that $u^L(i)$ does not depend on the number of free frames i^f, essentially because we are considering an ideal case with an unlimited amount of free pages. In a more realistic case, $u^L(i)$ would be limited

by the number of free frames available, but under the assumptions made $H_2^*(i)$ does not depend on i^f.
With considerations analogous to the one already made, it's easy to prove that $H_N^*(i)$ is monotonically decreasing and increasing with i^s and i^ℓ respectively. The last property we need to prove is that

$$\frac{b(x_0^s + x_0^\ell - 1 + u^L(i))}{x_0^s + x_0^\ell - 1 + u^L(i)}(x_0^\ell + u^L(i) - 1) + 2 \geq \frac{b(x_0^s - 1 + x_0^\ell + u^L(i))}{x_0^s - 1 + x_0^\ell + u^L(i)}(x_0^\ell + u^L(i)),$$

that holds if and only if

$$-\frac{b(x_0^s + x_0^\ell - 1 + u^L(i))}{x_0^s + x_0^\ell - 1 + u^L(i)} + 2 \geq 0$$

that is true considered that $b(z)/z \leq 1$. This last property we have proved could seem quite strange, but it is fundamental for the development of the proof. In fact, considered a given state i and a certain control law $\mu(x) = (u^\ell(i), u^s(i))$, in the general case there are 3 possible events:

- the disk performs a load operation
- the disk performs a store operation
- the disk performs no operation at all

The last property simply means that the most desirable option is the first, then the second and finally the third. Intuitively, completing a store operation is slightly better than doing nothing, simply because then future loads will be less delayed. The importance of this part will be more clear hereinafter.

Let's proceed by induction to prove the general case, with a dynamic programming approach in mind. We consider a general initial state $i = (x^f, x^q, x^\ell, x^s)$ and we write a recursive formula for $H_N^*(i)$, assuming $H_n^*(j)$

known $\forall j,\ \forall n \leq N-1$. As we have already said, if we do not consider the events related to the arrival process w_0^ϕ (that is independent from the system and the controllable inputs), there are 3 possible events related to the system:

- event A, the disk performs a load operation
- event B, the disk performs a store operation
- event C, the disk performs no operation at all

each of them with an associated probability that depends also on the controllable inputs.
$H_N^*(i)$ can be computed in the following way:

$$\begin{aligned} H_N^*(x^f, x^q, x^\ell, x^s) = \\ \max_{(u_0^\ell, u_0^s)\in U(i)} \left((N-1) + E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell - 1, x^s + u_0^s)]\right) \frac{b(x^s + x^\ell + u_0^\ell + u_0^s)}{x^s + x^\ell + u_0^\ell + u_0^s}(x_0^\ell + u_0^\ell) + \\ E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell, x^s + u^s - 1)] \frac{b(x^s + x^\ell + u_0^\ell + u_0^s)}{x^s + x^\ell + u_0^\ell + u_0^s}(x^s + u_0^s) + \\ E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell, x^s + u_0^s)](1 - b(x^s + x^\ell + u_0^\ell + u_0^s)). \end{aligned}$$

This recursive formula for $H_N^*(i)$ keeps into account the fact that, once one of the 3 events has happened, we consider to use the optimal policy over $N-1$ steps with the newly reached state q as initial state.
The symbol $E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell - 1, x^s + u_0^s)]$ stands for an expectation taken over the probability distribution of w_0^ϕ, a random variable we are supposing independent from anything else in the system. For the inductive hypothesis we have:

$$(N-1) + H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell - 1, x^s + u_0^s) \geq H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell, x^s + u_0^s - 1)$$

and taking an expectation of both sides yields

$$\left((N-1) + E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell - 1, x^s + u_0^s)]\right) \geq E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell, x^s + u_0^s - 1)].$$

In a similar way we find:

$$E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell, x^s + u_0^s - 1)] \geq E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell, x^s + u_0^s)].$$

These fundamental inequalities mean that when there are N steps left in the control horizon, the most favorable event for the system is that of a load

performed by the disk subsystem.

We will now consider the problem of determining the optimal values of u_0^ℓ, u_0^s. From the inductive hypothesis we know that $H_{N-1}^*()$ is a decreasing function of x^s, therefore it appears clear that increasing u_0^s reduces $H_N^*()$. Unfortunately it's not so easy to prove this statement because changing u_0^s also changes the probabilities of events A, B, C. In particular it increases the probability of event B (a store operation is performed) decreasing the ones of A e C. By noticing that $H_{N-1}^*()$ is a decreasing function of x^s, the following inequality holds:

$$E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell, x^s)] \geq E_{w_0^\phi}[H_{N-1}^*(x^q - u_0^\ell + w_0^\phi, x^\ell + u_0^\ell, x^s + u_0^s - 1)]$$

if $u_0^s > 0$. This means that the worst case (event C, no operations performed) with $u_0^s = 0$ is still better than event B(a store is performed) with $u_0^s > 0$. Therefore increasing the probability of event B with $u_0^s > 0$ does not benefit $H_N^*()$ in any case.

In conclusion, the best option is to choose $u_0^s = u_{opt}^s(i, 0) = 0$.

As far as u_0^ℓ is concerned, it's easy to see that increasing its value not only increases $H_{N-1}^*()$ (it's an increasing function of x^ℓ) but it also raises the probability of event A, which is the most favorable to the system.
Therefore the best option is to choose $u_0^\ell = u_{opt}^\ell(i, 0) = u^L(i)$.
With these choices for u_0^ℓ and u_0^s, we have for $i = (x^f, x^q, x^\ell, x^s)$

$$\begin{aligned} H_N^*(i) = \Big((N-1) + E_{w_0^\phi}[H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell + u^L(i) - 1, x^s)]\Big) \frac{b(x^s + x^\ell + u^L(i))}{x^s + x^\ell + u^L(i)} \Big(x^\ell + u^L(i)\Big) + \\ E_{w_0^\phi}[H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell + u^L(i), x^s - 1)] \frac{b(x^s + x^\ell + u^L(i))}{x^s + x^\ell + u^L(i)} (x^s) + \\ E_{w_0^\phi}[H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell + u^L(i), x^s)] \Big(1 - b(x^s + x^\ell + u^L(i))\Big). \end{aligned} \tag{5.7}$$

Now it's easy to see that since $u^L(i)$ does not depend on i^f, neither does $H_N^*(x^f, x^q, x^\ell, x^s)$. The fact that $H_N^*(x^q, x^\ell, x^s)$ is a decreasing and increasing function of x^s and x^ℓ respectively descends from considerations analogous at the one carried out while considering the optimal choice for u_0^ℓ and u_0^s. In fact increasing x^s has the same effect of increasing u_0^s, and the same property

holds for x^ℓ and u_0^ℓ.

It remains to prove that even while considering a control horizon of N steps, the most favorable event is always that of a load performed by the disk subsystem, that is:

$$H_N^*(x^q, x^\ell - 1, x^s) + N \geq H_N^*(x^q, x^\ell, x^s - 1) \geq H_N^*(x^q, x^\ell, x^s).$$

The inequality $H_N^*(x^q, x^\ell, x^s - 1) \geq H_N^*(x^q, x^\ell, x^s)$ descends directly from the monotonicity of $H_N^*(\cdot)$ that we have already proved. Let's prove that

$$H_N^*(x^q, x^\ell - 1, x^s) + N \geq H_N^*(x^q, x^\ell, x^s - 1). \tag{5.8}$$

By the means of equation 5.7, we can think $N + H_N^*(x^q, x^\ell - 1, x^s)$ as the expectation of a random variable $\mathbf{z}$ defined as follows:

$$\mathbf{z} = \begin{cases} N + (N-1) + E_{w_0^\phi}[H_{N-1}^*(x^\ell - 2 + u^L(i), x^s, x^q - u^L(i) + w_0^\phi)] & = a_1 \quad \text{L,L} \quad \text{with probability } p_1 \\ N + E_{w_0^\phi}[H_{N-1}^*(x^\ell - 1 + u^L(i), x^s - 1, x^q - u^L(i) + w_0^\phi)] & = a_2 \quad \text{L,S} \quad \text{with probability } p_2 \\ N + E_{w_0^\phi}[H_{N-1}^*(x^\ell - 1 + u^L(i), x^s, x^q - u^L(i) + w_0^\phi)] & = a_3 \quad \text{L,N} \quad \text{with probability } p_3 \end{cases}$$

simply by putting

$$\begin{aligned} i &= (x^f, x^q, x^\ell - 1, x^s) \\ p_1 &= \tfrac{b(x^s + x^\ell - 1 + u^L(i))}{x^s + x^\ell - 1 + u^L(i)}(x^\ell + u^L(i) - 1) \\ p_2 &= \tfrac{b(x^s + x^\ell - 1 + u^L(i))}{x^s + x^\ell - 1 + u^L(i)}(x^s) \\ p_3 &= 1 - b(x^s + x^\ell - 1 + u^L(i)). \end{aligned}$$

In particular, event a_1 represents the case of two consecutive loads performed by the disk subsystem, while event a_2 that of a load and a store performed in that order. If event a_3 occurs, the disk subsystem completes a load operation in the first step and no operation in the second.

Under these assumptions,

$$N + H_N^*(x^q, x^\ell - 1, x^s) = E[\mathbf{z}] = a_1 p_1 + a_2 p_2 + a_3 p_3.$$

In an analogous manner we can define

$$\mathbf{q} = \begin{cases} (N-1) + E_{w_0^\phi}[H_{N-1}^*(x^\ell - 1 + u^L(j), x^s - 1, x^q - u^L(j) + w_0^\phi)] & = b_2 \quad \text{S,L} \quad \text{with probability } \bar{p_2} \\ E_{w_0^\phi}[H_{N-1}^*(x^\ell + u^L(j), x^s - 2, x^q - u^L(j) + w_0^\phi)] & = b_1 \quad \text{S,S} \quad \text{with probability } \bar{p_1} \\ E_{w_0^\phi}[H_{N-1}^*(x^\ell + u^L(j), x^s - 1, x^q - u^L(j) + w_0^\phi)] & = b_3 \quad \text{S,N} \quad \text{with probability } \bar{p_3} \end{cases}$$

where

$$\begin{aligned} j &= (x^f, x^q, x^\ell, x^s - 1) \\ \bar{p_2} &= \tfrac{b(x^s+x^\ell-1+u^L(j))}{x^s+x^\ell-1+u^L(j)}(x^\ell + u^L(j)) \\ \bar{p_1} &= \tfrac{b(x^s-1+x^\ell+u^L(j))}{x^s+x^\ell-1+u^L(j)}(x^s - 1) \\ \bar{p_3} &= 1 - b(x^s - 1 + x^\ell + u^L(j)). \end{aligned}$$

In this way we obtain $H_N^*(x^q, x^\ell, x^s - 1)$ as an expectation of $\mathbf{q}$

$$H_N^*(x^q, x^\ell, x^s - 1) = E[\mathbf{q}] = b_1\bar{p_1} + b_2\bar{p_2} + b_3\bar{p_3}.$$

