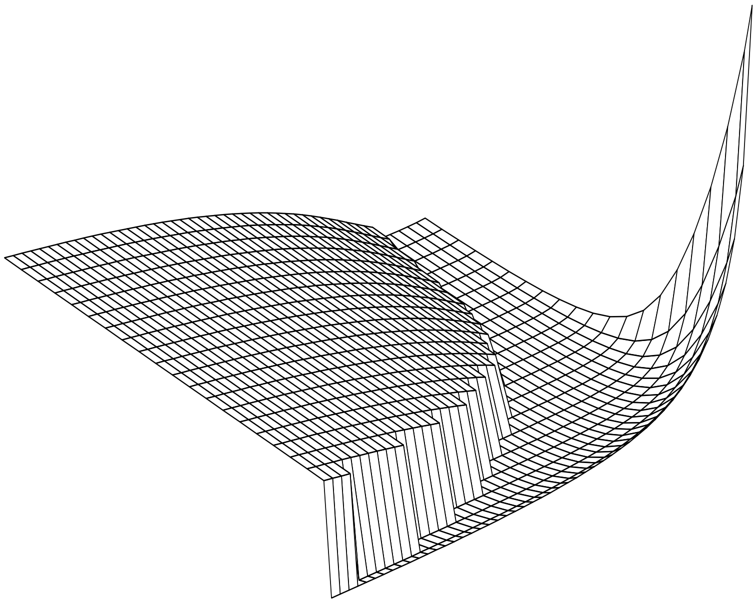
Cost of Searching

Probabilistic Analysis of the Self-Organizing
Move-to-Front and Move-to-Root Sorting Rules



Josefin Bodell

Josefin Bodell
Department of Mathematics
Division of Mathematical Statistics
Royal Institute of Technology
S – 100 44 Stockholm
Sweden

Cover: 3D illustration of density generated using Matlab
(cf. Chapter 4, Figure 4.6.5).

Printed in Sweden by KTH Högskoletryckeriet, Stockholm.

ISBN 91-7170-153-2

Preface

Perhaps one could say that the seed of this monograph was sown some time in November, December 1993 in Minnesota. At this time, my supervisor, professor Lars Holst, was a guest at the Institute of Mathematics and its Applications at the University of Minnesota and this is were he came across the issue of self-organizing sorting rules. In the beginning of 1994, I began to study the move-to-root rule and a year later, in January 1995, I propounded my licentiate thesis on the subject.

During the spring of 1995, Lars and I talked about possible ways to proceed. Among other ideas, we thought about introducing somehow different priorities into the problem. Still, it was not until after a talk I had with professor Jim Fill of the Johns Hopkins University, who visited the Mittag-Leffler Institute in May, that the two-user approach considered throughout this thesis really came to light. Since then, almost two years have passed and the monograph that you now see before you tells at least some of the story of what has happened during that time...

I had great fun working on this thesis and I would like to take the opportunity to thank everyone who contributed to making it such a good time.

Thanks, Lars, for your guidance and never-ending enthusiasm in supervising my work. Thank you, Colleagues and Work-mates, for your time and support and thank you, Friends, simply for being there. And last, but certainly not least, thanks Mum and Dad for *everything*.

Josefin Bodell
Stockholm, April 1997

Seek, and ye shall find

Matthew 7.7

Contents

Chapter 1

Introduction

How long time does it take to find an item when they are ordered according to certain simple but clever sorting rules? This is basically the question at issue in this monograph.

My dictionary defines 'sorting' as 'arranging things in groups; separating things of one type from things of other types'. In the context of this monograph though, perhaps a more adequate definition of the sorting concept would be 'arranging, ordering'.

1.1 Move-to-front

Imagine a number of books in a bookshelf. Whenever we want a certain book, we start looking for it in the upper left-most position, that is, we compare the title of the book we search for with that of the first book in the shelf. If the titles disagree we move on to the next book in the shelf, compare titles and in this manner successively work our way through the shelf to the requested book.

We, of course, want to find the desired book as quickly as possible. One idea is to use the fact that different books are more or less popular — we may need to have a look in the dictionary more often than we need the Collected Works of Shakespeare (or the other way around). Bearing this in mind, we arrange the books in order of decreasing popularities, the most popular one first, followed by the next most popular etc. Our search time is now the shortest possible since the book we most frequently want is the one most easily accessible while we must search longer for books we only seldom want.

Figure 1.1.1: *Move-to-front.*

But what if we do not know the popularities before-hand? Or what if they change in time? How should we arrange the books then? Somehow we wish to use a sorting rule that causes the books to 'automatically' arrange themselves in order of decreasing popularities. One solution is the following strategy: When we have located a book in the shelf, we pick it out, read it, and then put it back in front of all the others instead of in the position where it was found. After a while, popular books will have gathered at the front of the shelf while less popular ones are kept further back since they are seldomly requested. This is the *move-to-front* rule.

1.2 Probabilistic model

If the initial arrangement and the sequence of requests were known to us, we would in principle be able to tell where the books are at every given instant. In practice, however, this is not a very tempting approach even if we did in fact have access to the required information. Moreover, we would get no information about the performance of the move-to-front rule in general — if the request sequence was altered, we would have to do the calculations all over again. So instead, to get more general information about the performance, we use a probabilistic model.

Each time a request is made, let 'chance' decide which book we want. We let chance rule in such a way that the probability of requesting a certain book agrees with its popularity.

Usually (in most papers on this topic, cf. the bibliography), the requests are imagined to occur at each unit of time. In this monograph however, an extended model is used where the times at which requests are made are also random. Introducing this extra randomness in fact simplifies the calculations and as we shall see provides means to deal with popularities that change in time.

1.3 Search cost

How long do we have to look for a book in such a shelf? Every comparison between the title of the requested book and one in the shelf demands a constant amount of time and so we may measure the search time as the number of comparisons required to locate the desired book. This quantity is called the *search cost*[1].

Using the probabilistic model above, a number of questions may be answered.

- What is the probability of the search cost being equal to a certain number?

This is known as the probability function for the search cost. Another question closely related to the previous one is:

- What is the probability that we need to search less (longer) than a certain time? Or differently put, what is the probability that the requested book is to be found among the first (last) x books in the shelf?

The answer to the latter question is of interest when constructing computers, cf. Section 1.8 below.

Sometimes the probability function is very complicated and difficult to handle. For instance, this is the case for the move-to-root rule described below, Section 1.6. We might then choose to study simpler generic quantities.

- What is the mean search cost?

Before we make a request this value is our best guess of how long we shall

[1]In the case of the ordinary move-to-front bookshelf, the number of comparisons is equal to the number of books in front of the requested one *plus one* since we must compare titles also with the requested book to know that we are done. However, for aesthetic reasons, it is nowadays customary to take the search cost to be simply the number of items in front of the one requested.

need to look. We expect the cost to keep somewhere about this value but each search does not require the same cost.

- How much does the cost vary about the mean?

The spread of the cost is given by the variance. Note that if the probability function is known, the mean and variance follow.

1.4 Asymptotics

At first, the arrangement on the shelf depends heavily on the initial order of the books. After some time though, the shelf will have 'forgotten' how it started — the move-to-front rule is alone responsible for the arrangement. This state is called *stationarity* and appears when sufficiently many moves have been made to make the books keep within a set of permutations that all have expected search cost close to a constant value — the expected stationary search cost[2]. Then, many further requests are not likely to cause any significant change in the search cost.

- What does the shelf look like after long time?

Mathematically, 'long time' means that time tends to infinity. In practice, however, we do not want to wait forever. At least approximately we should be able to use the results sooner. How long time must pass before this approximation is acceptable in the sense that the search cost is sufficiently close to the stationary one?

- How long is 'long time'?

A parameter other than time which might tend to infinity is the number of books.

- What happens when the number of books becomes extremely large?

[2]Alternative terminology is used in several papers (cf. the bibliography). To retain clarity later on when both time and the number of records tend to infinity, I have knowingly avoided the common denomination of *asymptotic* search cost.

1.5 Binary search trees

So far we have used only the different popularities to arrange the books in the shelf. But why not use also the already present order between the books — the lexicographical order! Perhaps the ordinary linear bookshelf is not always the appropriate storage structure. Can we use the extra information in the lexicographical order to do better?

Imagine that you want to look up a certain word in a dictionary. How do you proceed to find it? Most people probably would not start at the letter 'a' working their way through all the words until they find the one they want. Instead one opens the dictionary at random and decides whether the requested word comes before or after a randomly chosen word in lexicographical order. If the requested word precedes the current entry, then one repeats the procedure but this time considering only the pages before the one just checked. Proceeding in this manner, successively narrowing down the possible locations, one rapidly finds the word.

Based on this idea we rebuild our bookshelf into a more mobile-like structure: Onto each book, mount two hooks, one in the lower left corner of the book and the other in the lower right. Select a first book, memorize its title and let this book hang from a hook in the ceiling. Select a second book. If the title of this book precedes that of the first lexicographically, hang it on the first book's left hook, otherwise hang it on the right hook. Proceed in this manner until all the n books have been hung on free hooks (see Figure 1.5.1). The 'bookshelf' has now assumed the structure of a *binary search tree*. The position of the first book, the one closest to the ceiling, is called the root. The tree grows from the root, at each node ramifying into at most two (hence the word binary) branches according to the lexicographical order — if less (preceding), go to the left, otherwise to the right. This order facilitates searching and for this reason such an ordered tree is called a search tree.

When searching a book in this structure we use *binary tree search*: First compare the requested book with the one in the root. If the titles agree we are done while if they do not we move down one level in the tree. If the requested book is lexicographically preceding the target book, look for it in the left subtree, otherwise search in the right. Repeat until the book has been found.

If the tree is *balanced*, that is, if all the levels are filled with books except the one closest to the floor, we are able to exclude half the remaining books in each comparison.

Figure 1.5.1: *Binary search tree.*

1.6 Move-to-root

The tree-analogue of the move-to-front rule is the *move-to-root* rule where the requested item is moved to the beginning of the tree structure, viz. the root. In the case of move-to-front the requested book is moved to the front while the relative order between the other books is left unchanged. The move-to-root case is more complicated. If we simply move the requested book to the root we ruin the order in the tree. Instead, we perform a series of *simple exchanges* in which we make the requested book the root of the next larger subtree, that is, we move it up one level at a time, at each stage keeping the lexicographical ordering intact, until the book has become root of the entire tree (cf. Section 5.2).

We measure the search cost as the number of comparisons[3] needed to locate a book in this structure. Using the terminology of family trees (which is customary), the search cost is now given by the number of *ancestors* — the books we pass on our path down the tree from the root to the requested book — plus one (the last comparison we need in order to know that we are done). Hence it suffices to keep track of the ancestors for each record and how the ancestry changes when a move-to-root operation is made.

The questions we ask are basically the same as in the move-to-front case, only since this is a more complex setting the answers are more difficult to get at. In fact, when it comes to general popularities, not much has been known about the search cost apart from the mean (including its rate of convergence) and variance (cf. Section 1.11.2). In addition, we shall derive two different expressions for the probability distribution. They are both however rather implicit and it is not obvious from either of them what a general formula for the search cost probability function might look like.

In contrast, the special case of equally popular books has been rather thoroughly investigated. The interest in this particular case turns out to be justified by more weighty reasons than merely being possible to calculate by hand, as we address the following question:

- What is the worst case and when does this occur?

The best that could possibly happen in terms of search cost is that we always want the same book. Then this book will always be in the first position where we find it immediately, both in the 'ordinary' linear shelf and in the alternative binary tree one. In a linear shelf, the worst case (giving

[3] Cf. the move-to-front search cost p. 3.

the maximal search cost) occurs if all the books are equally popular[4]. Then there are no impopular books which the move-to-front rule can put in the rear of the shelf and which we could more or less forget about.

It used to be conjectured that the same was true for trees but testing this conjecture, we find instead that the mean[5] cost becomes worse still if the lexicographically first and last few books are slightly more popular than the others. It is essential that it is the first and the last books that are the most popular ones. As root, such a book will force all the others to hang from just one of the hooks in order to maintain the lexicographical order whereby we loose the advantage of the tree structure. The explanation to the worst-case popularities is that in this case the tree will fluctuate between extremely unbalanced states.

So far we have only been able to obtain numerical results about this phenomenon (cf. Section 6.1.3). How to describe it analytically is still an open question.

1.7 Several users

What if several users with different priorities request the same information? In the books-in-a-shelf analogy this corresponds to one (high-priority) owner who rearranges the books according to his preferences using the move-to-front/move-to-root rule. In the sequel, this user is referred to as Controller. A low-priority user, Second User, gets to borrow the books but must return them where they were found in the shelf. Hence Second User's search cost will depend on Controller's preferences since these determine the arrangement of the books. The more similar the two users are in their taste for literature, the lower the search cost for Second User.

Throughout this monograph we shall assume that the time spent rearranging the shelf after a request is negligible and hence a requested book may always be found in the shelf — there are no blocking times. With this model, the search cost properties for Second User will apply to possible additional low-priority users who may also be allowed to borrow books but not rearrange them.

[4]The problem we address is to find the worst case *distribution* of Second User's (cf. Section 1.7) popularities, assuming that they are constant in time. If instead we had wanted to describe the worst possible *request sequence*, this would have been to request the record currently in the last position of the shelf or in the tree case, the one with most ancestors.

[5]Because of the complexity of the distribution we consider only the mean search cost.

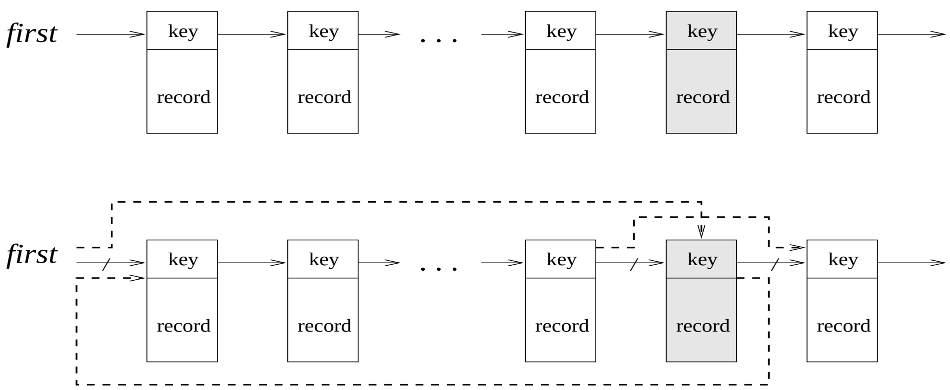


Figure 1.8.1: Moving the shaded record to the front of a linked list.

1.8 Application to computers

A single user

Consider a set of *records*, units of information, stored in a computer. Together, the collection of all the records form a *file*. (Sometimes the word 'table' is used to indicate that the file is small and frequently, large files or a group of files are called a 'database'.) To pursue our books-in-a-shelf analogy, we can liken the file of records to a bookshelf in a library. To each record an identifying field, a *key*, is associated. The keys correspond to the cards in the library card index. In order to find a book it is sufficient to find the corresponding card in the index. The *key value* of the card may be for instance the name of the author or the book title. In fact, we may have separate indices sorting the books with respect to different attributes. Similarly, a part of the record is chosen to define the record's key value. As in the library analogy, we may keep the records simultaneously sorted by different attributes letting each record have more than one key using different parts of the record.