Now if we consider the definition of $u^L(x) = \min(x^q, x^f, x^D - x^\ell - x^s)$, with $x^f = \infty$ we have

$$\begin{aligned} i &= (x^f, x^q, x^\ell - 1, x^s), \\ j &= (x^f, x^q, x^\ell, x^s - 1), \\ u^L(j) &= \min(x^q, x^D - x^\ell - x^s + 1) = u^L(i), \end{aligned}$$

which also means that $\bar{p_3} = p_3$. Moreover the following inequalities hold

$$a_1 \geq b_1 \qquad a_2 \geq b_2 \qquad a_3 \geq b_3.$$

In fact ,using the inductive hypothesis we obtain

$$\begin{aligned} &H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell + u^L(i), x^s - 2) \leq \\ &(N-1) + H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell - 1 + u^L(i), x^s - 1) \leq \\ &(N-1) + (N-1) + H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell - 2 + u^L(i), x^s) \leq \\ &(N-1) + (N) + H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell - 2 + u^L(i), x^s), \end{aligned}$$

hence taking an expectation on both sides yields

$$\begin{aligned} b_1 = E_{w_0^\phi}[H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell + u^L(i), x^s - 2)]) \leq \\ (N-1) + (N) + E_{w_0^\phi}[H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell - 2 + u^L(i), x^s)] = a_1. \end{aligned}$$

Obviously

$$\begin{aligned} &N + H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell + u^L(i) - 1, x^s - 1) \geq \\ &(N-1) + H_{N-1}^*(x^q - u^L(i) + w_0^\phi, x^\ell + u^L(i) - 1, x^s - 1), \end{aligned}$$

which means $a_2 \geq b_2$. Moreover

$$\begin{aligned}
&H^*_{N-1}(x^q - u^L(i) + w_0^\phi, x^\ell + u^L(i), x^s - 1) \leq \\
&(N-1) + H^*_{N-1}(x^q - u^L(i) + w_0^\phi, x^\ell - 1 + u^L(i), x^s) \leq \\
&N + H^*_{N-1}(x^q - u^L(i) + w_0^\phi, x^\ell - 1 + u^L(i), x^s),
\end{aligned}$$

that implies $a_3 \geq b_3$. Furthermore from the inductive hypothesis two more inequalities hold:

$$a_1 \geq a_2 \qquad b_2 \geq b_1.$$

As we have seen so far,

$$H^*_N(x^q, x^\ell - 1, x^s) + N \geq H^*_N(x^q, x^\ell, x^s - 1) \iff a_1 p_1 + a_2 p_2 + a_3 p_3 \geq b_1 \bar{p_1} + b_2 \bar{p_2} + b_3 \bar{p_3},$$

where $p_3 = \bar{p_3}$. In this case, considered that $a_3 \geq b_3$,

$$a_1 p_1 + a_2 p_2 \geq b_1 \bar{p_1} + b_2 \bar{p_2} \Longrightarrow a_1 p_1 + a_2 p_2 + a_3 p_3 \geq b_1 \bar{p_1} + b_2 \bar{p_2} + b_3 \bar{p_3},$$

where $p_1 + p_2 = \bar{p_1} + \bar{p_2} = 1 - p_3$. But if we consider that

$$a_1 \geq a_2 \geq b_2 \geq b_1,$$

it is clear that $a_1 p_1 + a_2 p_2 \geq b_1 \bar{p_1} + b_2 \bar{p_2}$ whichever are the probabilities $p_1, p_2, \bar{p_1}, \bar{p_2}$. In fact we have that

$$\frac{a_1 p_1 + a_2 p_2}{1 - p_3} \geq a_2 \geq b_2 \geq \frac{b_1 \bar{p_1} + b_2 \bar{p_2}}{1 - p_3}$$

which concludes our proof.

5.2.2 An estimate of the fault service time

In this ideal setting of unlimited free frames, it is also useful to estimate the expected cost associated with the optimal control policy described by equations 5.5 and 5.6.

Assuming $x_0^f = +\infty$ and $x_0^s = 0$, it is easy to see that, at any step k, $x_k^f = +\infty$ and $x_k^s = 0$. Therefore the transition equations of interest remain

$$x_{k+1}^q = x_k^q - u_k^\ell + w_k^\phi, \tag{5.9}$$

$$x_{k+1}^\ell = x_k^\ell + u_k^\ell - w_k^\ell. \tag{5.10}$$

As an additional simplification, we assume $x^D = \infty$ therefore obtaining two queues with infinite maximum length. Then we have

$$u_k^\ell = u^L(x_k) = \min\left(x_k^q, x_k^f, x^D - x_k^\ell - x_k^s\right) = x_k^q.$$

By substitution into equations 5.9 and 5.10, we easily obtain:

$$x_k^q = w_{k-1}^\phi,$$
$$x_{k+1}^\ell = x_k^\ell + w_{k-1}^\phi - w_k^\ell.$$

In this context, x_k^ℓ represents the number of requests in the disk subsystem's queue at step k, w_{k-1}^ϕ the arrival process while w_k^ℓ stands for the number of services completed by the disk in the time interval $[k, k+1]$. As we will see, the average length of this queue is closely related to the average *fault service time*. In fact, if the system has reached equilibrium, then let

$$N = \lim_{k\to\infty} E[x_k^\ell]$$

be the average length of the queue.
If W_i is a random variable that represents the time spent in the queue by request i before being served, we put

$$\overline{W}_i = E[W_i],$$

$$W = \lim_{i\to\infty} \overline{W}_i.$$

Then by Little's Theorem we have

$$L = \phi W,$$

where ϕ is the average arrival rate. Now it's easy to prove that the average *fault service time* when there are unlimited free frames is

$$E[T_{uff}] = 1 + W = 1 + \frac{L}{\phi},$$

where the term 1 represents the fact that a fault arriving at step $k-1$ enters the disk queue at step k, causing a minimum *fault service time* of at least one step.

We begin our study on the dynamics of the queue's length by noticing that, under the hypothesis we have made, it is represented by a discrete time Markov Chain. In fact it's easy to see that if we choose the length of the queue as the state of the system, then the transition probabilities depend only on the current state and not on past events. This assumption descends from the fact that the arrival process w_{k-1}^{ϕ} is assumed to be independent identically distributed (i.i.d.) while the probability distribution of the departure process w_k^{ℓ} is

$$P[w_k^{\ell} = 1 | x_k^{\ell} + w_{k-1}^{\phi} = i] = b(i) = b_i$$
$$P[w_k^{\ell} = 0 | x_k^{\ell} + w_{k-1}^{\phi} = i] = 1 - b(i) = 1 - b_i,$$

thus it is completely specified by the state of the Markov Chain x_k^{ℓ} and by the value of the random variable w_{k-1}^{ϕ} that is independent from past states of the Markov chain. Therefore x_k^{ℓ} exhibits the Markov property

$$P[x_{k+1}^{\ell} = a | x_k] = P[x_{k+1}^{\ell} = a | x_k, \mathcal{P}],$$

where $\mathcal{P}$ represents any event related to the past history of the process x_k^{ℓ}. We introduce now another simplification assuming a Bernoulli probability distribution for w_{k-1}^{ϕ}:

$$P[w_{k-1}^{\phi} = 1] = \phi,$$
$$P[w_{k-1}^{\phi} = 0] = 1 - \phi,$$
$$E[w_{k-1}^{\phi}] = \phi.$$

In this way we obtain a discrete time Death-Birth process. The graph of the Markov chain associated to the system becomes that shown in figure 5.1,

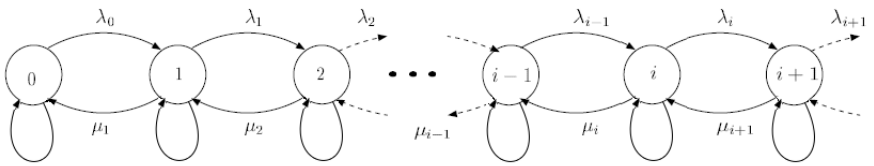

Figure 5.1: Graph of the Markov chain with a Bernoulli arrival process.

where

$$\begin{aligned} P[x_{k+1}^\ell = i+1 | x_k^\ell = i] &= \phi(1-b_{i+1}) &= \lambda_i \\ P[x_{k+1}^\ell = i-1 | x_k^\ell = i] &= (1-\phi)b_i &= \mu_i \\ P[x_{k+1}^\ell = i | x_k^\ell = i] &= (1-\phi)(1-b_i) + b_{i+1}\phi\ . \end{aligned}$$

To derive a stationary distribution, we note that if we find a probability distribution $\pi(i)$ such that

$$\pi(i)P[i \to j] = \pi(j)P[j \to i], \tag{5.11}$$

then $\pi(i)$ is a stationary distribution (note that this is not a necessary condition). In our case equation 5.11 becomes:

$$\pi(n)\phi(1-b_{n+1}) = \pi(n+1)(1-\phi)b_{n+1},$$

that is for every $n \in \mathbb{N}$

$$\pi(n+1) = \frac{\phi(1-b_{n+1})}{(1-\phi)b_{n+1}}\pi(n)$$

that yields

$$\pi(n) = \prod_{i=1}^{n} \frac{\phi(1-b_i)}{(1-\phi)b_i}\pi(0).$$

The factor $\pi(0)$ can be found by noticing that $\pi(\cdot)$ is a probability distribution and therefore

$$\sum_{k=0}^{\infty} \pi(k) = 1,$$

$$\sum_{k=0}^{\infty} \prod_{i=1}^{k} \frac{\phi(1-b_i)}{(1-\phi)b_i}\pi(0) = 1.$$

hence

$$\pi(0) = \frac{1}{\sum_{k=0}^{\infty} \prod_{i=1}^{k} \frac{\phi(1-b_i)}{(1-\phi)b_i}}. \tag{5.12}$$

We have to pay attention to the fact that the above equations are valid only if the infinite sum converges; a sufficient condition will be derived for that. If that happens, we can express the expected length of the queue L in a closed form formula as follows:

$$L = \sum_{k=0}^{\infty} \pi(k)k.$$

A sufficient condition for equation (5.12) to be well defined is

$$\exists s \in \mathbb{N} \mid \frac{\phi}{b_s} < 1,$$

which means that with a queue of length s the disk has enough bandwidth to serve on average more requests per step than those that arrive, on average, in one step. In fact we know that if $y \geq s$ then $b_y \geq b_s$, hence by noticing that $\frac{1-x}{x}$ is monotonically decreasing we have, for every $k \geq s$

$$\prod_{i=1}^{k} \frac{\phi(1-b_i)}{(1-\phi)b_i} \leq \frac{\phi^k}{(1-\phi)^k} \alpha \left(\frac{1-b_s}{b_s}\right)^{k-s},$$

where $\alpha = \prod_{i=1}^{s} \frac{(1-b_i)}{b_i}$. Moreover

$$\begin{aligned}
\sum_{k=0}^{\infty} \prod_{i=1}^{k} \frac{\phi(1-b_i)}{(1-\phi)b_i} \pi(0) &= \sum_{k=0}^{s-1} \prod_{i=1}^{k} \frac{\phi(1-b_i)}{(1-\phi)b_i} \pi(0) + \sum_{k=s}^{\infty} \prod_{i=1}^{k} \frac{\phi(1-b_i)}{(1-\phi)b_i} \pi(0) \\
&\leq a + \sum_{k=s}^{\infty} \frac{\phi^k}{(1-\phi)^k} \alpha \left(\frac{1-b_s}{b_s}\right)^{k-s} = a + \alpha\beta \sum_{k=s}^{\infty} \frac{\phi^k}{(1-\phi)^k} \left(\frac{1-b_s}{b_s}\right)^{k} \\
&= a + \alpha\beta \sum_{k=s}^{\infty} \left(\frac{\phi}{b_s}\right)^{k} \left(\frac{1-b_s}{1-\phi}\right)^{k} < \infty.
\end{aligned}$$

We shall now consider the case of a Poisson distributed arrival process, with the following probability distribution

$$P[w_k^{\phi} = h] = \frac{\phi^h e^{-\phi}}{h!} = a_h.$$

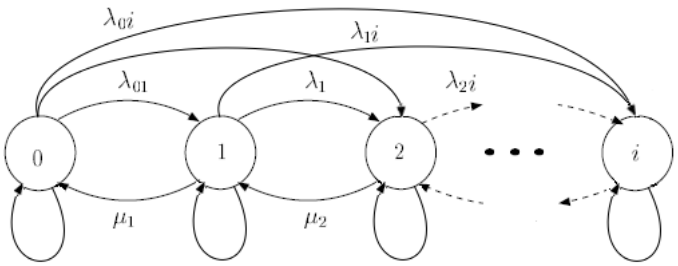

Figure 5.2: Graph of the Markov Chain with a Poisson arrival process.

Like in the Bernoulli case considered before, also in this case the average number of arrivals per step is ϕ. The resultant stochastic process that models the length of the queue is clearly still a Markov Chain but it is not a Death Birth process anymore. In fact in this new setting it is possible to see more than one arrival in one single step, as it is represented in figure 5.2 that shows the graph associated to the Markov Chain. The resulting discrete time dynamic process has the following transition probabilities:

$$p_0(t+1) = a_0(p_0(t) + p_1(t)b_1) + a_1 p_0(t)b_1,$$

$$p_n(t+1) = \textstyle\sum_{i=0}^{n} a_i(p_{n-i}(t)(1-b_n) + p_{n+1-i}b_{n+1}) + a_{n+1}p_0(t)b_{n+1},$$

where $p_j(t)$ is the probability of having j requests in queue at time t and b_j has the same meaning as in the previously considered case.
If we impose the stationary condition $p_j(t+1) = p_j(t) = \pi(j)\ \forall\ j \in \mathbb{N}$ we obtain the following set of equations:

$$\pi(n) = \gamma_n \pi(0) \quad \forall n \geq 0$$

where

$$\gamma_{n+1} = \frac{\gamma_n - \sum_{i=1}^{n} a_i(\gamma_{n-i}(1-b_n) + b_{n+1}\gamma_{n+1-i}) - a_{n+1}b_{n+1} - a_0\gamma_n(1-b_n)}{a_0 b_{n+1}}$$

and $\gamma_0 = 1$. If we add the further constraint:

$$\sum_{k=0}^{\infty} \pi(k) = \sum_{k=0}^{\infty} \gamma_k \pi(0) = 1,$$

we obtain the value of $\pi(0) = \frac{1}{\sum_{k=0}^{\infty} \gamma_k}$ and consequently of $\pi(j)\ \forall j \in \mathbb{N}$. Now that we have found a formula for the stationary probabilities of every state, we easily obtain

$$L = \sum_{k=0}^{\infty} \pi(k)k,$$

which leads to a closed form formula for the average *fault service time* $E[T_{uff}]$ in the case of a Poisson arrival process.

The lower bound just found on the *fault service time* is probably tight in settings with low fault rates (that is when store operations do not delay significantly load ones). In fact we are considering a case in which store operations never delay the loads, hence the more the assumption is realistic the more the bound is tight.

5.3 Optimal load policy

The aim of this section is to characterize the optimal time-variant control policy $\pi^* = \{\mu_k, k \in T = [0, 1, \ldots, N-1]\}$ for the original unrelaxed problem, with an approach similar to the one developed in the previous section. With the backwards approach of dynamic programming in mind, it is convenient to define

$$\pi^*(i,k) = (\widetilde{u}^{\ell}_{opt}(i,k), \widetilde{u}^{s}_{opt}(i,k)) = \mu_{N-k}(i) = (u^{\ell}_{opt}(i,N-k), u^{s}_{opt}(i,N-k)),$$

that is the optimal controllable inputs to be used for state i when there are $k-1$ steps left of the control horizon.
In particular, if the control horizon is in the form $[0, 1, \ldots, N-1]$ then $\pi^*(i,k)$ is relative to time $N-k$.
In particular we will prove that, with a control horizon of N steps, $\pi^*(i,k)$

is always in the form:

$$\pi^*(i,k) = (\widetilde{u}^{\ell}_{opt}(i,k), \widetilde{u}^{s}_{opt}(i,k)) = (u^L(i), \widetilde{u}^{s}_{opt}(i,k)) \ \forall\ i,\ \forall\ k = 1 \dots N,$$

where $u^L(i)$ is the largest admissible value for u^ℓ in the state i, that is $u^L(i) = \min(i^f, i^q, x^D - i^l - i^s)$. In this way the dimensionality of the minimization problem is reduced from 2 to 1, since the remaining unknown part regards only the optimal number of store operations to be issued.

To prove this statement, let's recall the reformulation 5.4 of the expected cost $J_{\pi,\ N}(i)$ over a finite horizon of N steps with control policy π and initial state i:

$$J_{\pi,\ N}(i) = Nx^l + Nx^q + \frac{N(N-1)}{2}\phi - E[\sum_{k=0}^{N-2} w^{\ell}_k(N-1-k) \mid x_0 = i].$$

It's clear that for $N = 1$, $J_{\pi,\ N}(i)$ is not influenced by the control policy and depends only on the initial state i.

Since the controllable inputs have effects only on the sum involving w^ℓ_k, trying to minimize $J_{\pi,\ N}$ over the possible control policies π is equivalent to maximize the following non negative index

$$H_{\pi,\ N}(i) = E[\sum_{k=0}^{N-2} w^{\ell}_k(N-1-k) \mid x_0 = i].$$

We will also call

$$H^*_N(i) = \max_{\pi} H_{\pi,N}(i)$$

the largest achievable value of $H_{\pi,N}(i)$ with an admissible control policy π over an horizon of N steps with initial state i. We will prove by induction the following properties:

- $\widetilde{u}^{\ell}_{opt}(i,k) = u^L(i)\ \forall\ i,\ \forall\ k = 1\ , \cdots,\ N$
- $H^*_N(x^f, x^q, x^l - 1, x^s) + N > H^*_N(x^f + 1, x^q, x^l, x^s - 1)$

- $H^*_N(x^f, x^q + 1, x^l - 2, x^s + 1) + N \geq H^*_N(x^f, x^q, x^l, x^s - 1)$ if $x^f = 0$
- $\forall\ \pi$, there exists π' such that $H_{\pi',\ N}(x^f - 1, x^q - 1, x^\ell + 1, x^s - 1) \geq H_{\pi,\ N}(x^f, x^q, x^l, x^s)$
- $\forall\ \pi$, there exists π' such that $H_{\pi',\ N}(x^f - 1, x^q - 1, x^\ell + 1, x^s) \geq H_{\pi,\ N}(x^f, x^q, x^l, x^s)$

We choose $N = 2$ as the base case of induction because, as we have already seen, $H_{\pi,\ 1}(i)$ is not well defined, in the sense that $H_{\pi,\ 1}(i) = 0$ whichever is the control policy π. Therefore we can choose $\widetilde{u}^\ell_{opt}(i, 1) = u^L(i)\ \forall\ i$, keeping in mind that any other choice would be correct.
With $N = 2$ we obtain

$$H_{\pi,\ 2}(i) = E[w^\ell_0 \mid x_0 = i] = \frac{b(x^s + x^l + u^l_0 + u^s_0)}{x^s + x^l + u^l_0 + u^s_0}(x^l + u^l_0),$$

where $i = (x^f, x^q, x^\ell, x^s) = (i^f, i^q, i^\ell, i^s)$. Considered that as an hypothesis we are considering a disk subsystem whose bandwidth is such that if $z \geq y$, $b(z)/z \leq b(y)/y$ (which means that increasing the number of request can never increase the bandwidth per request), it is clear that

$$\widetilde{u}^s_{opt}(i, 2) = 0\ \forall\ i.$$

Similarly it is easy to prove that

$$\widetilde{u}^\ell_{opt}(i, 2) = u^L(i)\ \forall\ i,$$

where $u^L(i)$ is the largest admissible value for u^ℓ in the state i. Therefore

$$H^*_2(i) = \frac{b(x^s + x^\ell + u^L(i))}{x^s + x^\ell + u^L(i)}(x^\ell + u^L(i)).$$

Moreover

$$u^L(x^f, x^q, x^\ell - 1, x^s) = \min(x^f, x^q, x^D - x^\ell - x^s + 1),$$

$$u^L(x^f{+}1, x^q, x^\ell, x^s{-}1) = \min(x^f{+}1, x^q, x^D{-}x^\ell{-}x^s{+}1) \leq \min(x^f, x^q, x^D{-}x^\ell{-}x^s{+}1){+}1.$$

As a consequence, if $i = (x^f, x^q, x^\ell - 1, x^s)$ then

$$H_2^*(x^f + 1, x^q, x^\ell, x^s - 1) \leq \frac{b(x^s - 1 + x^\ell + u^L(i) + 1)}{x^s - 1 + x^\ell + u^L(i) + 1}(x^\ell + u^L(i) + 1)$$
$$\leq \frac{b(x^s - 1 + x^\ell + u^L(i))}{x^s - 1 + x^\ell + u^L(i)}(x^\ell + u^L(i) + 1)$$

and also

$$H_2^*(x^f, x^q, x^\ell - 1, x^s) + 2 = \frac{b(x^s + x^\ell - 1 + u^L(i))}{x^s + x^\ell - 1 + u^L(i)}(x^\ell + u^L(i) - 1) + 2.$$

Therefore we can prove the second part of the base case by the following inequalities

$$H_2^*(x^f, x^q, x^\ell - 1, x^s) + 2 - H_2^*(x^f + 1, x^q, x^\ell, x^s - 1) \geq$$
$$\frac{b(x^s + x^\ell - 1 + u^L(i))}{x^s + x^\ell - 1 + u^L(i)}(x^\ell + u^L(i) - 1) + 2 - \frac{b(x^s - 1 + x^\ell + u^L(i))}{x^s - 1 + x^\ell + u^L(i)}(x^\ell + u^L(i) + 1)$$
$$= -2\frac{b(x^s + x^\ell - 1 + u^L(i))}{x^s + x^\ell - 1 + u^L(i)} + 2 \geq 0$$

considered that $b(z)/z \leq 1$.