The (single-user) move-to-front rule has long been used to keep records dynamically arranged in order close to that of descending popularities. Move-to-front may be implemented as a *linked list*. Let each record have a *pointer* pointing to the next record and let a *first* pointer point to the 'first' record, the one 'in front of' every other. A move-to-front rearrangement is then performed by resetting three pointers, see Figure 1.8.1. The bottom line is that the records are not really moved at all — they stay where they are. Only the pointers defining their arrangement are changed.

The move-to-root rule may be implemented in a similar way assigning two pointers to each record, a *left* pointer pointing to the subtree of records with smaller key values and a *right* pointer pointing to the subtree of those with greater key values. Again, we need a *first* pointer pointing to the root record. Moving a record to the root, several pointers must be reset, cf. Chapter 5. On the other hand, the search cost is reduced significantly compared to that of the move-to-front rule.

In recent years another related question has arisen, namely that of *cache* memories. A cache[6] is a small, fast (and expensive) buffer memory that one may choose to put between the processor and the primary memory to speed up the access time. When the processor requests a record in the primary memory, a copy is made and put in the cache. The next access of the record is thereby sped up considerably (provided that the record is still in the cache the next time it is requested). Generally speaking, memories can be made faster the smaller they are (to be really fast, the memory should be located on the same chip as the processor — a place of very limited space). Hence, the aim is to make the cache as small as possible but still large enough to contain the most frequently requested records since an item not found in the cache must be searched for in the 'ordinary' primary memory. What size should the cache optimally be? Knowledge about the probability of a *cache fault* — a request for a record that is not in the cache — helps to answer this question. (The link between caching and move-to-front is described in detail in Section 1.11.1.1 below.)

Two users

Two competing processes with different priorities using the same database is by no means an uncommon situation. For instance, imagine a supervising system watching over complex technical systems such as industrial processes, hospital equipment, security systems etc. There is a background process running continuously and hence making frequent accesses to the database, checking that everything is in order, counting control sums making sure that

[6] *Cache* is the French word for hiding place. In non-technical English, a cache is a place for hiding food, treasure or weapons.

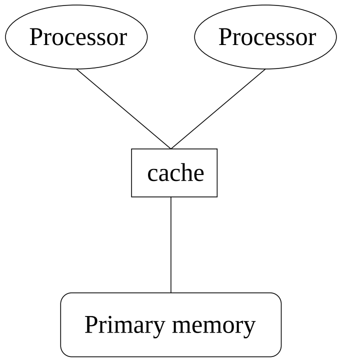


Figure 1.8.2: *Two competing processors sharing one cache.*

the memory cells are accurate and so on. This is the low-priority Second User.

When an alarm is registered, the database needs to be accessed quickly in order to interpret the alarm, determine whether it is a true or false one and decide what measures to take. This is the high-priority process, Controller, according to whose preferences the database is rearranged.

In the caching context, the two users situation answers to the presence of two processors sharing one cache memory, Figure 1.8.2. The high-priority Controller rearranges records while the low-priority Second User leaves them where they are.

The two users case is straightforwardly extended to several users (one Controller, all the others low-priority) allotting Second User status to all low-priority users. In fact, assuming no interference (such as blocking or interruption) between the users, our results are directly applicable to each of these users.

The introduction of several users can be the gateway to the study of general multilevel cache hierarchies, Figure 1.8.3, with different user priorities.

1.9 What's new?

Two users

The introduction of a second low-priority user besides the high-priority 'owner' is a new angle to the problem of self-organizing search. We focus on

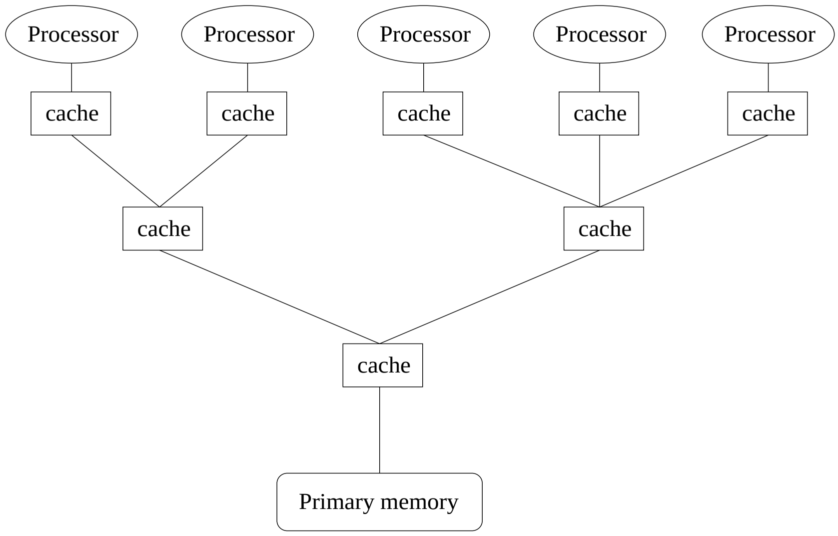


Figure 1.8.3: *Multilevel cache hierarchy.*

Second User's cost of searching and give answers to the questions mentioned above.

In Part I of this monograph, the move-to-front rule is considered. Assuming constant popularities and independent requests we derive the probability distribution (the generating function along with a rather implicit expression for the probability function), expectation (mean) and variance of Second User's search cost at time t given that the item list (bookshelf) was initially in arrangement π. Known analogues for the single-user distribution and expectation follow, as well as the previously unpublished single-user variance, if we let Controller and Second User be identical.

Similarly, we determine Second User's stationary search cost distribution in terms of the generating function, expectation and variance. Again, the known single-user analogues follow.

Further, we generalize the results of Fill [22] and Fill and Holst [23] to our two-users case in that we derive results concerning stochastic ordering of Second User's search cost, and use it to determine lower and upper bounds for the discrepancy between the stationary search cost and that in a move-to-front list started in arrangement π. (We also give the rate of convergence for some examples of request distributions.)

In Part II we turn to the case of binary search trees and the move-to-root rule. Under the same assumptions of constant popularities and independent requests, we determine the expectation and the variance[7] of Second User's search cost at time t as a function of the initial shape τ of the tree. The single-user analogues, too, are new.

We also determine the stationary expectation and variance and find that the known single-user expressions follow.

We find — numerically — the 'worst' choice of request probabilities in the single-user case. By 'worst' we understand the set of constant popularities giving the maximal expected search cost. They turn out to be close, but not equal, to uniformly distributed request probabilities.

To describe the distribution, we derive the generating function for Second User's search cost. In fact, we give two different expressions — one on recursive and one on integral form. It is by no means obvious from either of these what the general probability function is, however we do solve the recursive equation when all the request probabilities are equal, recapturing the known generating function (and probability function).

Popularities that change in time

So far, only constant popularities have been considered, but what if the preferences for different items (records/books) change in time?

The motivation for self-organizing rules such as move-to-front or move-to-root is their self-adjusting qualities — that they reorganize on basis of experience. This makes them particularly well suited for situations when the popularities change in time, hitherto an uninvestigated field. The analysis of the performance with *time-dependent request probabilities* is therefore a main point in this work.

Allowing the popularities to be time-dependent, we derive the probability function for Second User's search cost at time t when the move-to-front list is initially in arrangement π and give the generating function for the long-term cost. By 'long-term' we understand that enough time has elapsed to make sure that the list has 'forgotten' how it started and is hence governed by the move-to-front rule alone. (The expectation and variance follow but they are left as an exercise for the reader.) Since time-dependent request probabilities has not been treated before, the corresponding single-user results are also new.

[7] The derivation of the variance is found in Appendix A.

As the number of items becomes very large (tends to infinity), so does the search cost for 'regular' request probability distributions. However, considering the *fraction* of items we need to check before we find the one we want, that is, the search cost divided by the number of items, turns out to be quite a different matter. In fact, the distribution function for the search cost fraction tends to a surprisingly simple explicit expression, both when we consider only the long-term behaviour and when we consider the transient dependence on the initial arrangement. As usual, the single-user analogues follow immediately.

The asymptotic distribution functions are also valid for constant request probabilities satisfying the necessary regularity conditions. Hence, we have here a new method for determining the asymptotic search cost distribution also for the 'ordinary' move-to-front model assuming constant popularities. As an illustration, we carry this through for a couple of standard distributions.

Intuitively, one might expect that if the two users frequently want the same books, Second User's cost is less than if their preferences are often very different. We derive some results on stochastic ordering supporting this guess.

The model of time-dependent popularities may also be applied to the move-to-root situation. All the results concerning the model of constant request probabilities may straightforwardly be extended to this new situation. We show how to do this as we determine Second User's long-term expected search cost.

1.10 Open problems

There are of course a number of open problems which one can think of after reading the review of the field below. Some of those more closely related to the new results of this monograph are the following:

- Derive the 'worst' request probabilities for single-user move-to-root analytically.

- Determine a general explicit expression for the move-to-root search cost distribution.

- Derive the asymptotic distribution for the move-to-root search cost as the number of records tends to infinity.

- Find rates of convergence to stationarity for the move-to-root (single- and two-user) search cost.

- Determine rates of convergence for two-users move-to-front, not just lower and upper bounds that do not always agree.

- Extend the model of time dependent request probabilities — let the intensities of the Poisson processes (cf. the embedding procedure below) be *random* as opposed to deterministic functions of time.

- Study general multilevel cache hierarchies with different user priorities, cf. Section 1.8.

1.11 Review

The purpose of this section in to summarize previous work and review the developments in this field. This background is however not necessary to understand the rest of the monograph.

1.11.1 Move-to-front

Although the move-to-front rule has been extensively studied over the years, its proper origin is unknown. The main field of application has been within the domain of computer science where move-to-front has long been used for computer sorting. There is at least one other completely different area where the move-to-front rule is of interest, namely population genetics, briefly described in Section 1.11.3. However, since the computer sorting rule application is the more established one — its documented history begins in the 1960's but was already in use long before that — this is where we begin our survey.

Suppose that a linear list of n records is given and that whenever a record is requested, we start looking for it in the first position, comparing sequentially until we find it. Our aim is to minimize the time of search, measured as the *search cost* — the number of records in front of the requested one in the list. At first, the search cost was defined as the number of required comparisons (which equals the number of preceding records plus one). However, the expressions for the properties of the search cost have turned out to be more appealing if the number of preceding records is taken as the cost definition. Nevertheless, the original definition is still used in the move-to-root context (Section 1.11.2). (Despite the confusion this might cause, in this monograph, we have adopted these conventions.)

If the records have different popularities and these are *known*, then optimally the records should be arranged in order of decreasing popularities. Usually, though, the popularities are not known. In 1965, McCabe [57] suggested to use the self-organizing scheme now known as the *move-to-front* rule in this case: When a record has been requested, move it to the front (first position) of the list. Then, when move-to-front has been operating for some time, frequently requested records will automatically have gathered at the beginning of the list giving a low search cost.

It is assumed (in all the papers on self-organizing search in the bibliography as well as throughout this monograph) that only one record is requested at a time and that it is replaced before another request is made. Hence, there are no blocking times — the records can always be found in the list.

Methods like move-to-front are sometimes called *memory-free* self-organizing rules. This is to separate them from another method of solving the problem of unknown popularities: Count the requests and keep the file (dynamically) in order of decreasing request frequencies, see e.g. Hofri and Shachnai [36]. The law of large numbers then guarantees that eventually the file will be in optimal order. Memory, however, used to be expensive and one could often not afford to reserve the required memory space for the counters. Hence the interest in memory-free heuristics which reorganize the file without the use of extra information like frequency counts.

Nowadays, memory is generally cheap. One might think that this fact would have caused the interest in in self-organizing rules to cease. However, in addition to the mathematically interesting issues that arise in this field, there is also the problem of caching.

1.11.1.1 Caching

The move-to-front rule is closely related to the problems of *caching*. A cache is a small, high-speed memory that is placed between the much slower primary memory and the processor in computers in order to speed up the access time. Since the cache is small, there is only room for the most popular items in it. If an item is requested that is not in the cache, a *cache fault* occurs and one must search for the item in the primary memory. The *cache algorithm* specifies how to reorganize the cache on a cache fault.

There are several cache algorithms in use. LRU (Least Recently Used) replaces the least recently requested item (for the one causing the cache fault), FIFO (First In First Out) replaces the 'oldest' record to have entered the cache while RAND (RANDom) replaces any record in the cache with equal probability. OPT (OPTimal) replaces the record whose next request is most remote in time. Hence, OPT is rather an odd algorithm in that it depends on the future. For a brief summary of the performance of the different cache algorithms and references to further reading, see Flajolet, Gardy and Thimonier [24, pp. 218 - 219].

In their paper [24], Flajolet et al also gave an alternative description of LRU using move-to-front. Assume that the items in the cache are kept in a list arranged in order of increasing time since last requested. Then LRU is simply move-to-front where we keep track only of the first k items in the list (where k is the size of the cache).

1.11.1.2 Independent requests

The simplest, yet relatively general and hitherto most commonly considered model of how the requests are made is this: Whenever a request is made, it is for record i with probability $p_i > 0$, where the p_i's, $i = 1, \ldots, n$, are unknown constants such that $\sum_{i=1}^{n} p_i = 1$. Furthermore, the requests are independent.

Assuming that the number of records n in the list is finite, the $n!$ different permutations of the records may be viewed as states of a finite, irreducible, aperiodic and hence ergodic discrete-time Markov chain. Here, time is measured by the number of requests: when a request is made time leaps one unit step forward. The move-to-front Markov chain of permutations is also known as the *Tsetlin library* or the *heaps process.*

It is true that McCabe was the first to study move-to-front probabilistically motivated by the immediate computer sorting application. However, it would not be fair to forget M. L. Tsetlin. Imagine a heap of books, rearranged in such a way that after a request, the book is put back on top of the heap. Already back in 1963, Tsetlin [75] represented the heap by a *finite automaton*, recognizing that the books in the heap may be described by means of a Markov chain.

Stationary behaviour

Loosely speaking, the move-to-front list has reached stationarity when enough time has passed (sufficiently many moves have occurred) to ensure that the dependence on the initial arrangement has worn off, cf. Section 1.4.

McCabe [57] showed that according to the law of large numbers for ergodic Markov chains, the expected search cost tends to the expected stationary search cost in probability as time (the number of requests) tends to infinity. He also gave this expected stationary search cost and the variance (cf. Corollary 2.3.2.3 below). Tsetlin [75], in 1963, had independently derived a closely related result, namely the expected depth of book k in the heap — what we know as the expected stationary cost of searching record k. Arbitrary moments were derived in 1981 by Gonnet, Munro and Suwanda [30].