Similarly, if $i = (x^f, x^q, x^\ell, x^s - 1)$ and $j = (x^f, x^q + 1, x^\ell - 2, x^s + 1)$

$$u^L(i) = \min(x^f, x^q, x^D - x^\ell - x^s + 1)$$

and

$$u^L(j) = \min(x^f, x^q + 1, x^D - x^\ell - x^s + 1),$$

and consequently

$$u^L(j) \geq u^L(i).$$

Furthermore

$$H_2^*(x^f, x^q, x^\ell, x^s - 1) = H_2^*(i) = \frac{b(x^s + x^\ell - 1 + u^L(i))}{x^s + x^\ell - 1 + u^L(i)}(x^\ell + u^L(i)),$$

while

$$H_2^*(x^f, x^q + 1, x^\ell - 2, x^s + 1) + 2 = H_2^*(j) + 2 =$$
$$= \frac{b(x^s + x^\ell - 1 + u^L(j))}{x^s + x^\ell - 1 + u^L(j)}(x^\ell - 2 + u^L(j)) + 2 \geq$$
$$\frac{b(x^s + x^\ell - 1 + u^L(i))}{x^s + x^\ell - 1 + u^L(i)}(x^\ell - 2 + u^L(i)) + 2.$$

Hence

$$H_2^*(x^f, x^q+1, x^\ell-2, x^s+1)+2-H_2^*(x^f, x^q, x^\ell, x^s-1) \geq$$
$$\frac{b(x^s+x^\ell-1+u^L(i))}{x^s+x^\ell-1+u^L(i)}(x^\ell-2+u^L(i))+2-\frac{b(x^s+x^\ell-1+u^L(i))}{x^s+x^\ell-1+u^L(i)}(x^\ell+u^L(i))$$
$$=-2\frac{b(x^s+x^\ell-1+u^L(i))}{x^s+x^\ell-1+u^L(i)}+2.$$

Since we know that

$$-2\frac{b(x^s+x^\ell-1+u^L(i))}{x^s+x^\ell-1+u^L(i)}+2\geq 0$$

then

$$H_2^*(x^f, x^q+1, x^\ell-2, x^s+1)+2-H_2^*(x^f, x^q, x^\ell, x^s-1)\geq 0.$$

It remains to prove that for every policy π, there exists π' such that

$$H_{\pi',\ 2}(x^f-1, x^q-1, x^\ell+1, x^s-1) \geq H_{\pi,\ 2}(x^f, x^q, x^\ell, x^s).$$

In particular we can choose π' as the optimal control policy π^*, hence if

$$H_2^*(x^f-1, x^q-1, x^\ell+1, x^s-1) \geq H_2^*(x^f, x^q, x^\ell, x^s)$$

then

$$H_{\pi',\ 2}(x^f-1, x^q-1, x^\ell+1, x^s-1) = H_2^*(x^f-1, x^q-1, x^\ell+1, x^s-1)$$
$$\geq H_2^*(x^f, x^q, x^\ell, x^s) \geq H_{\pi,\ 2}(x^f, x^q, x^\ell, x^s),$$

whichever is π. If we put $i=(x^f-1, x^q-1, x^\ell+1, x^s-1)$ then

$$H_2^*(x^f-1, x^q-1, x^\ell+1, x^s-1)=\frac{b(x^s+x^\ell+u^L(i))}{x^s+x^\ell+u^L(i)}(x^\ell+1+u^L(i)).$$

If $j=(x^f, x^q, x^\ell, x^s)$ then $u^L(j)\leq u^L(i)+1$, hence

$$H_{\pi,\ 2}(x^f, x^q, x^\ell, x^s)=\frac{b(x^s+x^\ell+u^L(j))}{x^s+x^\ell+u^L(j)}(x^\ell+u^L(j))$$
$$\leq \frac{b(x^s+x^\ell+1+u^L(i))}{x^s+x^\ell+1+u^L(i)}(x^\ell+u^L(i)+1)$$
$$\leq \frac{b(x^s+x^\ell+u^L(i))}{x^s+x^\ell+u^L(i)}(x^\ell+u^L(i)+1)=H_2^*(x^f-1, x^q-1, x^\ell+1, x^s-1)$$

that concludes the proof for the base case of induction.

As far as the inductive step is concerned, let's consider a generic state $i = (x^f, x^q, x^\ell, x^s)$ and a generic control policy π such that the controllable inputs for state i are

$$\pi(i, N) = (u^\ell(i, N), u^s(i, N))$$

for a control horizon of N steps. Thus if it is in the form $[0, 1, \ldots, N-1]$, then $(u^\ell(i, N), u^s(i, N))$ are the controls used by the policy π at time 0. We will refer to this kind of time interval hereinafter, without any loss of generality considered that the dynamical system is time-invariant.
We wish to prove that if $(u^\ell(i, N) + 1, u^s(i, N) - 1)$ is an admissible control, then there exists a new policy π' that performs better, in the sense that

$$H_{\pi',\ N}(i) > H_{\pi,\ N}(i)$$

and consequently π is not optimal.
Since for the optimal substructure property an optimal policy over N steps is also optimal over the remaining $N - 1$ steps, we can assume that

$$H_{\pi,\ N-1}(x^f, x^q, x^\ell - 1, x^s) + (N - 1) \geq H_{\pi,\ N-1}(x^f + 1, x^q, x^\ell, x^s - 1)$$

because otherwise π could not be optimal.
With π as a control policy and i as initial state, considered the transition equations, the newly reached state has the following form:

$$(x_1|x_0 = i, \pi) = (x^f - u^\ell(i, N) + w_0^s, x^q - u^\ell(i, N) + w_0^\phi, x^\ell + u^\ell(i, N) - w_0^\ell, x^s + u^s(i, N) - w_0^s).$$

We choose π' such that the controllable inputs in state i are

$$\pi'(i, N) = (u^\ell(i, N) + 1, u^s(i, N) - 1) = (u^\ell + 1, u^s - 1)$$

hence

$$(x_1|x_0 = i, \pi') =$$

$$(x^f - u^\ell - 1 + \overline{w}_0^s, x^q - u^\ell - 1 + \overline{w}_0^\phi, x^\ell + u^\ell + 1 - \overline{w}_0^\ell, x^s + u^s - 1 - \overline{w}_0^s).$$

While comparing the two policies, it's possible to note that $\overline{w}_0^\phi = w_0^\phi$ since it is an exogenous input independent from anything else in the system, so they both have the same probability distribution. Moreover, considered its independence from the other stochastic variables, the joint probability distribution can be factorized.

By the means of Bellman equation:

$$H_{\pi,\ N}(i) = E_{w_0^\phi}[(H_{\pi,\ N-1}(i_\pi^\ell) + (N-1))P[w_0^s = 0, w_0^\ell = 1] +$$
$$H_{\pi,\ N-1}(i_\pi^s)P[w_0^s = 1, w_0^\ell = 0] + H_{\pi,\ N-1}(i_\pi^N)P[w_0^s = 0, w_0^\ell = 0]],$$

where

$$(i_\pi^\ell) = (x^f - u^\ell, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell - 1, x^s + u^s),$$
$$(i_\pi^s) = (x^f - u^\ell + 1, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell, x^s + u^s - 1),$$
$$(i_\pi^N) = (x^f - u^\ell, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell, x^s + u^s),$$

and $E_{w_0^\phi}[\cdot]$ stands for an expectation taken using the probability distribution of w_0^ϕ.

With π' as a control policy it becomes

$$H_{\pi',\ N}(i) = E_{w_0^\phi}[(H_{\pi',\ N-1}(i_{\pi'}^\ell) + (N-1))P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] +$$
$$H_{\pi',\ N-1}(i_{\pi'}^s)P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] + H_{\pi',\ N-1}(i_{\pi'}^N)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0]],$$

where in this case

$$(i_{\pi'}^\ell) = (x^f - u^\ell - 1, x^q - u^\ell - 1 + w_0^\phi, x^\ell + u^\ell + 1 - 1, x^s + u^s + 1),$$
$$(i_{\pi'}^s) = (x^f - u^\ell - 1 + 1, x^q - u^\ell - 1 + w_0^\phi, x^\ell + u^\ell + 1, x^s + u^s + 1 - 1),$$
$$(i_{\pi'}^N) = (x^f - u^\ell - 1, x^q - u^\ell - 1 + w_0^\phi, x^\ell + u^\ell + 1, x^s + u^s + 1).$$

For the inductive hypothesis we know that $\forall\ \pi$, there exists π' such that $H_{\pi',\ N-1}(x^f - 1, x^q - 1, x^\ell + 1, x^s - 1) \geq H_{\pi,\ N-1}(x^f, x^q, x^\ell, x^s)$, thus we can choose π' such that

$$H_{\pi',\ N-1}(i_{\pi'}^\ell) \geq H_{\pi,\ N-1}(i_\pi^\ell),$$
$$H_{\pi',\ N-1}(i_{\pi'}^s) \geq H_{\pi,\ N-1}(i_\pi^s),$$
$$H_{\pi',\ N-1}(i_{\pi'}^N) \geq H_{\pi,\ N-1}(i_\pi^N).$$

Therefore

$$H_{\pi',\ N}(i) \geq (H_{\pi,\ N-1}(i_\pi^\ell) + (N-1))P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] +$$
$$H_{\pi,\ N-1}(i_\pi^s)P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] + H_{\pi,\ N-1}(i_\pi^N)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0].$$

As far as the probabilities are concerned, the following inequalities hold:

$$P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] > P[w_0^s = 0, w_0^\ell = 1],$$
$$P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] < P[w_0^s = 1, w_0^\ell = 0],$$
$$P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0] = P[w_0^s = 0, w_0^\ell = 0].$$

At this point, since we know that $(H_{\pi,\ N-1}(i_\pi^\ell) + (N-1)) > H_{\pi,\ N-1}(i_\pi^s)$, it's easy to see that

$$H_{\pi',\ N}(i) > H_{\pi,\ N}(i),$$

whichever is the probability distribution of w_0^ϕ.