The stationary distribution[8] for the move-to-front Markov chain of permutations was given in 1972 by Hendricks [31]. Again, Tsetlin [75] had already derived this nine years earlier for his finite automaton. Later, in 1996, Fill and Holst [23] showed how one may exploit the close relationship between stationary move-to-front and sampling without replacement to derive the stationary distribution of the chain.

The stationary distribution of the chain was extended in 1974 by Letac [48] to the case of lists with a countable (infinite) number of records. He also gave necessary and sufficient conditions for transience, null recurrence and positive recurrence for the chain. In 1977, Nelson [59] continued along this track and derived ratios for the one-step transition probabilities of the Markov chain in the countable case to give some idea of the relative size of the limiting probabilities.

A 'real-life' computer scientist is probably more concerned about the search cost than about the behaviour of the entire chain of permutations. So not just knowing the expected search cost but the entire probability distribution for the search cost would be useful.

The generating function for the stationary search cost was first derived in 1992 by Flajolet et al [24, Theorem 5.1] in the context of long-run cache fault probability for LRU caching. In 1996, Fill and Holst [23] re-derived this formula (now with the stationary search cost in mind) using different methods.

Because of the complexity of the general stationary search cost distribution, in 1996 Fill [22] chose to consider the limiting distribution for the stationary search cost as the number of records n tends to infinity. Assuming the basic model of unknown, constant request probabilities and independent requests, he gave the limiting stationary search cost distributions for a number of request probability distributions: the uniform and geometric dis-

[8] Also known as the *steady-state* distribution.

tributions, Zipf's law (standard as well as generalized) and power law (see Section 3.5 for definitions of these distributions).

Transient behaviour and rates of convergence

Shortly after the clock was started, the current list permutation depends heavily on the initial arrangement and the stationary results are not valid. We need different formulas to describe the transient behaviour.

The t-step transition probabilities of the chain were derived in 1977 by Nelson [59] and essentially the same probabilities were derived in 1994 by Phatarfod [63] using the connection between move-to-front and the coupon-collector's problem.

The coupon collector's problem is the one every child encounters when manufacturers of for instance cereals put some coveted plastic object in the package. There are n different types of objects and each package is equally likely to contain any given type. If you buy one package at a time, how long do you have to wait until you have a full set of objects?

Suppose for instance that ultimately the order in the list is $3, 4, 2, 1, \ldots$. Then 3 must be the last item to have been requested, 4 is the item to have had its last request just before the request(s) for 3, 2 is the third last item to have had its last request, etc. The (end of the) request sequence might have looked like $\ldots, 1, 4, 2, 4, 2, 3, 3, 4, 3$. Viewing the sequence in *reversed order*, $3, 4, 3, 3, 2, 4, 2, 4, 1, \ldots$, only registering the first draw of each item, we get the ultimate list order above. This is a situation that occurs in the coupon-collector's problem.

Phatarfod [62] determined the eigenvalues of the transition probability matrix. All the eigenvalues are real, non-negative and (except for the value 0) in the form $p_i, p_i + p_j, p_i + p_j + p_k, \ldots, p_1 + \cdots + p_n = 1$ excluding the case where the summation is over $n-1$ elements. Phatarfod expressed the probability of a transition in t steps (requests) from one permutation to another in terms of the eigenvalues where the largest non-unit eigenvalue, $1 - \min_{i \neq j}(p_i + p_j)$ (with multiplicity 1), gives the rate of convergence to stationarity. He also gave the expected search cost at time step t (and a method for deriving the variance) assuming that the list is initially in random order.

In 1996, Fill [21] independently derived the t-step transition probabilities for the move-to-front permutation chain in the framework of card shuffling and used the formula to recapture the stationary distribution and the eigenvalues with corresponding multiplicities. When the request probabilities are identical for each of the records, the move-to-front rule is equivalent to the 'random-1-to-top shuffle' card-shuffling scheme — for a thorough analysis, Fill refers to a paper by Diaconis, Fill and Pitman [14]. In a subsequent paper, Fill [22, Section 4] considered rates of convergence to stationarity in terms of the so-called maximum separation.

If the process is lumped, disregarding the last l elements ($2 \leq l \leq n-1$) in each permutation, it still remains a Markov chain and Phatarfod [62] determined the eigenvalues and the multiplicity of the largest (non-unit) one also in this case. Fill [21] gave the t-step transition probabilities (Remark 2.2(g)), spectral decomposition of the lumped chain, the eigenvalues and their multiplicities (Remark 3.2(a)) and the discrepancy from stationarity after t requests in terms of maximum separation (Remark 4.2(e)).

The interest in the lumped process is motivated by the connection between the move-to-front rule and LRU caching (described above).

Again, results on the search cost might be of more practical interest than those concerning the entire permutation chain. In 1979, Bitner [5] derived probabilities that record i is located to the left of record j after t moves. Assuming each initial permutation to be equally likely, he gave the expected search cost after t requests. The expectation equals the stationary one plus an additional term depending on t. This latter term can be used to measure the rate of convergence to stationarity (we shall return to this issue shortly).

In 1991, Phatarfod [62] gave similar probabilities for three records. In principle, the same method could be used to give analogous probabilities for an arbitrary number of records, however, the general expression is extremely complicated.

Bitner [5] used *overwork*, the area between the cost curve and its asymptote, as a measure of convergence. Fill [21, Section 5] determined the number of time-steps (requests) needed to make the difference between the expected search cost at time-step t and that in stationarity small and in addition, Fill [22] determined rates in terms of total variation distance based on bounds derived by Fill and Holst [23] (cf. Chapter 3 below) between the search cost distributions at time-step t and in stationarity, which is a better measure since it does not only consider average behaviour. The convergence for search cost turns out to be much faster than that for the entire permutation chain.

Continuous time

So far, 'time' has been measured by the number of requests — the moves have been made at each discrete unit of time. However, the request times themselves are not of particular interest — only the search cost they give rise to is. In 1996, instead of the prevalent discrete-time approach, Fill and Holst [23] let the requests be made at time-points of a Poisson process of intensity one. This embedding technique was introduced by Holst [37] in 1981 in a paper on sequential occupancy problems where balls are thrown at random into m cells until k cells contain at least $l+1$ balls each. In this monograph, Poisson embedding is a general theme and proves to be the key to a majority of the results. The technique is described in detail below (Section 2.2).

Utilizing the embedding, Fill and Holst derived the distribution of the search cost, in stationarity as well as at finite time assuming that the list is initially in a given deterministic permutation. (These results are generalized in Chapter 2 below.) They also showed a Schur-concavity result (see the next section) for the expected stationary search cost and gave results on stochastic ordering regarding initial permutations of the list (cf. Section 3.2 below.)

Comparison with optimal order and worst-case behaviour

To get a more detailed understanding of the move-to-front rule's performance its search cost has been compared to that of the optimal static arrangement (in order of decreasing request probabilities — the arrangement we would choose had the p_i's been known). In 1973, Burville and Kingman [11] showed that if $p_1 \geq p_2 \geq \cdots \geq p_n$, then the expected stationary search cost for move-to-front is between m and $2m+1$ where $m = p_1 + 2p_2 + \cdots + np_n$ is the expected search cost for the optimal static arrangement. In 1976, Rivest [65] showed that the expected stationary search cost under the move-to-front rule is always less than twice that under optimal static ordering. The factor two was improved further in 1981 by Gonnet et al [30] and finally in 1988, Chung, Hajela and Seymour [13] proved, using Hilbert's inequalities, that the expected stationary search cost for move-to-front is less than $\pi/2$ times that of the optimal static order for any distribution of the p_i's and that this bound is the best possible.

Gonnet et al also gave 'worst cases' of the move-to-front rule in terms of conditions on the request probabilities such that the *ratio* resp. *difference* between the expected stationary search cost for move-to-front and that for optimal (static) order is maximized.

Fill and Holst [23] proved another kind of worst-case behaviour in that they showed that the expected stationary search cost is a Schur concave function. (In fact, in 1980 Kan and Ross [39, Proposition 1.5] presented the same conclusion, however in a different context, as they showed that the expected search cost of the k-in-a-row policy, cf. Section 1.11.1.7, is a decreasing function of k.) Roughly, this means that the more equal the request probabilities become, the greater the expected search cost. Hence, the search cost is maximal for equal request probabilities. The intuitive reason for this is that when all the records are equally popular, none of them can be cleared away (to be put far back in the list).

1.11.1.3 Dependent requests

Computational experiments have shown that the move-to-front rule in fact performs even better than expected (Knuth [43], p. 399). Why is that? One shortcoming of the model above is that successive requests are often not independent. Instead, when a record is requested it is likely to be requested again shortly. Also, a request for a certain record often increases the probability that a neighbour will soon be requested, or as Knuth puts it, 'small groups of keys tend to occur in bunches'. This feature is known as *locality*.

Several researchers have chosen to let a Markovian model describe this dependence. For instance in 1984, Lam, Leung and Siu [47] modeled the request sequence as a Markov chain with transition matrix $P = (p_{ij})$, where p_{ij} is the conditional probability that record j is requested given that the previous request was for record i. The Markov chain is assumed to have the stationary probabilities $p_1, \ldots, p_n$, where p_i is record i's stationary request probability (long-run request frequency fraction). Under this model they obtained the expected stationary search cost. The case of independent requests was shown to be a special case. Furthermore, $m = p_1+2p_2+\cdots+np_n$ is still a strict upper bound of the expected stationary search cost. Lam et al also showed that with dependent requests, the 'optimal' static arrangement as defined for independent requests (keep the records in order of decreasing stationary popularities) does not necessarily give the optimal cost. In fact, some request patterns favour move-to-front so that its cost is less than m.

In 1993, Phatarfod and Dyte [64] (communicated by Dobrow and Fill [15] referring to this preprint) derived the eigenvalues and their multiplicities for the transition matrix for the move-to-front chain under the Markovian model. Chassaing [12], also in 1993, considered the optimality of move-to-front under the same Markovian model and gave properties of the class of transition matrices such that move-to-front is optimal among rules that use no information about the future (so-called on-line rules, cf. p. 26) with respect to stationary search cost.

In 1995, Dobrow and Fill [15] derived the t-step transition probabilities and the stationary distribution (given in three different ways). They also recaptured the spectral structure and further gave bounds on the discrepancy between the distribution at some fixed time and in stationarity, both in terms of separation [15, Section 6] and total variation distance [15, Section 7].

The same year, Rodrigues [66] compared the stationary distributions and expected search costs of two different Markovian request models, one ruling out the possibility of two consecutive requests for the same item and the other imposing the so-called p-correlation (which means that the item currently requested is next requested again with probability p; otherwise the next request is for one of the other items with probability corresponding to their respective underlying popularities). In addition, Rodrigues [67] studied the rates of convergence in terms of total variation distance.

We shall not pay any (further) attention to dependence between requests however natural such a model may seem. In justification of this decision, let me quote Flajolet, Gardy and Thimonier [24, p. 224] where they refer to F. Baskett and A. Rafii [*The A0 inversion model of program paging behaviour*, Report CS-76-579 (1976), Stanford University, Stanford, CA] who by introducing *virtual* probabilities (computed according to their so-called A0 inversion model) in the model showed that it is possible to obtain 'excellent agreement between observed and predicted performance. Therefore, ... a system with *dependent* references can be modeled accurately by the *independent* reference model with modified access probabilities.'

1.11.1.4 Time-dependent request probabilities

Another objection to the simple unknown constant request probabilities model is that in reality, the popularities are probably *not* constant but rather change in time. To the best of my knowledge, nothing has been published on this problem (although remarks have been made that move-to-front is especially well suited for this situation) and Chapter 4, where some new results are presented, is a main point of this work.

1.11.1.5 Transposition rule

In his paper of 1965, McCabe [57] also introduced a less drastic self-organizing scheme than move-to-front — the *transposition rule*: When a record has been requested, move it one step closer to the front (in other words, transpose the requested record and its left neighbour). Throughout the years it has been extensively discussed whether the transposition rule should be preferred to the move-to-front rule. This suspicion is motivated as follows. The drawback of move-to-front is that when an impopular record is requested, it is still moved all the way to the front increasing the cost of many future requests to more popular records before it has been shifted far back again. This implies that move-to-front converges quickly but has rather large stationary cost. In contrast, the transposition rule is more stable — a record must be requested frequently to reach the front of the list and hence this rule converges slower but should keep closer to optimum in stationarity. In fact, in 1979 Bitner [5] showed that move-to-front indeed does converge much more quickly than the transposition rule. Although Bitner found no general form to measure the transposition rule's rate of convergence, he compared the two methods for several simple distributions and was led to the conclusion that the expected cost of the transposition rule decreases so slowly that even though its expected stationary cost is lower than that of move-to-front, move-to-front is still the better choice if few ($O(n^2)$) requests are made.

As for move-to-front under the same assumptions, the expected search cost under the transposition rule converges in probability to the expected stationary search cost when the requests are taken to be independent and the request probabilities constant. However, the transposition rule case is more complicated than the move-to-front one and McCabe [57] did not find any closed-form expression for this limit but conjectured that (in the long run) the transposition rule is at least as good as move-to-front. In 1976, Rivest [65] gave an expression for the expected stationary search cost under the transposition rule and showed that it is less than (or equal to) the expected search cost for move-to-front for all choices of constant request probabilities $p_1, \ldots, p_n$. Furthermore, he reported that Andrew Yao had shown that *if* an optimal memory-free permutation rule exists, then it is the transposition rule.

However, in 1982, Anderson, Nash and Weber [3] gave a counterexample to the conjecture that the transposition rule has the smallest asymptotic search cost among all memory-free heuristics in which they produced a rule which under certain p_i's has a smaller asymptotic expected search cost than the transposition rule. This rule however is a complex one which not only moves the requested record but also changes the relative ordering of the others. It further depends on the number of records in the list. The conclusion is that no optimal memory-free rule exists. Whether the transposition rule is optimal in the class of rules which just move the requested record some number of positions closer to the front is still an open question. (Some further papers that deal with the optimality of the transposition rule within different classes of rules are Kan and Ross [39] and Tenenbaum and Nemes [74].)

In the case of dependent requests the transposition rule cannot be the optimal. In 1984, Lam et al [47] gave numerical examples where move-to-front performs better.

1.11.1.6 Amortized analysis and move-to-front 'optimality'

While theorists have discussed which method to use, practitioners have long settled for move-to-front. In 1985, results based on *amortized analysis* were published supporting their decision. In such an analysis the rule is investigated in a worst-case sense, but since the worst-case behaviour for a single request is trivial (we must search the entire list), the worst-case number of comparisons made for *any particular sequence* of requests is considered. Usually, in the legal context, to 'amortize' is to end a debt by making regular payments into a special fund. In this context, the word hints at the cost being distributed over a series of requests. Bentley and McGeoch [4] showed that the total cost for move-to-front is at most twice the cost for the optimal static ordering. In fact, Sleator and Tarjan [70] showed that within a constant factor (less than 2), move-to-front is as efficient as *any* other heuristic, including algorithms based on knowledge of future requests in the sequence. Irani [38] improved the upper bound on move-to-front's cost to $2n/(n+1)$ times that of optimal order. Combining this with the result due to R. Karp and P. Raghavan (communicated by Irani [38]) stating that for any algorithm which reorders the records deterministically (such as move-to-front) there exists a request sequence such that $2n/(n+1)$ times the cost of optimal order is a lower bound for the search cost, he concluded that move-to-front in fact achieves the best possible ratio between its cost and that of the optimal static order for every list of length n.