Let's now consider a generic state $i = (x^f, x^q, x^\ell, x^s)$ and a generic control policy π such that the controllable inputs for state i are

$$\pi(i, N) = (u^\ell(i, N), u^s(i, N)).$$

We will prove that if $(u^\ell(i, N) + 1, u^s(i, N))$ is an admissible control, then there exists a new policy π' that performs better, in the sense that

$$H_{\pi',\ N}(i) > H_{\pi,\ N}(i)$$

and hence π is not optimal.
With π as a control policy and i as initial state, considered the transition equations, the newly reached state has the following form:

$$(x_1|x_0 = i, \pi) = (x^f - u^\ell(i, N) + w_0^s, x^q - u^\ell(i, N) + w_0^\phi, x^\ell + u^\ell(i, N) - w_0^\ell, x^s + u^s(i, N) - w_0^s).$$

We choose π' such that the controllable inputs in state i when there are $N-1$ steps left are

$$(u^\ell(i,N)+1, u^s(i,N)) = (u^\ell+1, u^s),$$

and consequently

$$(x_1|x_0 = i, \pi') =$$
$$(x^f - u^\ell - 1 + \overline{w}_0^s, x^q - u^\ell - 1 + \overline{w}_0^\phi, x^\ell + u^\ell + 1 - \overline{w}_0^\ell, x^s + u^s - \overline{w}_0^s).$$

In this case

$$(i^\ell_{\pi'}) = (x^f - u^\ell - 1, x^q - u^\ell - 1 + w_0^\phi, x^\ell + u^\ell + 1 - 1, x^s + u^s),$$
$$(i^s_{\pi'}) = (x^f - u^\ell - 1 + 1, x^q - u^\ell - 1 + w_0^\phi, x^\ell + u^\ell + 1, x^s + u^s - 1),$$
$$(i^N_{\pi'}) = (x^f - u^\ell - 1, x^q - u^\ell - 1 + w_0^\phi, x^\ell + u^\ell + 1, x^s + u^s).$$

For the inductive hypothesis we know that $\forall\ \pi$, there exists π' such that $H_{\pi',\ N-1}(x^f - 1, x^q - 1, x^\ell + 1, x^s) \geq H_{\pi,\ N-1}(x^f, x^q, x^\ell, x^s)$, hence we can choose π' such that

$$H_{\pi',\ N-1}(i^\ell_{\pi'}) \geq H_{\pi,\ N-1}(i^\ell_\pi),$$
$$H_{\pi',\ N-1}(i^s_{\pi'}) \geq H_{\pi,\ N-1}(i^s_\pi),$$
$$H_{\pi',\ N-1}(i^N_{\pi'}) \geq H_{\pi,\ N-1}(i^N_\pi).$$

Therefore

$$H_{\pi',\ N}(i) \geq (H_{\pi,\ N-1}(i^\ell_\pi) + (N-1))P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] +$$
$$H_{\pi,\ N-1}(i^s_\pi)P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] + H_{\pi,\ N-1}(i^N_\pi)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0].$$

As far as the probabilities are concerned, the following inequalities hold:

$$P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] > P[w_0^s = 0, w_0^\ell = 1],$$
$$P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] < P[w_0^s = 1, w_0^\ell = 0],$$
$$P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0] = P[w_0^s = 0, w_0^\ell = 0].$$

At this point, since we know that $(H_{\pi,\ N-1}(i^\ell_\pi) + (N-1)) > H_{\pi,\ N-1}(i^s_\pi)$, it's easy to see that

$$H_{\pi',\ N}(i) > H_{\pi,\ N}(i),$$

whichever is the probability distribution of w_0^ϕ.
Therefore we have proved that any control policy π such that it does not issue the maximum number of load operations possible for very state i is not optimal. Consequently

$$\widetilde{u}_{opt}^{\ell}(i,N) = u^L(i) \ \forall \ i.$$

Now let's consider a generic state $j = (x^f - 1, x^q - 1, x^\ell + 1, x^s - 1)$ and a generic control policy π; we need to show that there exists π' such that $H_{\pi',\ N}(x^f - 1, x^q - 1, x^\ell + 1, x^s - 1) \geq H_{\pi,\ N}(x^f, x^q, x^\ell, x^s)$.
If we put $i = (x^f, x^q, x^\ell, x^s)$, let $(u^\ell(i,N), u^s(i,N))$ be the controllable inputs chosen by the policy π.

$$u^\ell(i,N) \leq u^L(i) = \min\left(x^q, x^f, x^D - x^\ell - x^s\right),$$

$$u^s(i,N) \leq u^S(i) = \min\left(x^F - x^s - x^f, x^D - x^\ell - x^s - u^\ell(i,N)\right),$$

$$u^L(j) = \min\left(x^q - 1, x^f - 1, x^D - x^\ell - x^s\right) \geq u^L(i) - 1.$$

Now if $u^\ell(i,N) = u^\ell \neq 0$ then $u^\ell(i,N) - 1$ is an admissible control for state j. In that case

$$u^S(j) = \min\left(x^F - x^s - x^f + 2, x^D - x^\ell - x^s - u^\ell(i,N) + 1\right) \geq u^S(i) + 1.$$

Therefore we can choose π' such that the controllable inputs for state j are

$$(u^\ell(i,N) - 1, u^s(i,N) + 1).$$

With this choice,

$$(x_1|x_0 = i, \pi)$$
$$= (x^f - u^\ell(i,N) + w_0^s, x^q - u^\ell(i,N) + w_0^\phi, x^\ell + u^\ell(i,N) - w_0^\ell, x^s + u^s(i,N) - w_0^s),$$

$$(x_1|x_0 = j, \pi') = (x_1|x_0 = i, \pi)$$
$$= (x^f - u^\ell(i,N) + w_0^s, x^q - u^\ell(i,N) + w_0^\phi, x^\ell + u^\ell(i,N) - w_0^\ell, x^s + u^s(i,N) - w_0^s).$$

Therefore if we choose $\pi'(x,k) = \pi(x,k)\ \forall\ k = 1,\dots,N-1$ we get

$$H_{\pi',\ N}(x^f - 1, x^q - 1, x^\ell + 1, x^s - 1) = H_{\pi,\ N}(x^f, x^q, x^\ell, x^s).$$

Otherwise if $u^\ell(i,N) = u^\ell = 0$, we can choose π' such that the controllable inputs for state j are

$$(0, u^s(i,N)).$$

With this choice,

$$\begin{aligned}(x_1|x_0 = i, \pi) &= (x^f + w_0^s, x^q + w_0^\phi, x^\ell - w_0^\ell, x^s + u^s(i,N) - w_0^s),\\ (x_1|x_0 = j, \pi') &= (x^f - 1 + w_0^s, x^q - 1 + w_0^\phi, x^\ell + 1 - w_0^\ell, x^s - 1 + u^s(i,N) - w_0^s).\end{aligned}$$

At this point we can use the same argument used before to show that $H_{\pi',\ N}(x^f - 1, x^q - 1, x^\ell + 1, x^s - 1) > H_{\pi,\ N}(x^f, x^q, x^\ell, x^s)$. In fact if we put

$$\begin{aligned}(i^\ell_{\pi'}) &= (x^f - 1, x^q - 1 + w_0^\phi, x^\ell + 1 - 1, x^s - 1 + u^s),\\ (i^s_{\pi'}) &= (x^f - 1 + 1, x^q - 1 + w_0^\phi, x^\ell + 1, x^s - 1 + u^s - 1),\\ (i^N_{\pi'}) &= (x^f - 1, x^q - 1 + w_0^\phi, x^\ell + 1, x^s - 1 + u^s),\end{aligned}$$

and

$$\begin{aligned}(i^\ell_{\pi}) &= (x^f, x^q + w_0^\phi, x^\ell - 1, x^s + u^s),\\ (i^s_{\pi}) &= (x^f + 1, x^q + w_0^\phi, x^\ell, x^s + u^s - 1),\\ (i^N_{\pi}) &= (x^f, x^q + w_0^\phi, x^\ell, x^s + u^s),\end{aligned}$$

we can use exactly the same argument.

A very similar argument can be developed to show that for every admissible policy π, there exists π' such that $H_{\pi',\ N}(x^f - 1, x^q - 1, x^\ell + 1, x^s) \geq H_{\pi,\ N}(x^f, x^q, x^\ell, x^s)$. If we put $i = (x^f, x^q, x^\ell, x^s)$, let $\pi(i,N) = (u^\ell(i,N), u^s(i,N))$ be the controllable inputs chosen by the policy π.

$$\begin{aligned}u^\ell(i,N) \leq u^L(i) &= \min\left(x^q, x^f, x^D - x^\ell - x^s\right),\\ u^s(i,N) \leq u^S(i) &= \min\left(x^F - x^s - x^f, x^D - x^\ell - x^s - u^\ell(i,N)\right).\end{aligned}$$

Let's call $j = (x^f - 1, x^q - 1, x^\ell + 1, x^s)$, then

$$u^L(j) = \min\left(x^q - 1, x^f - 1, x^D - x^\ell - x^s - 1\right) = u^L(i) - 1.$$

Now if $u^\ell(i,N) = u^\ell \neq 0$ then $u^\ell(i,N) - 1$ is an admissible control for state j. In that case

$$u^S(j) = \min\left(x^F - x^s - x^f + 1, x^D - x^\ell - x^s - u^\ell(i,N)\right) \geq u^S(i).$$

Therefore we can choose π' such that the controllable inputs for state j are

$$(u^\ell(i,N) - 1, u^s(i,N)).$$

With this choice,

$$(x_1|x_0 = i, \pi) = (x^f - u^\ell(i,N) + w_0^s, x^q - u^\ell(i,N) + w_0^\phi, x^\ell + u^\ell(i,N) - w_0^\ell, x^s + u^s(i,N) - w_0^s),$$

$$(x_1|x_0 = j, \pi') = (x_1|x_0 = i, \pi)$$
$$(x^f - u^\ell(i,N) + w_0^s, x^q - u^\ell(i,N) + w_0^\phi, x^\ell + u^\ell(i,N) - w_0^\ell, x^s + u^s(i,N) - w_0^s).$$

Therefore if we choose $\pi'(x,k) = \pi(x,k)\ \forall\quad k = 1, \ldots, N-1$ we get $H_{\pi',\ N}(x^f - 1, x^q - 1, x^\ell + 1, x^s - 1) = H_{\pi,\ N}(x^f, x^q, x^\ell, x^s)$.
The case $u^\ell(i,N) = 0$ is solved by putting the controllable inputs for state j

$$(0, u^s(i,N))$$

and the using the same argument developed for the previous case.