The transposition rule does not share this 'optimality'. In an empirical study, Bentley and McGeoch [4] concluded that neither the above probabilistic analysis nor their amortized approach describes the behaviour completely realistically. In their application, the transposition rule is clearly not the best heuristic (as was predicted in terms of asymptotic search cost) and yet it never performed as badly as predicted by the amortized results.

1.11.1.7 Related rules

Several variants and competitors to the move-to-front rule have been suggested. Here is a brief overview.

Memory-free rules

Rivest [65] introduced the *move-up-k* rule: if the requested record is in position 1 through $k+1$, move it to the front, otherwise move it k positions closer to the front. The expected asymptotic search cost for move-up-k is surmised to be less than (or equal to) the cost of move-up-$(k+1)$ supported by simulation results.

Gonnet, Munro and Suwanda [30] conjectured that for $j > k$, the expected stationary search cost of move-up-k is less than that of move-up-j. On the other hand, the former converges slower than the latter. Simulation results by Tenenbaum [73] support this conjecture.

Nelson [59] considered the Markov chain for a *move-to-k*-rule (where move-to-front reappears when $k = 1$) and concludes that when $k > 1$, the chain is never positive recurrent.

Counter scheme

One of the advantages of the move-to-front rule is that it requires no extra memory. Disregarding the memory restriction, the *counter scheme*, where each record is associated with a frequency counter and the list is kept in non-increasing order of their request frequencies, proves to be a well-performing rule. Indeed, Hofri and Shachnai [35] proved the counter scheme to be optimal with respect to the expected search cost at the mth request, for any $m \geq 1$, among all rules that use no advance information about the p_i's (assumed to be unknown constants) and have no knowledge about future requests. Algorithms that use no information about future requests are called *on-line*. Conversely, an algorithm is *off-line* if it can see the entire request sequence before servicing. Hofri and Shachnai remarked that it is noteworthy that such an optimal scheme does exist since among the memory-free rules none is optimal. Further, Hofri and Shachnai [36] presented ways to bound the counter scheme's memory space requirements without damaging the performance and set a limit on the time required to implement this scheme. They determined the expected search cost and considered its convergence as the number of requests grows.

Meta-algorithms

In order to slow down the convergence, reducing the effects of rare requests and the time spent reorganizing, the meta-algorithm at each request decides whether to move (according to some pre-specified rule such as move-to-front or the transposition rule) or leave the list unchanged.

The most straightforward modification of the move-to-front rule into a meta-algorithm is the one suggested by McCabe [57] — move only once every kth request to the list. The stationary search cost is the same as for ordinary move-to-front but k times as many requests are needed for the convergence. This rule is not memory-free since it requires the maintenance of a counter in order to keep track of the number of requests made (to determine when to move). The same goes for all the rules in this subsection. However, although some of the rules also require additional memory space to maintain pointers, they use significantly less space than the counter scheme above.

Gonnet et al [30], Kan and Ross [39] and Phatarfod [62] have all considered *k-in-a-row* rules where a record is moved only when it has been requested k times in a row. Kan and Ross showed that as k tends to infinity, the proportion of time that the ith most popular item spends in the ith position of the list will tend to one.

Alternatively, Gonnet et al suggested that the requests be batched into groups of k requests where a move is made only if all k requests in a batch is for the same record. They proved that *batch-k* has lower expected stationary search cost than k-in-a-row.

Bitner [5] combined the k-in-a-row idea with the use of counters. Assign to each record a counter counting the number of requests to this record. In the *wait-c-then-move* rules, a record is moved after it has been requested c times (not necessarily in a row) and its counter is then reset. *Wait-c-then-move-and-clear* works the same way only now *all* counters are reset after each move.

Randomized rules

One can also randomize in order to make less frequent moves. Kan and Ross [39] proposed a randomized version of k-in-a-row: move the requested record i with probability α_i and leave the list unchanged with probability $1 - \alpha_i$, $i = 1, \ldots, n$, $0 \leq \alpha_i \leq 1$. If all the α_i's are equal, the expected stationary search cost will be identical to that of the original, non-randomized rule. Generally, the cost will depend on the α_i-values. However, the transposition rule turns out to be an exception — the results in this case are independent of the randomization constants. Assuming the underlying rule to be move-to-front, Phatarfod [62] derived eigenvalues and their multiplicities and concluded that the convergence to stationarity for this scheme is slower than that of ordinary, non-randomized move-to-front.

Oommen and Hansen [60], [61] considered a similar scheme in which on the n:th access of record i, it is moved to the front with systematically *decreasing* probability α_i. Hence, ultimately the moves tend to cease and the Markovian representation of the records' permutations is absorbing rather than ergodic. They found that this scheme is 'expedient but always worse than ordinary move-to-front'. They also investigated a stochastic move-to-rear strategy in the same manner, in which seldomly requested records are expected to keep moving at later times and therefore will end up at the rear of the list. This latter scheme is optimal in the sense that the probability of the list converging to the optimal arrangement can be made arbitrarily close to one by making the α_i's decrease in a suitable way. Hofri and Shachnai [35] discussed the (dis)advantages of this method in comparison with the counter scheme.

Irani [38] presented another randomized move-to-front algorithm, called SPLIT. In this algorithm, every item in the list maintains a pointer pointing to another item in the list. If item i is requested, move it to the front with probability $1/2$ and with probability $1/2$ insert it before the item pointed to by i's pointer. Item i's pointer is then set to point to the first record. For arbitrary request sequences, the expected cost of SPLIT is shown to be within $31/16$ of that of the optimal.

Kapoor and Reingold [40] randomized move-to-front and move-to-root (Section 1.11.2 below) to reduce the number of data moves. Measuring time in discrete steps t, each step corresponding to a request, the randomized algorithms rearrange the records only with probability $1/f(t)$, where $f(t)$ is a real function of t. On basis of the discrete time Markov chain of permutations (trees) for ordinary move-to-front (move-to-root), the randomized rule is associated with an analogous Markov model. The stationary distribution stays the same (under some weak condition) and so does the expectation and variance, however for dependent requests the cost under the randomized rules differ from that of the original ones. The expected search cost at time-step t and the number of moves up to this time is given for both the randomized move-to-front and move-to-root. Convergence is slower but the number of data arrangements is reduced, which is good if the cost of a data move is not negligible.

Weighted list updating

d'Amore, Marchetti-Spaccamela and Nanni [2] wrote a paper on the *weighted* list updating problem. Now, the cost of searching is not only determined by the number of comparisons needed. Instead, to each record a weight is associated so that the cost of requesting a certain record depends on which record it is, not only its position in the list. In this case, the optimal static ordering is such that the records are arranged in decreasing (at least non-increasing) order of the ratio between the number of occurrences of the current record in the whole request sequence and the record's weight. In contrast to the results above, the cost of move-to-front when such weights are considered is *not* within a constant factor of the cost for the optimal static ordering. However, two modified rules were presented by d'Amore et al, both of which do have stationary costs within a factor 2 of that of the optimal static arrangement. These are the *counting-* and the *random move-to-front* rule. The former uses counters to determine when to move a record to the front: When record i is requested, increase its counter by d/w_i, where w_i is the associated weight and d is a constant optimally chosen as $d = \min_j w_j$. If the counter reaches or exceeds 1, the record is moved to the front and 1 is subtracted form the counter. The latter rule uses biased coins instead of counters: When record i is requested move it to the front with probability d/w_i (d and w_i as defined above).

Hybrids

Different rules are preferable in different situations. Attempting to combine the fortes of different rules, hybrids have been proposed.

Bitner [5] suggested a combination of move-to-front and the transposition rule where move-to-front operates until stationarity is reached. Then the hybrid switches to the transposition rule. This is to utilize the fast convergence of move-to-front combined with the steadier stationary behaviour of the transposition rule.

A hybrid of the transposition rule and the counter scheme was suggested by Lam [46]. Here the transposition rule rules until stationarity where the counter scheme takes over. The intent is to avoid the mistakes likely to be made by the counter scheme in the beginning of the request sequence.

Tenenbaum and Nemes [74] suggested the methods POS(k) (if the requested record is in position $k+1$ through n, move it to position k, otherwise transpose it with the preceding record in the list) and SWITCH(k) (if the requested record is in position 1 through k, move it to the front, otherwise transpose it with its predecessor). In particular, POS(1) and SWITCH(n) equal move-to-front while POS($n-1$) and SWITCH(2) equal the transposition rule. When $p_1 = p$ and $p_i = (1-p)/(n-1), i = 2, \ldots, n$, the cost of POS($k$) is greater than or equal to the cost of POS($k+1$) while the cost of SWITCH(k) is less than (or equal to) the cost of SWITCH(k).

POS(k) and SWITCH(k) are both special cases of what Lam [46] calls *move-forward* rules. When i is requested, move it to position l_i, $i = 1, \ldots, n$. Lam provided partial orderings within the class of move-forward rules. He gave stationary probabilities for the list permutations (states in the Markov chain mentioned above) and compared costs when $p_i = (1-p)/(n-1)$, $i = 2, \ldots, n$, and $p_1 = p$.

Utilizing dependence

Assuming Markov dependent requests, Konneker and Varol [45] suggested some modified heuristics using the dependence, for instance the *backup search ordering*: Suppose that the transition probability p_{ij} in a $n \times n$ Markov matrix is the conditional probability of a request for record j following one for i. If x is the record currently being searched and i is the most recently requested one, then start by comparing with record j, where j is such that $\max_j p_{ij}$ and after that according to decreasing order of the unconditional request probabilities.

In addition, extensions of move-to-front to the case of doubly linked lists are considered, for instance by Matthews, Rotem and Bretholz [55] and Valiveti and Oommen [76], while Valiveti, Oommen and Zgierski [77] considered move-to-front ‘when the individual queries consist of accesses to a subset of the elements stored, as opposed to the individual elements themselves’.

1.11.2 Move-to-root

Is the method of sequential search always the appropriate one? No, as McCabe [57] remarked, it is not. In order to utilize the lexicographical order between the stored items we may instead choose a *binary search tree* storage structure (cf. Section 1).

1.11.2.1 Binary search tree

A binary tree is either empty or a root node with a left and a right binary subtree. To make it a search tree we enforce an order between the items in the nodes: All records with smaller key values are in the left subtree while all records with greater (or equal) key values are in the right. Since we assume all key values to be distinct, there will be no equal keys. When constructing a binary search tree, begin by selecting a record which is put in the root node. Select a second record. If its key value is less than that of the record in the root, put the second one in the left subtree, otherwise put it in the right subtree. Select a third record and repeat the procedure. Figure 1.11.1 shows an example of a binary search tree where the records have been inserted in the order 7,2,5,8,4,1,6,3 of their key values. Several other permutations yield the same tree. In fact, there are

$$\binom{2n}{n}/(n+1) \tag{1.1}$$

(the Catalan numbers, Knuth [43, p. 434]) possible binary search trees consisting of n nodes. Compare this to $n!$ — the number of possible permutations of the same items.

Consequent on the tree insertion algorithm, *inorder* or *systematic traversal* — recursively visit the left subtree, then visit the root, then visit the right subtree — produces the records in ascending (or lexicographical) order. (There are several alternative ways to traverse — systematically visit every node in — a tree. See for instance Sedgewick [68, pp. 44-49].) This property can actually be taken as an alternative definition — if inorder traversal yields the items in order, then the structure is a binary search tree. When a record in this tree is requested, we start looking for it in the root, working our way down through the tree exactly as when we constructed it. If the requested record is equal to the root we are done, if it is less than the root, compare it to the root of the left subtree and otherwise compare it to the root of the right subtree. The *search cost*, the number of comparisons

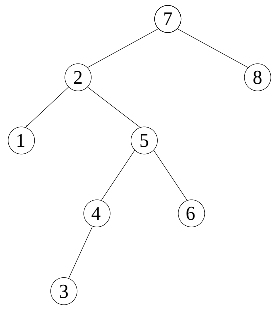


Figure 1.11.1: *Binary search tree.* The nodes have been inserted in the order 7,2,5,8,4,1,6,3 (of their key values).

needed to find the requested record, equals the number of records we pass on our path down from the root to the record we request. In the mathematical terminology of trees these are called the *ancestors* of the requested records in analogy with genealogical trees.

Knuth [43] has traced this method back to 1946 when it was mentioned by Mauchly [56] in the first (published) discussion of computer sorting.

1.11.2.2 Move-to-root

Given constant independent request probabilities, one can construct an optimal binary search tree (a tree of minimum cost), Knuth [43, pp. 433-477], but what if the popularities are not known? In analogy with move-to-front, in 1978 Allen and Munro [1] suggested that in this case whenever a record has been requested it should be moved to the root. The trick is to move a record to the root without destroying the order between the records in the process. Allen and Munro solved this problem by performing a series of *simple exchanges* (analogous to the transposition rule) in which the requested record is moved up one level in the tree disturbing the rest of the tree as little as possible yet preserving the binary search tree property. The simple exchanges are performed until the requested record has been made root of the entire tree. This is the *move-to-root* rule. As for move-to-front, the point is that when move-to-root has been operating for some time, frequently requested records will automatically have gathered near the beginning of the structure, that is in this case, close to the root. In Section 5.2 we describe in detail how the simple exchanges and consequently move-to-front work.

Throughout this monograph, we shall assume that a requested record is always in the tree. For the case of missing records, see for instance Allen and Munro [1] and Mahmoud [52].

1.11.2.3 Optimality

Allen and Munro [1] assumed the records to be requested independently, with unknown constant probabilities. They derived the expected stationary search cost of move-to-root (cf. p. 31) and compared it with that of the optimal static binary search tree (although the cost for the optimal search tree is not known on closed algebraic form) and found that the cost for move-to-root is within a constant factor of that of the optimal. In particular, when the request probabilities are equal ($p_i = 1/n, i = 1, \ldots, n$) the expected stationary cost $\simeq (2 \ln 2) \log n$, Knuth [43, pp. 74-75], whereas the cost for an optimal balanced tree $\simeq \log n$. So in this case the bound is essentially achieved. Note that the optimal tree is not necessarily the tree which has the most popular record in its root, and both subtrees recursively ordered in the same way etc. A tree defined by this property is instead a *monotonic tree*, considered in 1979 by Bitner [5], and it may, if we are unlucky, become a list. This happens for instance if $p_1 > p_2 > \cdots > p_n$. Then if the p_i's are approximately equal the monotonic tree has a much higher cost than the optimal one which is more balanced. This is because when the records are almost equally popular the shape of the tree is much more important for the cost of searching than the location of the most frequently requested record. On the other hand, if the probabilities are not that close, high-probability records will be put near the root and the monotonic tree rule will perform nearly optimally.

In the present tree case, we need not worry about the possibility that the less drastic simple exchange scheme, moving the requested record only one step closer to the root, might actually do better than move-to-root (cf. the transposition rule vs. move-to-front discussion above). As Allen and Munro [1] concluded, no single memory-free tree heuristic is ever uniformly better than all other such heuristics. In particular, the simple exchange scheme is not uniformly better than move-to-root for all sets of p_i's just as move-to-root is not uniformly better than the simple exchange scheme.

1.11.2.4 Random permutation model and equal request probabilities

Bitner [5] also considered *random trees* and showed that the expected search cost for a random tree is greater than (or equal to) that of the monotonic tree. A random tree is a tree built by successively inserting the n records where each of the $n!$ insertion sequences (permutations of $1, \dots, n$) are considered equally likely. Note that the class of random trees is quite distinct form the class obtained by letting the different possible trees be equally likely. The random tree model is also known as the *random permutation model* and results concerning it, including those mentioned below, are presented by Mahmoud [52, Chapter 2] in his book of 1992 on random search trees. Given any set of (constant) probabilities $p_1, \dots, p_n$, assuming each of the $n!$ assignments of probabilities to records to be equally likely, Bitner derived the expected search cost of a random tree. It agrees with the result we get if we assume all the records to be equally popular (cf. Corollary 6.1.2.4 below). The formula appears for the first time in papers by Windley [78], Booth and Colin [9] and Hibbard [34] who seem to have developed the method of tree insertion independently (see Knuth[43, Section 6.2.2]). The expression for the variance of the search cost when all the records are equally popular seems to have been given first by G. D. Knott in an unpublished paper, Knuth [43, Exercise 6.2.2/6, p. 672]. In 1984, probability function and generating function for the stationary search cost were given by Brown and Shubert [10] (see Louchard [51, equation (13)], Mahmoud [52, Section 2.5] and Section 6.2.3 below for alternative proofs). Brown and Shubert also studied the *height*, the greatest occupied level or the number of ancestors in the longest path from the root to a leaf (a node with no children), and vacancies.

For large n, the expectation and variance of the search cost (still assuming equal popularities) are of the same order and properly rescaled, the search cost converges to a constant in probability ($S_n/\ln n \xrightarrow{P} 2$) and to a standard normal distribution in law. The latter result is due to Louchard [51] who established it in 1987 using the generating function.

In principle, the results as stated above have nothing to do with the move-to-root rule. However, when the request probabilities are equal, move-to-root has no effect on the cost and the expectation, variance and probability function for the stationary search cost coincide with those for the random permutation model.

1.11.2.5 Independent requests

In addition to the expected stationary search cost of move-to-root and an expression for the variance, Allen and Munro [1] also gave the rate of convergence to stationarity for the expected search cost, starting with a random tree.

The simple exchange (and move-to-root) process can be viewed as a finite Markov chain with one state for each of the possible binary search trees. Still assuming that requests are made independently at each discrete unit of time with constant request probabilities, in 1995 Dobrow and Fill [16] mapped the set of permutations to the set of binary search trees (a many-to-one mapping) and showed that the transition probabilities for the move-to-root Markov chain of trees can be obtained by lumping the corresponding move-to-front Markov chain of permutations. The stationary distribution of the move-to-root chain was given explicitly and the t-step transition probabilities were given on a more 'tree-based' and 'computation-friendly' form than just in terms of the lumped move-to-front chain. Finally, they gave the eigenvalues for move-to-root and their multiplicities.

In another paper of 1995, Dobrow and Fill [17] considered rates of convergence to stationarity for the move-to-root Markov chain and showed that for many examples (uniform, geometric, generalized Zipf's and power law request probabilities, cf. Section 3.5), move-to-root converges twice to four times as fast as move-to-front. They also studied rates of convergence for the expected search cost.

Considering only the special case of equal request probabilities, Fill [20] in 1996 presented some further results on the distribution of the stationary move-to-root Markov chain — for each possible shape of the tree, he considered the stationary probability that the tree looks like this. Note that the stationary chain is identical to the one arising in the random permutation model. Fill found the trees for which the stationary probabilities are minimal or maximal and determined the corresponding 'extreme' probabilities. Further, to establish 'typical values' of the probability distribution, he assumed the trees to be (a) uniformly distributed among all possible binary search trees and (b) that the trees themselves are distributed according to the stationary distribution he studied. These distributions were chosen in order to define what 'typical' trees are. For each of the two suggestions of tree distributions, the law of large numbers was established.

1.11.2.6 Related rules

In addition to the monotonic tree already mentioned, Bitner [5] considered the *limited single rotation*, *total single rotation* and *double rotation* rules. Given the request frequencies of the records so far, the single rotation rule rotates (perform a simple exchange) an ancestor (passed on the path to the requested node) one level closer to the root if the number of requests for records which would be moved up by the rotation exceeds the number of those which would be moved down. The total single rotation rule and the double rotation rule are more sophisticated variations on this theme.

Sleator and Tarjan [71] introduced *splay trees* — self adjusting binary search trees similar to move-to-root in that it moves a requested record to the root. The difference is that in splaying, the rotations are made in pairs in an order depending on the structure of the access path. Splay trees have also been studied by Mäkinen [58] and Schoenmakers [69].

1.11.3 Population genetics

Another angle of approach to the move-to-front rule is via the field of population genetics. The aim is to describe genetic evolution and there are a number of different models for the transmission of hereditary genetic material from parent to offspring. The introduction of a new model, the *neo-Darwinian synthesis*, including a stochastic element as opposed to the traditional
deterministic pattern, is referred to by Ewens [19] in 1990 as 'one of the great scientific achievements of the first half of this century'. In this model, one of two genes is randomly chosen to be passed on to the offspring and it also takes into consideration the fact that two genetically identical individuals may not have the same number of offsprings.

Genes are classified in terms of *alleles.* Genes coding for proteins that basically serve the same purpose (for example determines the length or hair colour of the individual) but which may give the individual different properties are called alleles. An individual with two identical alleles of a gene is called a *homozygote* while an individual with different alleles is called a *heterozygote.*

For the homozygote, the properties (length, hair colour) of the individual, the *fenotype*, is a direct reflection of the identical alleles, the *genotype.* Since the heterozygote has different alleles, they may give rise to new combinations of properties (length, hair colour), depending on whether any of the alleles involved is dominant or recessive.

Much interest has been taken in the changes of the frequencies of different alleles at some gene locus (position in the DNA). For instance, a model allowing some fixed number k of possible alleles has been considered by Ewens [19, Section 3]. In the symmetric mutation case, where a gene of one allelic type mutates to any one of the other possible allelic types with equal probability, the stationary joint distribution of the allelic frequencies $p_1, \ldots, p_{k-1}$ (p_k is determined by the others so that $p_k = 1 - p_1 - \cdots - p_{k-1}$) may be derived and turns out to be the Dirichlet distribution.

Population genetics and move-to-front are connected via the concept of *size-biased permutations*: Choose an item i_1 among $1, \ldots, n$ at random in such a way that $i_1 := i$ with probability p_i. Let p_{i_1} be the first component of the size-biased permutation. Then choose another item i_2 from the remaining ones $1, \ldots, i_1 - 1, i_1 + 1, \ldots, n$ so that $i_2 := i$ ($\neq i_1$) with probability $p_i/(1 - p_{i_1})$ and let p_{i_2} be the second component, etc.

In a list organized according to the move-to-front rule, the first item is i with probability p_i. Hence, if we label the items $1, 2, 3, \ldots$ beginning at the front successively working our way through the list, we get a size-biased labelling.

It can be shown (Ewens [19, pp. 213-215]) that the distribution of allelic frequencies when labelled by size-biasing is identical to the one when labelled by age. This is due to time reversibility, see Ewens [19, p. 211]) which guarantees that the distribution when labelled by age is the same as the distribution when labelled by the length of future persistence in the population.

In 1991, Donnelly [18] observed that instead of the current permutation of the items in the list $(i_1, i_2, \ldots)$ one may consider the current permutation of their corresponding *popularities* $(p_{i_1}, p_{i_2}, \ldots)$. He showed that, given the initial permutation of popularities μ, the Markov chain of permutations converges in distribution to the size-biased permutation of μ. He also showed that invariance (for definitions and details, see Donnelly [18, Section 3]) under size-biased permutation is equivalent to invariance under the move-to-front operation. Furthermore, he generalized the move-to-front rule in order to model partial size-biased permutations — instead of just moving one item to the front, choose j items without replacement and move them all to the front in the relative order in which they were chosen. Finally, limiting results when the initial permutation of popularities corresponds to the Poisson-Dirichlet distribution were given. The results recapture and extend those obtained by Kingman [42] in 1975.

Part I

Move-to-front for two users

It takes all sorts (to make a world).

Idiom

Part I of this monograph may itself be viewed as divided into two parts. In Chapter 2 – 3 we extend the move-to-front model to the case where two users with different priorities, preferences (and request frequencies) share the same data. The underlying probabilistic model is the basic one assuming independent requests and unknown constant request probabilities. We derive the probability distribution at finite time and in stationarity for the low-priority user's search cost and study the convergence. As a consequence, known single-user results are recaptured.

In Chapter 4 we introduce a different probabilistic model allowing the request probabilities to change in time. The requests are still assumed to be independent. This time-dependence is a new approach to the problem of the behaviour of the move-to-front rule and sheds some new light also over properties of the standard single-user case.

The results in Chapter 2 may all be extended to the case of time-dependent requests. As the number of records tends to infinity, the distribution function for the fraction of records needed to be checked before the requested one is found may be given explicitly on rather a handy form. As an illustration, the results are applied to some examples. We also study stochastic ordering for this asymptotic search cost.

Chapter 2

Search cost distribution

All previous work on the move-to-front rule concerns either the order of the items in the list (the permutation chain) or the cost of searching the item next to be moved. Now, what if *another* user is involved, one who is allowed to use the information (records in the database/items in the list) but who does not get to rearrange the items according to his taste? What is the cost of searching for this second user? This is our new angle.

The extension to two users with different priorities is not purely academic but answers to a rather common situation in applications, see Section 1.8.

Assuming independent requests and constant request probabilities, we generalize the results on the search cost for the 'ordinary' move-to-front rule given by Fill and Holst [23] to the case of two users. Using methods similar to theirs, we find the transient as well as the stationary distribution, expectation and variance for the second user's search cost. The known single-user formulas follow.

In order to make the description of the move-to-front rule for two users self-contained, we repeat the definition.

2.1 Move-to-front for two users

Suppose that there are n records (units of information) in a file or a database. Each record has an identifying *key*, with a *key value*, for use in searching, cf. Section 1.8. The records are stored in a list organized in the following manner:

When a record is requested, its key value is compared sequentially to the keys of the records in the list. When the requested record has been found, it is moved from the current position in the list to be replaced in front of all the others. This is the *move-to-front* rule — a self-organizing search strategy which maintains fast access by keeping frequently requested records at the beginning of the list. One may think of this as books in a shelf being moved to the front after having been read or books in a heap[1] where the last read book is put on top of the others, cf. Chapter 1.

The usefulness of move-to-front is due to its dynamic properties — it is self-adjusting and therefore especially suitable when the request probabilities are unknown or time-dependent.

Throughout this monograph we shall assume that the keys are distinct. Hence, without loss of generality, we label the keys $1, \dots, n$. The corresponding record is assigned the same label as its key. Furthermore, we assume that a request is always serviced — there are no blocking times and the requested record is always found.

Suppose now that two users share the same data. The high-priority 'Controller' is the owner of the list and requests records at the rate of one per unit of time. Whenever Controller makes a request, it is for record i with probability $p_i > 0$, $(\sum_{i=1}^n p_i = 1)$ independently of all previous requests. After having been found and read, the record is moved to the front of the list.

The 'Second User', who also requests the records in the list, but at a possibly different rate λ per unit of time, might have other preferences. On a request, Second User calls record i with probability q_i, $(\sum_{i=1}^n q_i = 1)$ independently of all previous requests (including requests made by Controller). However, this second, low-priority user must *not* rearrange the list according to his preferences.

Using what is known about the move-to-front rule for a single user, analogous results can be obtained for the case of two users. We shall study *Second User's search cost* defined as the number of records in front of the one requested by Second User.

As we assume $p_i > 0$ for every $i = 1, \dots, n$, we disregard the possibility of records with zero probability of being requested by Controller. If such records were present, they would successively be shifted back towards the rear of the list, keeping their initial relative order. In the single-user case, these records would certainly never be requested (since Controller is the only user present) and the stationary cost (cf. for instance Section 2.3.2) would be the same as for a list consisting only of those records with positive

[1] Hence, move-to-front is also known as the *heaps process*, cf. p. 17.

request probability. Introducing a second user, allowed to request records such as these, would result in a search cost that always, even after the rest of the list (records with positive Controller request probabilities) has reached stationarity, depends on the initial arrangement. As this would complicate our expressions without contributing any additional principal understanding, we shall confine ourself to the case when Second User requests (possibly a subset of) records with positive Controller request probabilities.

The search cost concept was originally introduced by McCabe [57] in 1965, cf. Section 1.11, as the position of the requested record in the list. The intuitive appeal of this definition is that the search cost then equals the number of comparisons needed in order to find a desired record. However, formulas concerning this quantity have turned out to become slightly less complicated for an alternative (and equivalent) definition: Let the search cost be the number of records *in front of* a requested one, that is, the number of necessary comparisons minus one. By convention, this definition is used when discussing the move-to-front rule and the original one when considering the tree analogue, cf. Part II.

2.2 Poisson embedding

The move-to-front rule (for a single user) has often been studied assuming that the requests are made at each (discrete) unit of time, in other words, 'time' has been measured by the number of requests to the list. However, the problem of finding for example properties of the probability distribution for the search cost simplifies considerably if the technique of Poisson embedding is applied (cf. Fill and Holst [23]) and we shall use this idea in the case of two users. The situation we model in this way is the common one where the request probabilities of the different records are unknown constants and the requests are made independently of each other.

Imagine that Controller requests records at time-points according to a Poisson process of intensity one. To each time-point the label i is assigned with probability $p_i > 0$, independently of the labels of other time-points. This generates Controller's sequence of requests.

Considering only the requests that have been assigned the label i, these occur at the time-points of a Poisson process of intensity p_i. In fact, the embedding defines n independent Poisson processes with intensities $p_1, \ldots, p_n$ respectively (see Figure 2.2.1), where at the time-points of the i:th process record i is requested and subsequently moved to the front.