It remains to prove that

$$H_N^*(x^f, x^q, x^\ell - 1, x^s) + N > H_N^*(x^f + 1, x^q, x^\ell, x^s - 1).$$

If we put $i = (x^f + 1, x^q, x^\ell, x^s - 1)$, let $\pi^*(i,N) = (u^\ell(i,N), u^s(i,N)) = (u^L(i), u^s(i,N)) = (u^\ell, u^s)$ be the controllable inputs chosen by the optimal

control policy π^*.

$$u^L(i) = \min(x^q, x^f + 1, x^D - x^\ell - x^s + 1),$$
$$u^s(i,N) \leq u^S(i) = \min(x^F - x^s - x^f, x^D - x^\ell - x^s - u^L(i) + 1).$$

Let's call $j = (x^f, x^q, x^\ell - 1, x^s)$; then

$$u^L(j) = \min(x^q, x^f, x^D - x^\ell - x^s + 1).$$

There are two possible cases: $u^L(j) = u^L(i)$ and $u^L(j) = u^L(i) - 1$ and we will consider them separately.
If $u^L(j) = u^L(i)$ then $(u^L(i), u^s(i,N))$ is an admissible set of controls for state j. In fact

$$u^S(j) = \min(x^F - x^s - x^f, x^D - x^\ell - x^s - u^L(i) + 1) = u^S(i).$$

We define a control policy π' such that the controllable inputs in j are exactly $(u^L(i), u^s(i,N))$. With this choice,

$$(x_1|x_0 = i, \pi^*) =$$
$$(x^f + 1 - u^\ell(i,N) + w_0^s, x^q - u^\ell(i,N) + w_0^\phi, x^\ell + u^\ell(i,N) - w_0^\ell, x^s - 1 + u^s(i,N) - w_0^s),$$

while

$$(x_1|x_0 = j, \pi') =$$
$$(x^f - u^\ell(i,N) + w_0^s, x^q - u^\ell(i,N) + w_0^\phi, x^\ell - 1 + u^\ell(i,N) - w_0^\ell, x^s + u^s(i,N) - w_0^s).$$

By the means of Bellman equation:

$$H_{\pi^*,\ N}(i) = E_{w_0^\phi}[(H_{\pi^*,\ N-1}(i_\pi^\ell) + (N-1))P[w_0^s = 0, w_0^\ell = 1] +$$
$$H_{\pi^*,\ N-1}(i_\pi^s)P[w_0^s = 1, w_0^\ell = 0] + H_{\pi^*,\ N-1}(i_\pi^N)P[w_0^s = 0, w_0^\ell = 0]],$$

where

$$(i_\pi^\ell) = (x^f + 1 - u^\ell, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell - 1, x^s - 1 + u^s),$$
$$(i_\pi^s) = (x^f + 1 - u^\ell + 1, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell, x^s - 1 + u^s - 1),$$
$$(i_\pi^N) = (x^f + 1 - u^\ell, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell, x^s - 1 + u^s).$$

Regarding π'

$$H_{\pi',\ N}(j) = E_{w_0^\phi}[(H_{\pi',\ N-1}(j^\ell_{\pi'}) + (N-1))P[\overline{w}^s_0 = 0, \overline{w}^\ell_0 = 1] +$$
$$H_{\pi',\ N-1}(j^s_{\pi'})P[\overline{w}^s_0 = 1, \overline{w}^\ell_0 = 0] + H_{\pi',\ N-1}(j^N_{\pi'})P[\overline{w}^s_0 = 0, \overline{w}^\ell_0 = 0]],$$

where in this case

$$(j^\ell_{\pi'}) = (x^f - u^\ell, x^q - u^\ell + w^\phi_0, x^\ell + u^\ell - 1 - 1, x^s + u^s),$$
$$(j^s_{\pi'}) = (x^f - u^\ell + 1, x^q - u^\ell + w^\phi_0, x^\ell + u^\ell - 1, x^s + u^s - 1),$$
$$(j^N_{\pi'}) = (x^f - u^\ell, x^q - u^\ell + w^\phi_0, x^\ell + u^\ell - 1, x^s + u^s).$$

We can also write

$$H_{\pi',\ N}(j) + N = E_{w_0^\phi}[(H_{\pi',\ N-1}(j^\ell_{\pi'}) + (N-1) + N)P[\overline{w}^s_0 = 0, \overline{w}^\ell_0 = 1] +$$
$$(H_{\pi',\ N-1}(j^s_{\pi'}) + N)P[\overline{w}^s_0 = 1, \overline{w}^\ell_0 = 0] + (H_{\pi',\ N-1}(j^N_{\pi'}) + N)P[\overline{w}^s_0 = 0, \overline{w}^\ell_0 = 0]].$$

If we put $\pi'(x,k) = \pi^*(x,k) \quad \forall \ k \leq N-1$, then for the inductive hypothesis

$$H_{\pi',\ N-1}(j^N_{\pi'}) + (N-1) = H_{\pi^*,\ N-1}(j^N_{\pi'}) + (N-1) \geq H_{\pi^*,\ N-1}(i^N_\pi).$$

Moreover

$$H_{\pi',\ N-1}(j^N_{\pi'}) + (N) = H_{\pi^*,\ N-1}(j^N_{\pi'}) + (N) \geq H_{\pi^*,\ N-1}(i^N_\pi).$$

It's also possible to see that $P[\overline{w}^s_0 = 0, \overline{w}^\ell_0 = 0] = P[w^s_0 = 0, w^\ell_0 = 0]$.
Hence if

$$(H_{\pi',\ N-1}(j^\ell_{\pi'}) + (N-1) + N)P[\overline{w}^s_0 = 0, \overline{w}^\ell_0 = 1] + (H_{\pi',\ N-1}(j^s_{\pi'}) + N)P[\overline{w}^s_0 = 1, \overline{w}^\ell_0 = 0]$$
$$> \tag{5.13}$$
$$(H_{\pi^*,\ N-1}(i^\ell_\pi) + (N-1))P[w^s_0 = 0, w^\ell_0 = 1] + H_{\pi^*,\ N-1}(i^s_\pi)P[w^s_0 = 1, w^\ell_0 = 0],$$

then

$$H_{\pi',\ N}(j) + N > H_{\pi^*,\ N}(i).$$

Considered that

$$(j^s_{\pi'}) = (i^\ell_\pi),$$

the following inequality holds

$$(H_{\pi',\ N-1}(j^s_{\pi'}) + N) > (H_{\pi^*,\ N-1}(i^\ell_\pi) + (N-1)),$$

but also from the inductive hypothesis

$$(H_{\pi',\ N-1}(j^\ell_{\pi'}) + (N-1) + N) > (H_{\pi',\ N-1}(j^s_{\pi'}) + N) > \\ (H_{\pi^*,\ N-1}(i^\ell_\pi) + (N-1)) > H_{\pi^*,\ N-1}(i^s_\pi).$$

Therefore whichever are the probabilities $P[\overline{w}^s_0 = 0, \overline{w}^\ell_0 = 1]$ and $P[\overline{w}^s_0 = 1, \overline{w}^\ell_0 = 0]$, inequality (5.13) is satisfied.

Considered that by definition of the optimal control policy $H_{\pi^*,\ N}(j) \geq H_{\pi',\ N}(j)$, we have that

$$H_{\pi^*,\ N}(j) + N \geq H_{\pi',\ N}(j) + N > H_{\pi^*,\ N}(i),$$

which concludes this case of the proof.

If

$$\min(x^q, x^f + 1, x^D - x^\ell - x^s + 1) = x^f + 1 = u^L(i),$$

then $u^L(j) = u^L(i) - 1$. In this case $(u^L(i) - 1, u^s(i, N) + 1)$ is an admissible set of controls for state j. In fact if we assume x^F is large enough ($x^F > 2(x^D + 1)$ suffices), then

$$u^S(j) = \min(x^F - x^s - x^f, x^D - x^\ell - x^s - u^L(i) + 2) = \\ x^D - x^\ell - x^s - u^L(i) + 2 = u^S(i) + 1.$$

In our model, $u^L(j) = u^L(i) - 1$ only if x^f is small, hence if x^F is large enough u^S is imposed by the maximum length x^D of the disk's queue, not by x^F. We define a control policy π' such that the controllable inputs in j are exactly $(u^L(i) - 1, u^s(i, N) + 1)$. With this choice,

$$(x_1 | x_0 = i, \pi) = (w^s_0, x^q - u^L(i) + w^\phi_0, x^\ell + u^L(i) - w^\ell_0, x^s + u^s(i, N) - 1 - w^s_0),$$

$$(x_1|x_0 = j, \pi') =$$

$$(w_0^s, x^q - u^L(i) + 1 + w_0^\phi, x^\ell - 1 + u^L(i) - 1 - w_0^\ell, x^s + u^s(i, N) + 1 - w_0^s).$$

By the means of Bellman equation:

$$H_{\pi^*,\ N}(i) = E_{w_0^\phi}[(H_{\pi^*,\ N-1}(i_\pi^\ell) + (N-1))P[w_0^s = 0, w_0^\ell = 1] +$$
$$H_{\pi^*,\ N-1}(i_\pi^s)P[w_0^s = 1, w_0^\ell = 0] + H_{\pi^*,\ N-1}(i_\pi^N)P[w_0^s = 0, w_0^\ell = 0]],$$

where

$$(i_\pi^\ell) = (0, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell - 1, x^s - 1 + u^s),$$
$$(i_\pi^s) = (1, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell, x^s - 1 + u^s - 1),$$
$$(i_\pi^N) = (0, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell, x^s - 1 + u^s).$$

The values of the index H with π' is

$$H_{\pi',\ N}(j) = E_{w_0^\phi}[(H_{\pi',\ N-1}(j_{\pi'}^\ell) + (N-1))P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] +$$
$$H_{\pi',\ N-1}(j_{\pi'}^s)P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] + H_{\pi',\ N-1}(j_{\pi'}^N)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0]],$$

where in this case

$$(j_{\pi'}^\ell) = (0, x^q + 1 - u^\ell + w_0^\phi, x^\ell - 1 + u^\ell - 1 - 1, x^s + u^s + 1),$$
$$(j_{\pi'}^s) = (1, x^q + 1 - u^\ell + w_0^\phi, x^\ell - 1 + u^\ell - 1, x^s + u^s + 1 - 1),$$
$$(j_{\pi'}^N) = (0, x^q + 1 - u^\ell + w_0^\phi, x^\ell - 1 + u^\ell - 1, x^s + u^s + 1).$$

We can also write

$$H_{\pi',\ N}(j) + N = E_{w_0^\phi}[(H_{\pi',\ N-1}(j_{\pi'}^\ell) + (N-1) + N)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] +$$
$$(H_{\pi',\ N-1}(j_{\pi'}^s) + N)P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] + (H_{\pi',\ N-1}(j_{\pi'}^N) + N)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0]].$$

We choose $\pi'(x, k) = \pi^*(x, k) \quad \forall \ \ k \leq N - 1$. For the inductive hypothesis

$$H_{\pi',\ N-1}(j_{\pi'}^N) + (N-1) = H_{\pi^*,\ N-1}(j_{\pi'}^N) + (N-1) \geq H_{\pi^*,\ N-1}(i_\pi^N),$$

hence it obviously follows

$$H_{\pi',\ N-1}(j_{\pi'}^N) + (N) = H_{\pi^*,\ N-1}(j_{\pi'}^N) + (N) \geq H_{\pi^*,\ N-1}(i_\pi^N).$$

It's also possible to see that $P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0] = P[w_0^s = 0, w_0^\ell = 0]$.
Now if

$$\begin{aligned}(H_{\pi',\ N-1}(j_{\pi'}^\ell) + (N-1) + N)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] + (H_{\pi',\ N-1}(j_{\pi'}^s) + N)P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] \\ > \\ (H_{\pi^*,\ N-1}(i_\pi^\ell) + (N-1))P[w_0^s = 0, w_0^\ell = 1] + H_{\pi^*,\ N-1}(i_\pi^s)P[w_0^s = 1, w_0^\ell = 0],\end{aligned} \tag{5.14}$$

then

$$H_{\pi',\ N}(j) + N > H_{\pi^*,\ N}(i).$$

Now let's consider the two states $j_{\pi'}^s$ and i_π^ℓ. We want to show that $H_{\pi^*,\ N-1}(j_{\pi'}^s) \geq H_{\pi^*,\ N-1}(i_\pi^\ell)$.