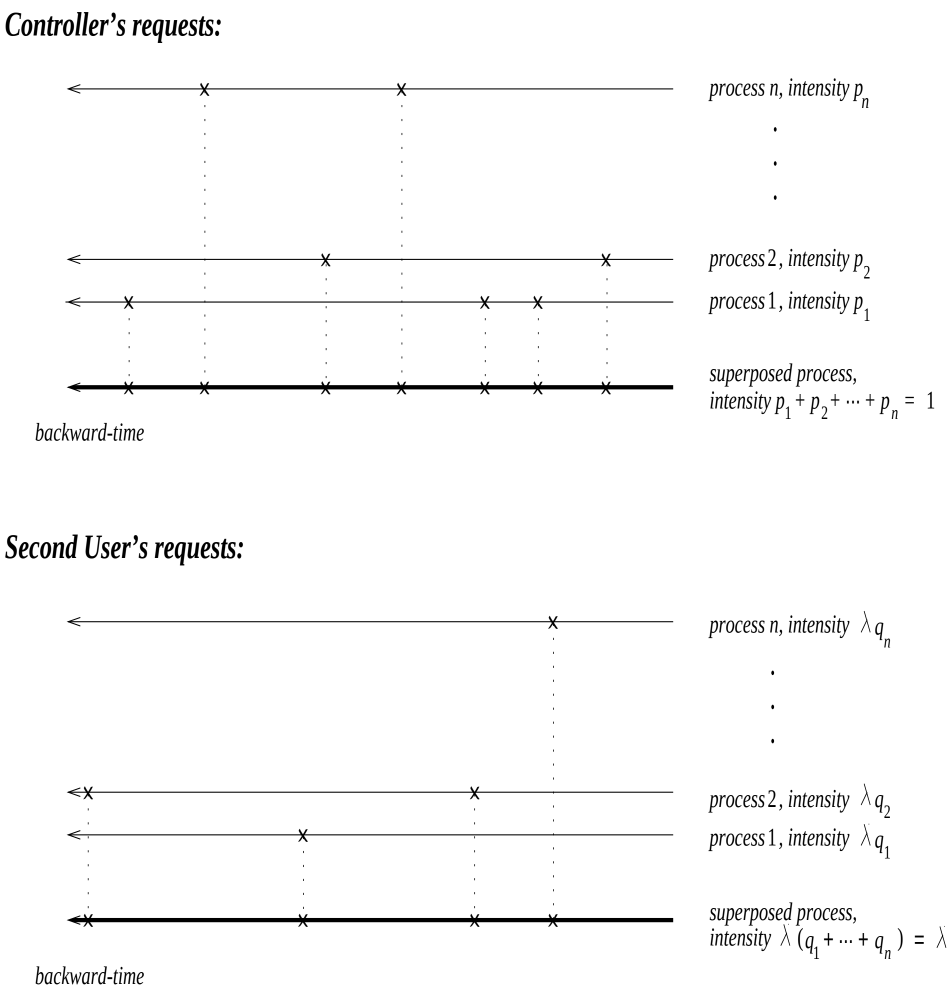


Figure 2.2.1: *Poisson embedding.* Controller requests record i with probability $p_i > 0$ at the rate of one per unit of time while Second User requests record i with probability q_i at the rate of λ per unit of time.

Similarly, Second User makes requests at time-points according to a Poisson process of intensity λ, independently of Controller's requests, and to each of these time-points the label i is assigned with probability q_i, independently of the labels of other time-points.

We shall regard these processes as going *backwards* in time[2] starting at some time t ('now'). Let T_i be the time elapsed since the most recent occurrence of an event in process i. (Note that this event may have occurred before time 0.) Alternatively, T_i may be said to be the backward waiting time until the 'first' occurrence in this process. Then, by the embedding, $\{T_i\}_{i=1}^n$ are independent exponential random variables with mean $1/p_i$.

Analogously, if T_i^{SU} denotes the time since the most recent occurrence in the Poisson process according to which Second User requests record i, then $\{T_i^{SU}\}_{i=1}^n$ are exponential random variables with mean $1/\lambda q_i$, independent of each other as well as of the T_i's.

In what follows, we shall study Second User's search cost on a request, that is, we *know* that Second User makes a request at the current instant. The cost depends only on the list arrangement which in turn is determined solely by Controller's requests — previous requests by Second User do not effect the permutation. Hence, we shall have no need for the Poisson process model for Second User's request and, consequently, our results will be independent of λ. Nevertheless, this model provides means to express unconditional properties of Second User's search cost and we stress the point that we make no restriction on the request frequencies.

2.3 Second User's search cost

Let $S(t)$ denote the search cost for Second User at time t, that is, if Second User makes a request at this time, $S(t)$ is the number of records in front of the requested one. Note that the search cost depends on n, the number of records. However, for ease of notation, we suppress this dependence.

In this section, we derive the distribution, expectation and variance of Second User's search cost, first at time t given the initial permutation of the list (Theorem 2.3.1.2 - 2.3.1.3) and then in stationarity (Corollary 2.3.2.1 - 2.3.2.2). The corresponding results for the single-user case are collected in Corollary 2.3.1.4 - 2.3.1.5 and Corollary 2.3.2.3 - 2.3.2.4.

[2] For clarity, we shall assign the superscript * (e.g. t^*, u^*) to instants in *backward* time to distinguish them from instants in ordinary forward time (e.g. t, u).

2.3.1 Search cost as a function of time

Introduce the indicator random variables

$$\mathbf{1}_{ij}(t) \quad = \begin{cases} 1 & \text{if record } i \text{ precedes record } j \text{ in the list at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

Note that for all $t > 0$, $j \neq i$ and $k \neq i, j$, $\mathbf{1}_{ji}(t)$ is independent of $\mathbf{1}_{ki}(t)$. Then, the cost of searching record i at time t is

$$S_i(t) = \sum_{j:j\neq i} \mathbf{1}_{ji}(t), \tag{2.1}$$

and the probability function for the search cost at time t becomes

$$P(S(t) = k) = \sum_{i=1}^{n} q_i P(S_i(t) = k).$$

Initially, at time 0, the list is in permutation π. Now, given that Second User requests record k at time t, let $S_k^\pi(t)$ be the number of records in front of record k at this time in a list started in permutation π. $S^\pi(t)$ denotes the search cost at time t given that Second User makes any request at time t, that is, $S^\pi(t) = \sum_k q_k S_k^\pi(t)$. Let $\oplus_j F_j$ denote the distribution of the sum of independent random variables with respective distributions F_j. Then, the distribution of $S_i^\pi(t)$ is given by Fill and Holst [23, Proposition 2.1], where we write Bern(p) for the Bernoulli distribution with success probability p.

Proposition 2.3.1.1 (Fill and Holst, 1996)

$$P(S_i^\pi(t) = k) = \int_0^t P_{u^*}(S_i^\pi(t) = k) p_i e^{-p_i u^*} du^* + P_t(S_i^\pi(t) = k) e^{-p_i t},$$

where $P_{u^*}(S_i^\pi(t) = k)$ *is the probability function of the distribution*

$$\begin{cases} \oplus_{j\neq i} Bern(1 - e^{-p_j u^*}) & \text{if } u^* \in (0, t) \\ \oplus_{j\neq i} Bern(1 - \mathbf{1}_{ij}(0) e^{-p_j t}) & \text{if } u^* = t. \end{cases}$$

Remark. Note that $\mathbf{1}_{ij}(0) = 1$ if and only if i precedes j in the initial permutation π and so $\mathbf{1}_{ij}(0)$ is deterministic.

The proof below is slightly different from the one in Fill and Holst [23].

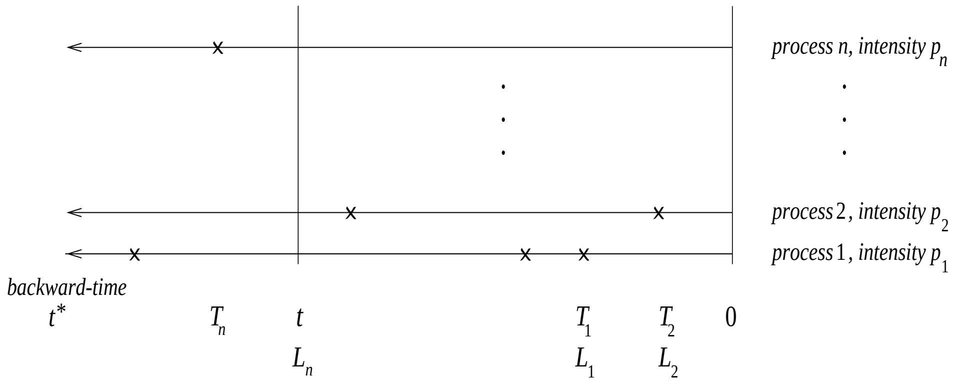


Figure 2.3.1: *Poisson embedding.* At time t, Second User makes a request. This defines the starting point of the backward-time. Controller requests according to independent Poisson processes evolving backwards in time. T_i is the (backward) waiting time until the first occurrence in process i while L_i is the 'actual' time elapsed since Controller last requested record i. If Controller has not requested this record since the clock was started (time t ago), then L_i is taken to be t, otherwise, L_i coincides with T_i.

Proof. For $i = 1, \ldots, n$, let L_i denote the time elapsed since Controller's last request for record i, see Figure 2.3.1. If Controller has never requested i then $L_i := t$. Since the move-to-front process has only been running for time t, this is the limit for how far back in time we can go. Hence, with the notation from Section 2.2, L_i is distributed as $\min\{T_i, t\}$.

Conditional on $L_i = u^*$, the event that record j precedes record i in the list splits into two cases:
(i) If $0 < u^* < t$, Controller has requested i at least once before time t and $\mathbf{1}_{ji}(t) = 1$ if and only if he has requested j after his most recent request for i. So, in this case,

$$P_{u^*}(\mathbf{1}_{ji}(t) = 1) = P(L_j < u^*) = 1 - e^{-p_j u^*}$$

by the embedding.
(ii) If $u^* = t$, Controller has never requested i. In this case $\mathbf{1}_{ji}(t) = 1$ if Controller has requested j or has *not* requested j but j preceded i already in the initial permutation. Hence,

$$P_t(\mathbf{1}_{ji}(t) = 1) = (1 - e^{-p_j t}) + \mathbf{1}_{ji}(0)e^{-p_j t} = 1 - \mathbf{1}_{ij}(0)e^{-p_j t}$$

and the result follows.

□

Theorem 2.3.1.2 *For a list started in permutation π the probability function for Second User's search cost at time t is*

$$P(S^\pi(t) = k) = \sum_{i=1}^{n} q_i P(S_i^\pi(t) = k), \tag{2.2}$$

where the distribution of $S_i^\pi(t)$ is as in Proposition 2.3.1.1.
Moreover, the expectation and variance at this time are

$$\begin{aligned} E(S^\pi(t)) \quad = \quad \sum_{1 \le i < j \le n} \Bigg[& \frac{q_i p_j + p_i q_j}{p_i + p_j} \left(1 - e^{-(p_i+p_j)t}\right) \\ & + \left(q_i \mathbf{1}_{ji}(0) + q_j \mathbf{1}_{ij}(0)\right) e^{-(p_i+p_j)t} \Bigg] \end{aligned}$$

and

$$
\begin{aligned}
\operatorname{Var}(S^\pi(t)) &= E(S^\pi(t))\,(1 - E(S^\pi(t))) \\
+ \quad 2 & \sum_{1 \le i < j < k \le n} \left[\frac{1}{p_i + p_j + p_k} \left(\frac{(q_i p_j + p_i q_j) p_k}{p_i + p_j} + \frac{(q_i p_k + p_i q_k) p_j}{p_i + p_k} \right.\right. \\
& \qquad \left. + \frac{p_i (q_j p_k + p_j q_k)}{p_j + p_k} \right) \\
& + \left(\frac{q_i p_i + q_j p_j}{p_i + p_j} - (q_i \mathbf{1}_{ij}(0) + q_j \mathbf{1}_{ji}(0)) \right) e^{-(p_i + p_j)t} \\
& + \left(\frac{q_i p_i + q_k p_k}{p_i + p_k} - (q_i \mathbf{1}_{ik}(0) + q_k \mathbf{1}_{ki}(0)) \right) e^{-(p_i + p_k)t} \\
& + \left(\frac{q_j p_j + q_k p_k}{p_j + p_k} - (q_j \mathbf{1}_{jk}(0) + q_k \mathbf{1}_{kj}(0)) \right) e^{-(p_j + p_k)t} \\
& + \Big(q_i \mathbf{1}_{ij}(0) \mathbf{1}_{ik}(0) + q_j \mathbf{1}_{ji}(0) \mathbf{1}_{jk}(0) + q_k \mathbf{1}_{ki}(0) \mathbf{1}_{kj}(0) \\
& \qquad \left.\left. - \frac{q_i p_i + q_j p_j + q_k p_k}{p_i + p_j + p_k} \right) e^{-(p_i + p_j + p_k)t} \right]
\end{aligned}
$$

where the deterministic indicators $\mathbf{1}_{ji}(0) = 1$ if and only if j precedes i in the initial permutation.

Proof. The probability function is immediate from Proposition 2.3.1.1. Determining the expectation, we have

$$
E(S^\pi(t)) \quad = \quad \sum_i q_i E(S_i^\pi(t)) = \sum_i q_i \sum_{j : j \neq i} P(\mathbf{1}_{ji}(t) = 1).
$$

By the embedding, (cf. the proof of Proposition 2.3.1.1),

$$
\begin{aligned}
P(\mathbf{1}_{ji}(t) = 1) \quad &= \quad \int_0^t (1 - e^{-p_j u^*}) p_i e^{-p_i u^*} du^* \\
& \qquad + (1 - \mathbf{1}_{ij}(0) e^{-p_j t}) \int_t^\infty p_i e^{-p_i u^*} du^* \\
&= \quad \frac{p_j}{p_i + p_j} (1 - e^{-(p_i + p_j)t}) + \mathbf{1}_{ji}(0) e^{-(p_i + p_j)t}.
\end{aligned}
$$

Hence,

$$\begin{aligned}
E(S^\pi(t)) &= \sum_{i,j:j\neq i} \frac{q_i p_j}{p_i+p_j}(1-e^{-(p_i+p_j)t}) + \sum_{i,j:j\neq i} q_i \mathbf{1}_{ji}(0) e^{-(p_i+p_j)t} \\
&= \sum_{1\leq i<j\leq n} \left[\frac{q_i p_j + p_i q_j}{p_i+p_j}\left(1-e^{-(p_i+p_j)t}\right) \right. \\
&\qquad \left. + \left(q_i \mathbf{1}_{ji}(0) + q_j \mathbf{1}_{ij}(0)\right) e^{-(p_i+p_j)t} \right]
\end{aligned}$$

which equals the expectation in the statement.

The variance is given by

$$\mathrm{Var}\left(S^\pi(t)\right) = E\left(S^\pi(t)^2\right) - \left[E\left(S^\pi(t)\right)\right]^2,$$

where

$$\begin{aligned}
E\left(S^\pi(t)^2\right) &= \sum_i q_i E\left(S_i^\pi(t)^2\right) = \sum_i q_i E\left(\left[\sum_{j:j\neq i} \mathbf{1}_{ji}(t)\right]^2\right) \\
&= \sum_i q_i E\Bigg(\sum_{j:j\neq i} \mathbf{1}_{ji}(t)^2 + \sum_{j:j\neq i}\sum_{k:k\neq i,j} \mathbf{1}_{ji}(t)\mathbf{1}_{ki}(t)\Bigg).
\end{aligned}$$

Since $\mathbf{1}_{ji}(t)^2 = \mathbf{1}_{ji}(t)$ and by the symmetry of the last term we have,

$$E\left(S^\pi(t)^2\right) = E\left(S^\pi(t)\right) + 2\sum_i q_i \sum_{j:j\neq i} \sum_{\substack{k\,:\,k>j \\ k\neq i}} P(\mathbf{1}_{ji}(t)\mathbf{1}_{ki}(t) = 1).$$

The product $\mathbf{1}_{ji}(t)\mathbf{1}_{ki}(t) = 1$ if and only if both indicators are 1. This requires that if Controller has requested record i, then he must have requested both j and k after his last request for i. If, on the other hand, Controller has never requested i, then j precedes i if j has been requested by Controller or has not been requested but was initially located in front of i in the list, cf. the proof of Proposition 2.3.1.1. The same goes for record k. Using the embedding as before, conditioning on the time of Controller's last request for i, we get

$$\begin{aligned}
&P(\mathbf{1}_{ji}(t)\mathbf{1}_{ki}(t) = 1) \\
&= \int_0^t (1-e^{-p_j u^*})(1-e^{-p_k u^*}) p_i e^{-p_i u^*}\, du^* \\
&\quad + [1-\mathbf{1}_{ij}(0)e^{-p_j t}][1-\mathbf{1}_{ik}(0)e^{-p_k t}]e^{-p_i t}
\end{aligned}$$

which after some calculations becomes

$$
\begin{aligned}
&\frac{p_j p_k}{p_i+p_j+p_k}\left(\frac{1}{p_i+p_j}+\frac{1}{p_i+p_k}\right)\\
&+\left(\frac{p_i}{p_i+p_j}-\mathbf{1}_{ij}(0)\right)e^{-(p_i+p_j)t}+\left(\frac{p_i}{p_i+p_k}-\mathbf{1}_{ik}(0)\right)e^{-(p_i+p_k)t}\\
&+\left(\mathbf{1}_{ij}(0)\mathbf{1}_{ik}(0)-\frac{p_i}{p_i+p_j+p_k}\right)e^{-(p_i+p_j+p_k)t}. \qquad (2.3)
\end{aligned}
$$

Hence

$$
\begin{aligned}
\operatorname{Var}(S^\pi(t)) &= E(S^\pi(t))\,(1-E(S^\pi(t)))\\
&+\; 2\sum_{\substack{i,j,k:j\neq i,\,k>j\\ k\neq i}} q_i\;\Bigg[\frac{p_j p_k}{p_i+p_j+p_k}\left(\frac{1}{p_i+p_j}+\frac{1}{p_i+p_k}\right)\\
&\quad+\left(\frac{p_i}{p_i+p_j}-\mathbf{1}_{ij}(0)\right)e^{-(p_i+p_j)t}+\left(\frac{p_i}{p_i+p_k}-\mathbf{1}_{ik}(0)\right)e^{-(p_i+p_k)t}\\
&\quad+\left(\mathbf{1}_{ij}(0)\mathbf{1}_{ik}(0)-\frac{p_i}{p_i+p_j+p_k}\right)e^{-(p_i+p_j+p_k)t}\Bigg].
\end{aligned}
$$

Rewriting this sum, we have (after some changes of summation indices),

$$
\begin{aligned}
\operatorname{Var}(S^\pi(t)) &= E(S^\pi(t))\,(1-E(S^\pi(t)))\\
&+\; 2\sum_{1\le i<j<k\le n}\Bigg[\frac{1}{p_i+p_j+p_k}\Bigg(\frac{(q_ip_j+p_iq_j)p_k}{p_i+p_j}+\frac{(q_ip_k+p_iq_k)p_j}{p_i+p_k}\\
&\qquad\qquad+\frac{p_i(q_jp_k+p_jq_k)}{p_j+p_k}\Bigg)\\
&\quad+\left(\frac{q_ip_i+q_jp_j}{p_i+p_j}-(q_i\mathbf{1}_{ij}(0)+q_j\mathbf{1}_{ji}(0))\right)e^{-(p_i+p_j)t}\\
&\quad+\left(\frac{q_ip_i+q_kp_k}{p_i+p_k}-(q_i\mathbf{1}_{ik}(0)+q_k\mathbf{1}_{ki}(0))\right)e^{-(p_i+p_k)t}\\
&\quad+\left(\frac{q_jp_j+q_kp_k}{p_j+p_k}-(q_j\mathbf{1}_{jk}(0)+q_k\mathbf{1}_{kj}(0))\right)e^{-(p_j+p_k)t}
\end{aligned}
$$

$$+\Big(q_i\mathbf{1}_{ij}(0)\mathbf{1}_{ik}(0)+q_j\mathbf{1}_{ji}(0)\mathbf{1}_{jk}(0)+q_k\mathbf{1}_{ki}(0)\mathbf{1}_{kj}(0) - \frac{q_ip_i+q_jp_j+q_kp_k}{p_i+p_j+p_k}\Big)e^{-(p_i+p_j+p_k)t}\Big]$$

and we are done.

□

The probability function in equation (2.2) is in rather an implicit form. Alternatively, we may express the distribution in terms of a generating function.

Theorem 2.3.1.3 *The generating function for Second User's search cost at time t is*

$$E\left(z^{S^\pi(t)}\right) = \int_0^t e^{-u^*}\left[\sum_{i=1}^n \frac{q_ip_i}{1+(e^{p_iu^*}-1)z}\right]\left[\prod_{j=1}^n\left(1+(e^{p_ju^*}-1)z\right)\right]du^* + e^{-t}\sum_{i=1}^n q_i\prod_{j\neq i}\left[\mathbf{1}_{ij}(0)+\left(e^{p_jt}-\mathbf{1}_{ij}(0)\right)z\right].$$

Proof. Let $E_{u^*}(S_i^\pi(t))$ denote the expectation associated with the probability function $P_{u^*}(S_i^\pi(t)=k)$ in Proposition 2.3.1.1. By conditioning on Second User's current request being for record i and that the time since Controller's last request for this record $L_i=u^*$, we have

$$E\left(z^{S^\pi(t)}\right) = \sum_i q_i\left[\int_0^t p_ie^{-p_iu^*}E_{u^*}\left(z^{S_i^\pi(t)}\right)du^* + e^{-p_it}E_t\left(z^{S_i^\pi(t)}\right)\right].$$

Proposition 2.3.1.1 now gives

$$E\left(z^{S^\pi(t)}\right) = \sum_i q_i\left[\int_0^t p_ie^{-p_iu^*}\prod_{j\neq i}\left[e^{-p_ju^*}+(1-e^{-p_ju^*})z\right]du^* + e^{-p_it}\prod_{j\neq i}\left[\mathbf{1}_{ij}(0)e^{-p_jt}+\left(1-\mathbf{1}_{ij}(0)e^{-p_jt}\right)z\right]\right] \qquad (2.4)$$

$$
\begin{aligned}
&= \int_0^t \sum_i \left[q_i p_i \frac{\prod_j \left[1 + (e^{p_j u^*} - 1)z\right]}{1 + (e^{p_i u^*} - 1)z} \right] \exp\left\{-u^* \sum_j p_j\right\} du^* \\
&\quad + \sum_i q_i \prod_{j \neq i} \left[\mathbf{1}_{ij}(0) + \left(e^{p_j t} - \mathbf{1}_{ij}(0)\right) z\right] \exp\left\{-t \sum_j p_j\right\} \\
&= \int_0^t e^{-u^*} \left[\sum_i \frac{q_i p_i}{1 + (e^{p_i u^*} - 1)z}\right] \left[\prod_j \left(1 + (e^{p_j u^*} - 1)z\right)\right] du^* \\
&\quad + e^{-t} \sum_i q_i \prod_{j \neq i} \left[\mathbf{1}_{ij}(0) + \left(e^{p_j t} - \mathbf{1}_{ij}(0)\right) z\right].
\end{aligned}
$$

□

When Controller and Second User are identical, putting $q_i = p_i$, $i = 1, \ldots, n$, we recapture the expectation and probability distribution for the search cost for a single user.

Corollary 2.3.1.4 (Fill and Holst, 1996) *For a list started in permutation π the probability function for the (single user's) search cost at time t is*

$$P(S^\pi(t) = k) = \sum_{i=1}^n p_i P(S_i^\pi(t) = k),$$

where the distribution of $S_i^\pi(t)$ is as in Proposition 2.3.1.1.

The expectation at this time is

$$
\begin{aligned}
E(S^\pi(t)) \quad = \quad \sum_{1 \le i < j \le n} \Bigg[& \frac{2 p_i p_j}{p_i + p_j} \left(1 - e^{-(p_i + p_j)t}\right) \\
& + \left(p_i \mathbf{1}_{ji}(0) + p_j \mathbf{1}_{ij}(0)\right) e^{-(p_i + p_j)t} \Bigg],
\end{aligned}
$$

where the deterministic indicators $\mathbf{1}_{ji}(0) = 1$ if and only if j precedes i in the initial permutation.

Similarly, we have the single-user variance.

Corollary 2.3.1.5 *Under the conditions in the preceding corollary,*

$$
\begin{aligned}
&\operatorname{Var}\left(S^{\pi}(t)\right) = E\left(S^{\pi}(t)\right)\left(1 - E\left(S^{\pi}(t)\right)\right) \\
&+ \quad 2 \sum_{1 \le i < j < k \le n} \left[\frac{2 p_i p_j p_k}{p_i + p_j + p_k} \left(\frac{1}{p_i + p_j} + \frac{1}{p_i + p_k} + \frac{1}{p_j + p_k} \right) \right. \\
&\quad + \left(\frac{p_i^2 + p_j^2}{p_i + p_j} - \left(p_i \mathbf{1}_{ij}(0) + p_j \mathbf{1}_{ji}(0)\right) \right) e^{-(p_i + p_j)t} \\
&\quad + \left(\frac{p_i^2 + p_k^2}{p_i + p_k} - \left(p_i \mathbf{1}_{ik}(0) + p_k \mathbf{1}_{ki}(0)\right) \right) e^{-(p_i + p_k)t} \\
&\quad + \left(\frac{p_j^2 + p_k^2}{p_j + p_k} - \left(p_j \mathbf{1}_{jk}(0) + p_k \mathbf{1}_{kj}(0)\right) \right) e^{-(p_j + p_k)t} \\
&\quad + \Big(p_i \mathbf{1}_{ij}(0) \mathbf{1}_{ik}(0) + p_j \mathbf{1}_{ji}(0) \mathbf{1}_{jk}(0) + p_k \mathbf{1}_{ki}(0) \mathbf{1}_{kj}(0) \\
&\qquad \left. - \frac{p_i^2 + p_j^2 + p_k^2}{p_i + p_j + p_k} \Big) e^{-(p_i + p_j + p_k)t} \right].
\end{aligned}
$$

Finally, the single-user generating function at time t follows from Theorem 2.3.1.3.

Corollary 2.3.1.6 *The generating function for a single user's search cost at time t is*

$$
\begin{aligned}
E\left(z^{S^{\pi}(t)}\right) &= \int_0^t e^{-u^*} \left[\sum_{i=1}^{n} \frac{p_i^2}{1 + (e^{p_i u^*} - 1) z} \right] \left[\prod_{j=1}^{n} \left(1 + (e^{p_j u^*} - 1) z\right) \right] du^* \\
&\quad + e^{-t} \sum_i p_i \prod_{j \ne i} \left[\mathbf{1}_{ij}(0) + \left(e^{p_j t} - \mathbf{1}_{ij}(0)\right) z \right].
\end{aligned}
$$

2.3.2 Stationary search cost

In stationarity, enough time has elapsed to ensure that the move-to-front rule is solely responsible for the organization of the records, independently of the initial arrangement (cf. Section 1.11.1.2). Passing to the limit in Theorem 2.3.1.2 considerably simplifies the expressions for the mean and variance. As we let time t in Proposition 2.3.1.1 tend to infinity, the probability function for the (stationary) search cost, given by equation (2.2), is simplified, yet still rather implicit. We shall confine ourselves to the generating function, given in Theorem 2.3.2.2.

Let S^{stat} be Second User's stationary search cost.

Corollary 2.3.2.1 *The expectation and variance of Second User's stationary search cost are*

$$
\begin{aligned}
E\left(S^{stat}\right) &= \sum_{1 \le i < j \le n} \frac{q_i p_j + p_i q_j}{p_i + p_j} \\
\operatorname{Var}\left(S^{stat}\right) &= E\left(S^{stat}\right)\left(1 - E\left(S^{stat}\right)\right) \\
&\quad + 2 \sum_{1 \le i < j < k \le n} \frac{1}{p_i + p_j + p_k} \left(\frac{(q_i p_j + p_i q_j) p_k}{p_i + p_j} + \frac{(q_i p_k + p_i q_k) p_j}{p_i + p_k} \right. \\
&\qquad \left. + \frac{p_i (q_j p_k + p_j q_k)}{p_j + p_k} \right).
\end{aligned}
$$

Remark. In stationarity the probability that both j and k precede i in the list is given by

$$
\frac{p_j p_k}{p_i + p_j + p_k} \left(\frac{1}{p_i + p_j} + \frac{1}{p_i + p_k} \right). \tag{2.5}
$$

This follows from the embedding; when $t \to \infty$, $P(\mathbf{1}_{ji}(t)\mathbf{1}_{ki}(t) = 1)$ tends to the expression (2.5) as we let $t \to \infty$ in (2.3).

But there is also another way to view this probability: In order for j and k to precede i in the list they must have been requested after the last request for i. This is the case if we first request k from i, j, k and then request j from i, j — now j precedes k which precedes i in the list — or if we first request j from i, j, k and then k from i, k. Hence the required probability is

$$
\frac{p_k}{p_i + p_j + p_k} \frac{p_j}{p_i + p_j} + \frac{p_j}{p_i + p_j + p_k} \frac{p_k}{p_i + p_k}
$$

which equals the expression in (2.5) above.

Corollary 2.3.2.2 *The generating function for Second User's search cost in stationarity is*

$$
E\left(z^{S^{stat}}\right) = \int_0^\infty e^{-u^*} \left[\sum_{i=1}^{n} \frac{q_i p_i}{1 + (e^{p_i u^*} - 1) z} \right] \left[\prod_{j=1}^{n} \left(1 + (e^{p_j u^*} - 1) z\right) \right] du^*.
$$

Proof. As time t tends to infinity in Theorem 2.3.1.3, the last sum, by (2.4),

$$e^{-t}\sum_i q_i \prod_{j\neq i}\left[\mathbf{1}_{ij}(0)+\left(e^{p_j t}-\mathbf{1}_{ij}(0)\right)z\right] \le \sum_i e^{-p_i t}\prod_{j\neq i}\left[e^{-p_j t}+z\right]$$
$$\le \sum_i e^{-p_i t}\prod_{j\neq i}[1+z] \longrightarrow 0.$$

□

Again, the single-user analogues follow when $q_i = p_i$, $i = 1,\ldots,n$.