$$\pi^*(i_\pi^\ell, N-1) = (u^L(i_\pi^\ell), u^s) = (0, u^s),$$

while

$$\pi^*(j_{\pi'}^s, N-1) = (u^L(j_{\pi'}^s), \overline{u}^s) = (1, \overline{u}^s)$$

since a store has been performed (so there is at least one slot free in the disk queue) and $x^q + 1 - u^\ell + w_0^\phi \geq 1$. Since $x^F > 2(x^D + 1)$,

$$u^S(j_{\pi'}^s) \geq u^S(i_\pi^\ell) - 1$$

because u^S is imposed by the maximum length x^D of the disk queue (in fact x^f is small). As a consequence,

$$\pi''(j_{\pi'}^s, N-1) = (u^L(j_{\pi'}^s), u^s) = (1, u^s - 1)$$

is a valid policy. It's easy to see that with this choice

$$H_{\pi'',\ N-1}(j_{\pi'}^s) = H_{\pi^*,\ N-1}(i_\pi^\ell),$$

$$H_{\pi^*,\ N-1}(j_{\pi'}^s) \geq H_{\pi'',\ N-1}(j_{\pi'}^s) = H_{\pi^*,\ N-1}(i_\pi^\ell),$$

hence

$$(H_{\pi',\ N-1}(j_{\pi'}^s) + N) > (H_{\pi^*,\ N-1}(i_\pi^\ell) + (N-1))$$

but also from the inductive hypothesis

$$(H_{\pi',\ N-1}(j^{\ell}_{\pi'}) + (N-1) + N) > (H_{\pi',\ N-1}(j^{s}_{\pi'}) + N) > \\ (H_{\pi^*,\ N-1}(i^{\ell}_{\pi}) + (N-1)) > H_{\pi^*,\ N-1}(i^{s}_{\pi}).$$

Therefore whichever are the probabilities $P[\overline{w}^s_0 = 0, \overline{w}^\ell_0 = 1]$ and $P[\overline{w}^s_0 = 1, \overline{w}^\ell_0 = 0]$, inequality (5.14) is satisfied. Considered that by definition of the optimal control policy $H_{\pi^*,\ N}(j) \geq H_{\pi',\ N}(j)$, it follows that

$$H_{\pi^*,\ N}(j) + N \geq H_{\pi',\ N}(j) + N > H_{\pi^*,\ N}(i),$$

which concludes this case of the proof.

The last property to prove is

$$H^*_N(0, x^q + 1, x^\ell - 2, x^s + 1) + N \geq H^*_N(0, x^q, x^\ell, x^s - 1).$$

Let's call $i = (0, x^q, x^\ell, x^s - 1)$ and $j = (0, x^q + 1, x^\ell - 2, x^s + 1)$. The two states have the same set of admissible controls: in fact $u^L(i) = u^L(j) = 0$ and, considered that $x^f = 0$ and $x^F > 2(x^D + 1)$ we have again that the limitation on $u^S(i)$ is imposed by the queue of requests seen by the disk I/O subsystem.
If $\pi^*(i, N) = (0, u^s)$ then $(0, u^s)$ is a valid set of controls also for state j. With this choice,

$$(x_1 | x_0 = i, \pi) = (w^s_0, x^q + w^\phi_0, x^\ell - w^\ell_0, x^s + u^s - 1 - w^s_0),$$
$$(x_1 | x_0 = j, \pi') = (w^s_0, x^q + 1 + w^\phi_0, x^\ell - 2 - w^\ell_0, x^s + u^s + 1 - w^s_0).$$

By the means of Bellman equation:

$$H_{\pi^*,\ N}(i) = E_{w^\phi_0}[(H_{\pi^*,\ N-1}(i^\ell_\pi) + (N-1))P[w^s_0 = 0, w^\ell_0 = 1] + \\ H_{\pi^*,\ N-1}(i^s_\pi)P[w^s_0 = 1, w^\ell_0 = 0] + H_{\pi^*,\ N-1}(i^N_\pi)P[w^s_0 = 0, w^\ell_0 = 0]],$$

where

$$(i_\pi^\ell) = (0, x^q + w_0^\phi, x^\ell - 1, x^s - 1 + u^s),$$
$$(i_\pi^s) = (1, x^q + w_0^\phi, x^\ell, x^s - 1 + u^s - 1),$$
$$(i_\pi^N) = (0, x^q + w_0^\phi, x^\ell, x^s - 1 + u^s).$$

The value of $H_{\pi',\ N}(j)$ is

$$H_{\pi',\ N}(j) = E_{w_0^\phi}[(H_{\pi',\ N-1}(j_{\pi'}^\ell) + (N-1))P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] +$$
$$H_{\pi',\ N-1}(j_{\pi'}^s)P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] + H_{\pi',\ N-1}(j_{\pi'}^N)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0]],$$

where in this case

$$(j_{\pi'}^\ell) = (0, x^q + 1 + w_0^\phi, x^\ell - 3, x^s + u^s + 1),$$
$$(j_{\pi'}^s) = (1, x^q + 1 + w_0^\phi, x^\ell - 2, x^s + u^s + 1 - 1),$$
$$(j_{\pi'}^N) = (0, x^q + 1 + w_0^\phi, x^\ell - 2, x^s + u^s + 1).$$

We can also write

$$H_{\pi',\ N}(j) + N = E_{w_0^\phi}[(H_{\pi',\ N-1}(j_{\pi'}^\ell) + (N-1) + N)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 1] +$$
$$(H_{\pi',\ N-1}(j_{\pi'}^s) + N)P[\overline{w}_0^s = 1, \overline{w}_0^\ell = 0] + (H_{\pi',\ N-1}(j_{\pi'}^N) + N)P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0]].$$

It's also possible to see that $P[\overline{w}_0^s = 0, \overline{w}_0^\ell = 0] = P[w_0^s = 0, w_0^\ell = 0]$.
We choose $\pi'(x,k) = \pi^*(x,k) \quad \forall \quad k \leq N-1$. For the inductive hypothesis

$$H_{\pi',\ N-1}(j_{\pi'}^N) + (N-1) = H_{\pi^*,\ N-1}(j_{\pi'}^N) + (N-1) \geq H_{\pi^*,\ N-1}(i_\pi^N),$$

hence it obviously follows

$$H_{\pi',\ N-1}(j_{\pi'}^N) + (N) = H_{\pi^*,\ N-1}(j_{\pi'}^N) + (N) \geq H_{\pi^*,\ N-1}(i_\pi^N).$$

Now let's consider the two states

$$(j_{\pi'}^s) = (1, x^q + 1 - u^\ell + w_0^\phi, x^\ell - 1 + u^\ell - 1, x^s + u^s + 1 - 1),$$
$$(i_\pi^\ell) = (0, x^q - u^\ell + w_0^\phi, x^\ell + u^\ell - 1, x^s - 1 + u^s).$$

We want to show that $H_{\pi^*,\ N-1}(j_{\pi'}^s) \geq H_{\pi^*,\ N-1}(i_\pi^\ell)$. For what we have just said,

$$\pi^*(i_\pi^\ell, N-1) = (u^L(i_\pi^\ell), u^s) = (0, u^s)$$

while

$$\pi^*(j^s_{\pi'}, N-1) = (u^L(j^s_{\pi'}), \overline{u}^s) = (1, \overline{u}^s).$$

In a way identical to the previously considered case, it is possible to define a policy π'' such that

$$H_{\pi'',\ N-1}(j^s_{\pi'}) = H_{\pi^*,\ N-1}(i^\ell_\pi),$$

$$H_{\pi^*,\ N-1}(j^s_{\pi'}) \geq H_{\pi'',\ N-1}(j^s_{\pi'}) = H_{\pi^*,\ N-1}(i^\ell_\pi).$$

We can therefore conclude our proof by using again the inductive hypothesis

$$\begin{aligned}(H_{\pi',\ N-1}(j^\ell_{\pi'}) + (N-1) + N) > (H_{\pi',\ N-1}(j^s_{\pi'}) + N) > \\ (H_{\pi^*,\ N-1}(i^\ell_\pi) + (N-1)) > H_{\pi^*,\ N-1}(i^s_\pi).\end{aligned}$$

Therefore whichever are the probabilities $P[\overline{w}^s_0 = 0, \overline{w}^\ell_0 = 1]$ and $P[\overline{w}^s_0 = 1, \overline{w}^\ell_0 = 0]$, we are able to compare $H_{\pi',\ N}(j)$ with $H_{\pi^*,\ N}(i)$. Considered that by definition of the optimal control policy $H_{\pi^*,\ N}(j) \geq H_{\pi',\ N}(j)$, we have that

$$H_{\pi^*,\ N}(j) + N \geq H_{\pi',\ N}(j) + N > H_{\pi^*,\ N}(i)$$

which concludes the proof.

In conclusion we have proved that if it is possible, it is always convenient to forward load operations to the disk subsystem as soon as possible. Even if the proof developed in the present section does not define the optimal policy completely, it provides some insights regarding store operations that will be discussed in the following section.

5.4 Considerations on the store policy

Although the proof presented in section 5.3 did not prove anything about the choice of $u^s_{opt}(x,k)$ (that is on the structure of the optimal policy regarding store operations), it provided some useful insights about it.

As we have seen in the previous section, one of the effects of increasing u^s is that it changes the probability distribution of the events related to the disk, namely

- event A, the disk performs a load operation
- event B, the disk performs a store operation
- event C, the disk performs no operation at all

In detail the probability of event B is increased, while those of events A and C are decreased. As we have seen in the proof, event A is always more desirable than event B, and it is possible to prove that event B is always better or equal than event C (it is pretty obvious). Since event B is not the best nor the worst event, it is not clear when the probability shift induced by increasing u^s is convenient, but it is presumable that it is so only when there are few I/O operations in queue. To be more precise, let's begin considering the case $u^s = 0$. When there are few requests in the disk I/O queue($y = x^\ell + x^s + u^\ell$ is small), event C (which is always the worst) has an high probability equal to $1 - b(x^\ell + x^s + u^\ell)$. To understand quantitatively what is the effect of increasing u^s we note that the probabilities of events C and A as functions of u^s are respectively

$$P(C, u^s) = 1 - b(y + u^s)$$

and

$$P(A, u^s) = \alpha\ b(y + u^s)/(y + u^s).$$

As we have just said, they are both decreasing functions of u^s, but when y is small $P(C, u^s)$ decreases faster than $P(A, u^s)$, making the probability shift more likely to be convenient.

Obviously it also depends on how much event B is better than event C. For example, at the second last step they are exactly equivalent (having freed a frame for the last step is useless), hence the shift is never convenient. As a

general rule, the shift becomes less and less convenient as we get closer to the end of the control horizon.
This fact emphasizes the difficulty of tackling the time-dependent problem, because in this case the optimal store control law does actually depend on the time argument. In this context, finding a time-dependent closed form control law appears extremely hard, thus we should probably concentrate on stationary control policies.
Considered that the optimal control policy for the control horizon T is in the form:

$$\pi^* = \{\mu_k, k \in T\},$$

where

$$\mu_k(x) = (u^{\ell}_{opt}(x,k), u^{s}_{opt}(x,k)) \qquad k \in T = [0, \ldots, N-1],$$

at least in principle the optimal stationary policy can be obtained by the following formula

$$\mu_s(x) = \lim_{N\to\infty} \mu_0(x) = \lim_{N\to\infty} (u^L(x), u^s_{opt}(x,0)) = (u^L(x), \lim_{N\to\infty} u^s_{opt}(x,0)),$$

if the limit exists (which is not guaranteed). Even if the limit exists, this solution is not feasible, considered that we are not even able to find the non-stationary control law.
An alternative solution is to solve the average cost per step problem, hoping to find a stationary optimal solution, which, again, is not guaranteed. However in many cases of interest, as stated in [2], it is often the case. To that end, section 5.5 is dedicated to explore some regularity properties of the infinite horizon problem.

Even if we are not able to solve the problem mathematically, some further considerations can be made, in order to complete the heuristic reasoning previously developed.

In a stationary condition, when there is an infinite number of steps toward the end of the control horizon, the comparison between events A, B and C can be carried out without worrying about the time dependence. In other words, the comparison depends only on the current state, not on the number of steps left in the control horizon.
In this case a target policy

$$u^s(x) = \max\{T_g(x) - x^\ell - x^s - u^L(x), 0\} \tag{5.15}$$

is likely to be a good choice, if the target $T_g(x)$ is properly chosen. In this way the number of loads is imposed as $u^\ell = u^L$, then, if there are less requests than $T_g(x)$ in the disk queue, stores are added to reach the target value $T_g(x)$. Of course, $T_g(x)$ also represents a threshold on the length of the disk queue under which the probability shift is convenient. It is certainly a function of the state x, and it is likely to be some decreasing function of x^f. In fact when there are few free frames left, a successful store operation is more desirable because it draws away from the costly situation of an empty pool of free frames. Moreover, to the end of designing a simple feedback control law, it would be very interesting to understand if $T_g(x)$ should depend on the entire state x or only on the x^f component.
The obstacle that does not allow us to formalize the previous considerations into a rigorous proof is that u^s does not simply change the probability distribution of the events A, B, C but also influences the next reached state by incrementing x^s. In practice this means that once a store is added to the disk queue , it cannot be removed by the means of controllable inputs (u^s is supposed to be non-negative). If we assume to let the OS remove requests from the I/O queue once they are inserted (which is not so unrealistic), then a target policy could be proved to be optimal. Otherwise, it is not so easy to understand which is the effect of inserting requests than cannot be removed. However, perhaps it is possible to compensate this problem with a lower target (in respect to the one calculated when u^s can be negative). This

conjecture is confirmed by simulative results stated in [3], where *target policies* are reported to be semi-optimal in comparison with the solution found through dynamic programming.

5.5 The infinite horizon problem

In this section we will study the regularity properties of the infinite horizon problem, with reference to the case with a finite number of states. In other words we will consider a setting where the cardinality of the set S defined by equation 2.2 of the admissible states is finite, a property that holds if and only if x^Q is finite.
In the outlined setting, we will prove that the average cost per step $J_\pi(i)$ associated with the optimal control law π^* defined as follows:

$$J_{\pi^*}(i) = \limsup_{N\to\infty} \frac{1}{N} J_{\pi^*,\ N}(i)$$

is independent upon the initial state i.
We'll make use of the following Theorem, whose proof can be found in [1], Chapter 4:

Theorem 1 *If for every two states i and j there exists a stationary policy π (depending on i and j) such that, for some k,*

$$P[x_k = j | x_0 = i, \pi] > 0,$$

then the optimal average cost per stage is the same value λ for all initial states i.

To prove the existence of π, we first consider a subset $Q \subset S$ of the admissible state space S defined as follows:

$$Q = \{(x^f, x^q, x^\ell, x^s) \in \mathcal{Z}^4 | 0 < x^f,\ 0 < x^q < x^Q,\ 0 < x^\ell,\ 0 < x^s \\ x^f + x^s < x^F,\ x^\ell + x^s < x^D, \}.$$

Considered the definition of the set S of the admissible states, it is easy to see that they are both polyhedrons (defined by a set of linear inequalities). Moreover, Q is actually the polyhedron S without the external "shell". It's also possible to think Q as the set of states in which it is possible to issue a load or a store operation and there's already both a store and a load operation request in the disk subsystem queue.
In fact for every $x \in Q$

$$u^L(x) = \min(x^q, x^f, x^D - x^\ell - x^s) \geq 1$$

and

$$u^S(x) = \min(x^D - x^\ell - x^s - u^\ell, x^F - x^f - x^s),$$

hence the following controls are admissible for every $x \in Q$

$$\begin{aligned} A: &\quad (u^\ell(x), u^s(x)) = (1, 0), \\ B: &\quad (u^\ell(x), u^s(x)) = (0, 1), \\ C: &\quad (u^\ell(x), u^s(x)) = (0, 0). \end{aligned}$$

Since the disk could complete either a load operation, a store one or nothing, the following states are reachable from state x with probability greater than zero:

$$x_{k+1} = \begin{pmatrix} x^f_{k+1} \\ x^q_{k+1} \\ x^\ell_{k+1} \\ x^s_{k+1} \end{pmatrix} = x_k + \Delta_x(k) = x + \Delta_x(k),$$

with $\Delta_x(k)$ that can assume the following values with probability greater than zero depending on the control used:

$$A: \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ -1 \\ 0 \\ 0 \end{pmatrix}$$

$$B: \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$C: \begin{pmatrix} 0 \\ 0 \\ -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

By summing a maximum number of two of the previous Δ_x vectors, it's possible to obtain $\pm e_i$, where e_i is the i-th vector of the canonical $\mathbb{R}^4$ base. This means that every state that is a neighbor of x in the lattice is reachable from x in at most 2 steps. Since the polyhedron Q is clearly connected, this means that every admissible state (including those belonging to the "shell") is reachable from a state $x \in Q$.

It remains to prove that given an initial state $x \in S,\ x \notin Q$ it is possible to reach Q with probability greater than zero. Without discussing formally the numerous possible cases, it's intuitive that with the following choice of controls:

Case	Control policy adopted
$u^L(x) = 0 = u^S(x)$	$(u^\ell(x), u^s(x)) = (0, 0)$
$u^L(x) = 0, u^S(x) \neq 0$	$(u^\ell(x), u^s(x)) = (0, u^S(x))$
$u^L(x) \neq 0, u^S(x) = 0$	$(u^\ell(x), u^s(x)) = (u^L(x), 0),$

S is reached with probability greater than zero.

The result just found is interesting because it represents a good starting point to prove that the optimal control policy is stationary. Such a result would be quite useful with respect to the heuristic considerations developed in section 5.4.

Chapter 6

Conclusions and remarks

The most important result obtained in this work is indeed the characterization of the optimal load policy. Of course, it is not enough to actually implement an optimal *page I/O manager*, but at least the complexity of the problem is significantly reduced. Many efforts have been put toward the goal of finding the optimal store policy, and even if we were not able to find the solution, the proof proposed in section 5.3 is certainly a good starting point to that end. However we should be aware that the optimal time dependent solution may be complex and there is the possibility that it could not be described in a simple closed form easy to grasp. Actually, even if the complexity of the proof regarding the optimal control policy may be due to a unfortunate formalization of the problem (as it is often the case with problems involving probability distributions), it could also be a clear sign that an elegant closed form solution to the problem does not exist.

If that is the case, then there is no other solution than turn to a simulative approach in which candidate policies are compared with the optimal solution found through dynamic programming techniques. Clearly, while this turnaround to the problem is feasible, a mathematically sound solution would be more appreciated, mainly for the insights to the problem that it would provide. However, even in the worst case in which there is a resort

to simulations, the characterization provided in this work is of great interest because now one can concentrate on a one-dimensional problem.
As a future direction, it would be interesting to find analytical expressions for the total cost achieved by such candidate policies, in particular for the promising target policy described by equation 5.15.
The solution to the problem with unlimited free frames, although resulting from an application of well known techniques of the Queuing Theory, provided a useful lower bound to the *fault service time*, that is likely to be tight in the cases of low fault rates (that is when store operations don't delay significantly the loads).
Finally the model proposed in [3] shows a good tradeoff between an unavoidable simplification of the reality and some non-linearities that make the solution of the problem far from trivial.
In this context, the *disk subsystem* models developed are probably too complicated to be used at this stage, but they retain a certain interest on their own, besides their utilization into this particular context. In fact they focus on an aspect that was given little or no attention at all in literature, and they manage to predict very well the results obtained in simulations. Indeed their major limit is the total lack of experimental validation that would be certainly needed to provide a stronger scientific result. In that sense, a non neglectable feature of real disks that should probably be taken into account is their cache, that we haven't considered at all. To that end, a good starting point could be [9].
In conclusion, even if the work did not satisfy completely all our initial goals, namely to find a closed form control law for the *page I/O manager*, the results found are encouraging and induce us to continue the work towards designing a *page I/O manager* that could lead to significant improvements to VMM systems performance.

Bibliography

[1] Dimitri Bertsekas. *Dyanamic Programming and Optimal Control, Vol I* Athena Scientific, Belmont, Massachusetts , 1995

[2] Dimitri Bertsekas. *Dyanamic Programming and Optimal Control, Vol II* Athena Scientific, Belmont, Massachusetts , 1995

[3] Gianfranco Bilardi, Kattamuri Ekanadham, Pratap Pattnaik. *A nearly optimal autonomic control algorithm for Page I/O in virtual memory management*, Manuscript, 2006

[4] Stefano Ermon. *Modelli statistici per l'analisi delle prestazioni degli hard-disk in funzione del carico di lavoro* Tesi di laurea, Università di Padova, 2006

[5] D. Gross and C. Harris. *Fundamentals of Queueing Theory.* Wiley, 3rd edition, 1998

[6] Jeffrey O. Kephart and David M. Chess. *The vision of autonomic computing* IEEE Computer, January 2003.

[7] Athanasios Papoulis. *Probability, Random variables, and Stochastic Processes* McGraw-Hill , 3rd edition , 1991

[8] C. Ruemmler and J. Wilkes. *An Introduction to Disk Drive Modeling.* IEEE Computer, March 1994.

[9] Elizabeth Shriver, Arif Merchant and John Wilkes. *An analytic behavior model for disk drives with readahead caches and request reordering* ACM SIGMETRICS Performance Evaluation Review, Volume 26, Issue 2, 1998

[10] Elizabeth Shriver. *Performance modeling for realistic storage devices.* PhD Thesis, New York University, May 1997

[11] A. Silberschatz, P. B. Galvin, G. Gagna. *Operating Systems Concepts* Wiley , 7th edition , 2005

[12] William Stallings. *Operating Systems* Prentice Hall , 3rd edition , 1998

[13] Andrew S. Tanenbaum. *Modern Operating Systems* Prentice Hall, 2nd edition, 2001