Corollary 2.3.2.3 (McCabe, 1965) *The expected stationary search cost and the variance for single-user move-to-front are*

$$\begin{aligned} E\left(S^{stat}\right) &= 2\sum_{1\le i<j\le n}\frac{p_i p_j}{p_i+p_j} \\ \mathrm{Var}\left(S^{stat}\right) &= E\left(S^{stat}\right)\left(1-E\left(S^{stat}\right)\right) \\ &\quad + 4\sum_{1\le i<j<k\le n}\frac{p_i p_k p_j}{p_i+p_j+p_k}\left(\frac{1}{p_i+p_j}+\frac{1}{p_i+p_k}+\frac{1}{p_j+p_k}\right). \end{aligned}$$

Corollary 2.3.2.4 (Fill and Holst, 1996) *The generating function for a single user's search cost in stationarity is*

$$E\left(z^{S^{stat}}\right) = \int_0^\infty e^{-u^*}\left[\sum_{i=1}^n \frac{p_i^2}{1+(e^{p_i u^*}-1)z}\right]\left[\prod_{j=1}^n\left(1+(e^{p_j u^*}-1)z\right)\right]du^*.$$

We conclude this chapter with a remark. If instead of considering the number of records in front of the requested one, we had used the original definition taking the number of required comparisons to be the search cost, we would have arrived at slightly different results. For instance, we would get the following expectation and variance.

Corollary 2.3.2.5 *Let C^{stat} be the number of comparisons required in stationarity to find a record requested by Second User. Then*

$$\begin{aligned}
E\left(C^{stat}\right) &= 1 + \sum_{1\le i<j\le n} \frac{q_i p_j + p_i q_j}{p_i + p_j} \\
\mathrm{Var}\left(S^{stat}\right) &= \left(E\left(C^{stat}\right) - 1\right)\left(2 - E\left(C^{stat}\right)\right) \\
&\quad + 2 \sum_{1\le i<j<k\le n} \frac{1}{p_i + p_j + p_k}\left(\frac{(q_i p_j + p_i q_j)p_k}{p_i + p_j} + \frac{(q_i p_k + p_i q_k)p_j}{p_i + p_k}\right. \\
&\quad\quad \left. + \frac{p_i(q_j p_k + p_j q_k)}{p_j + p_k}\right).
\end{aligned}$$

Proof. The relation $C^{stat} = 1 + S^{stat}$ combined with previous results on $E\left(S^{stat}\right)$ and $E\left(\left(S^{stat}\right)^2\right)$ gives the desired expressions.

□

In Part II we shall study the so-called move-to-root rule, where instead of keeping the records in a list we use a binary search tree. Comparing the expectation and variance of Corollary 2.3.2.5 with the analogous formulas for the move-to-root rule in Theorem 6.1.2.1, one may note that they show a remarkable resemblance.

Chapter 3

Convergence

Comparing the results in the previous chapter, the formulas for the stationary search cost are clearly much more appealing than those describing the transient one. Consequently, when possible, we wish to describe move-to-front by its stationary behaviour. For how long must move-to-front be operating until the search cost reaches stationarity? In order to answer this question we need a way to measure the discrepancy between the search cost at some time t and that in stationarity. For a move-to-front list with a fixed number of records and certain request probabilities, how long time must pass until this discrepancy is less than some specified number? Of course, when the entire list permutation process has reached stationarity (and it will since it is an irreducible aperiodic Markov chain with finite state space, governed only by Controller's independent requests with constant probabilities), then so has the search cost. But, as the examples in Fill [22] show, the convergence of the search cost may be much faster than that of the permutation process.

In this chapter, we generalize the results of Fill [22] and Fill and Holst [23] to the case of two users. We give upper and lower bounds for the discrepancy (measured by the total variation distance defined in (3.1)) between the search cost at time t and in stationarity. On the way, we show the intuitively appealing result that the more the initial arrangement differs from the one agreeing with Second User's taste, the greater the search cost at any time. Finally, in Section 3.5, we look at some examples.

3.1 Total variation distance

We use essentially the same notation as in Chapter 2, only we let $S^{stat}(t)$ denote the stationary search cost to stress that the sample paths do indeed depend on the time t although the distribution of the stationary cost does not. In the previous chapter this quantity was labeled S^{stat}.

The *total variation distance* between the distribution of Second User's search cost at time t for a list started in permutation π and that of one in stationarity is defined as

$$\sup_A |P(S^\pi(t) \in A) - P(S^{stat}(t) \in A)|.$$

Here, the supremum is taken over all subsets A of $\{0, \dots, n-1\}$.

If Second User makes a request at time t, what is the greatest search cost discrepancy, due to the different initial list arrangements, for all possible future Second User requests? This quantity, given by

$$TV^\pi(t) := \sup_{t' \geq t} \sup_A |P(S^\pi(t') \in A) - P(S^{stat}(t') \in A)|, \tag{3.0}$$

is surely non-increasing in t. We shall consider the 'worst case' of this distance with respect to the initial arrangement,

$$TV(t) := \sup_\pi TV^\pi(t), \tag{3.1}$$

and give upper and lower bounds for it in Sections 3.3 - 3.4. In Section 3.5 we give rates of convergence in this sense for some distributions of Controller's and Second User's request probabilities. In order to do so, we use the lower and upper bounds given in Corollary 3.3.1 and Theorem 3.4.1. However, before we can get the lower bound, we need the inequality in Theorem 3.2.3 in the next section.

3.2 Stochastic ordering

For a single user, Fill and Holst [23, Corollary 4.1 - 4.2] showed that, roughly put, 'the farther the list begins away from the optimal arrangement, the greater the search cost at any time'. Here, we show that this intuitively clear result holds also for two users.

Label the records in decreasing order of Second User's request probabilities, $q_1 \geq q_2 \geq \cdots \geq q_n$, and recall that by $S^\pi(t)$ we mean the search cost due to a request by Second User at time t when the list is initially in permutation π. In the following we use the notation $id := (1, 2, \ldots, n)$ and $rev := (n, n-1, \ldots, 1)$.

Theorem 3.2.1 *For each $t \in [0, \infty)$, $S^\pi(t)$ is stochastically minimized by $\pi = id$ and stochastically maximized by $\pi = rev$.*

Proof. We shall use a coupling argument. Consider two lists, one started in permutation π and the other in permutation π', where

$$\pi := (1, 2, \ldots, k, k+1, \ldots, n)$$

and

$$\pi' := (1, 2, \ldots, k-1, k+1, k, k+2, \ldots, n),$$

that is, π' is obtained from π by moving $k+1$ to the left of k.

Let the sequence of Controller's requests be the same for the two lists. Then, if T denotes the first time Controller requests either of the records k or $k+1$, the arrangements in the two lists will be identical when $t \geq T$ (because at time T the requested record is moved to the front). Furthermore, let $P_{t<T}(A) := P(A \cap \{t < T\})$. Then, using that for $t < T$,

$$\begin{aligned} S_k^\pi(t) &= S_{k+1}^{\pi'}(t) \\ S_{k+1}^\pi(t) &= S_k^{\pi'}(t) \\ S_i^\pi(t) &= S_i^{\pi'}(t) \qquad \text{for all } i \neq k, k+1, \end{aligned}$$

we have for any $c \in \{0, \ldots, n-1\}$,

$$\begin{aligned} &P_{t<T}(S^{\pi'}(t) \geq c) - P_{t<T}(S^\pi(t) \geq c) \\ &\quad = \sum_{i=1}^n q_i \left[P_{t<T}(S_i^{\pi'}(t) \geq c) - P_{t<T}(S_i^\pi(t) \geq c)\right] \\ &\quad = q_k \left[P_{t<T}(S_k^{\pi'}(t) \geq c) - P_{t<T}(S_k^\pi(t) \geq c)\right] \\ &\qquad\quad + q_{k+1} \left[P_{t<T}(S_{k+1}^{\pi'}(t) \geq c) - P_{t<T}(S_{k+1}^\pi(t) \geq c)\right] \\ &\quad = q_k \left[P_{t<T}(S_k^{\pi'}(t) \geq c) - P_{t<T}(S_k^\pi(t) \geq c)\right] \\ &\qquad\quad + q_{k+1} \left[P_{t<T}(S_k^\pi(t) \geq c) - P_{t<T}(S_k^{\pi'}(t) \geq c)\right] \\ &\quad = (q_k - q_{k+1}) \left[P_{t<T}(S_k^{\pi'}(t) \geq c) - P_{t<T}(S_k^\pi(t) \geq c)\right] \geq 0. \end{aligned}$$

The inequality holds since $q_k \geq q_{k+1}$ by our choice of indices. Hence, due to the coupling,

$$P(S^{\pi'}(t) \geq c) \geq P(S^{\pi}(t) \geq c)$$

for all t.

The same line of reasoning holds for any permutations π and π' where π' has been obtained from π by a transposition of two elements such that if $i < j$, $\pi(i) < \pi(j)$. Since every permutation can be constructed from id by a finite number of such transpositions[1] it follows that $S^{\pi}(t)$ is stochastically minimal for $\pi = id$.

Similar arguments give that $S^{\pi}(t)$ is stochastically maximal for $\pi = rev$.

□

Following Fill and Holst [23], we define

$$d_{\pi}(t) := \sup_c |P(S^{\pi}(t) \leq c) - P(S^{stat}(t) \leq c)|,$$

where c ranges from 0 to $n-1$. Then, $d_{\pi}(t)$ is the *Kolmogorov-Smirnov distance* between the distributions of Second User's search cost at time t in a list started in permutation π and one in stationarity. Similarly, let

$$\tilde{d}(t) := \sup_c \Big(P(S^{id}(t) \leq c) - P(S^{rev}(t) \leq c)\Big),$$

be the Kolmogorov-Smirnov distance between the search cost distributions for lists started in permutations id resp. rev. Note that Theorem 3.2.1 guarantees that $\tilde{d}(t)$ is non-negative. Now, consider

$$\hat{d}(t) := \sup_{\pi} d_{\pi}(t).$$

By Theorem 3.2.1 we have that, for any c,

$$\begin{aligned}
&\sup_{\pi} |P(S^{\pi}(t) \leq c) - P(S^{stat}(t) \leq c)| \\
&\quad = \sup[P(S^{stat}(t) \leq c) - P(S^{rev}(t) \leq c), \\
&\qquad\qquad P(S^{id}(t) \leq c) - P(S^{stat}(t) \leq c)].
\end{aligned}$$

and hence,

$$\hat{d}(t) = \sup[d_{rev}(t), d_{id}(t)],$$

[1]For instance, we can move an item to the front by transposing it successively with each of the items preceding it in the list. Moving items to the front in reversed order creates any permutation desired.

where

$$\begin{aligned} d_{rev}(t) &:= \sup_c [P(S^{stat}(t) \le c) - P(S^{rev}(t) \le c)] \ge 0 \\ d_{id}(t) &:= \sup_c [P(S^{id}(t) \le c) - P(S^{stat}(t) \le c)] \ge 0. \end{aligned}$$

So

$$\tilde{d}(t) \ge \hat{d}(t) \ge \frac{1}{2}\tilde{d}(t).$$

Lemma 3.2.2 (Fill and Holst, 1996) *As t increases, $S^{rev}(t)$ is stochastically non-increasing and $S^{id}(t)$ is stochastically non-decreasing.*

Proof. See Fill and Holst [23].

□

Since the distribution of $S^{stat}(t)$ does not depend on t, Lemma 3.2.2 tells us that both $\hat{d}(t)$ and $\tilde{d}(t)$ are non-increasing in t.

Now, we find lower bounds for $d_{rev}(t)$, $d_{id}(t)$ and $\tilde{d}(t)$ by averaging[2]:

Let

$$\Delta_{rev}(t) := E(S^{rev}(t)) - E(S^{stat}(t)).$$

Then

$$\begin{aligned} \Delta_{rev}(t) &= \sum_{c=0}^{n-1} P(S^{rev}(t) \ge c) - \sum_{c=0}^{n-1} P(S^{stat}(t) \ge c) \\ &= 1 + \sum_{c=0}^{n-1}(1 - P(S^{rev}(t) \le c)) - \left(1 + \sum_{c=0}^{n-1}(1 - P(S^{stat}(t) \le c))\right) \\ &= \sum_{c=0}^{n-1}(P(S^{stat}(t) \le c) - P(S^{rev}(t) \le c)) \\ &\le n d_{rev}(t). \end{aligned}$$

Similarly, if

$$\Delta_{id}(t) := E(S^{stat}(t)) - E(S^{id}(t))$$

then

$$\Delta_{id}(t) \le n d_{id}(t).$$

[2] This is the best we can do since we do not know how to find the maximizing values of c and therefore cannot compute $d_{rev}(t)$, $d_{id}(t)$ and $\tilde{d}(t)$.

Hence,

$$\begin{aligned} d_{rev}(t) &\geq \frac{\Delta_{rev}(t)}{n} \\ d_{id}(t) &\geq \frac{\Delta_{id}(t)}{n}. \end{aligned}$$

Now, $E(S^{stat}(t))$ is known from Corollary 2.3.2.1 while $E(S^{rev}(t))$ and $E(S^{id}(t))$ follow from Theorem 2.3.1.2. When the initial permutation is rev, the (deterministic) indicator $\mathbf{1}_{ji}(0) = 1$ if and only if $i < j$. Analogously, when the list is started in permutation id, $\mathbf{1}_{ij}(0) = 1$ if and only if $i < j$. Consequently,

$$\begin{aligned} \Delta_{rev}(t) &= \sum_{1 \leq i < j \leq n} \left[\frac{q_i p_j + p_i q_j}{p_i + p_j} \left(1 - e^{-(p_i+p_j)t}\right) + q_i e^{-(p_i+p_j)t} \right] \\ &\quad - \sum_{1 \leq i < j \leq n} \frac{q_i p_j + p_i q_j}{p_i + p_j} \\ &= \sum_{1 \leq i < j \leq n} (q_i - q_j) \frac{p_i}{p_i + p_j} e^{-(p_i+p_j)t} \geq 0 \end{aligned}$$

and

$$\begin{aligned} \Delta_{id}(t) &= \sum_{1 \leq i < j \leq n} \frac{q_i p_j + p_i q_j}{p_i + p_j} \\ &\quad - \sum_{1 \leq i < j \leq n} \left[\frac{q_i p_j + p_i q_j}{p_i + p_j} \left(1 - e^{-(p_i+p_j)t}\right) + q_j e^{-(p_i+p_j)t} \right] \\ &= \sum_{1 \leq i < j \leq n} (q_i - q_j) \frac{p_j}{p_i + p_j} e^{-(p_i+p_j)t} \geq 0. \end{aligned}$$

Note that $\Delta_{rev}(t)$ and $\Delta_{id}(t)$ are both decreasing in t since $e^{-\alpha t}$ is decreasing in t.

Finally, we collect our results in the main theorem of this section:

Theorem 3.2.3 *With the notation defined above,*

$$\begin{aligned} \tilde{d}(t) &\geq \hat{d}(t) := \sup_{\pi} d_\pi(t) = \sup(d_{rev}(t), d_{id}(t)) \\ &\geq \sup \left[\frac{1}{2}\tilde{d}(t), \frac{\Delta_{rev}(t)}{n}, \frac{\Delta_{id}(t)}{n} \right]. \end{aligned}$$

Furthermore, $\tilde{d}(t), \hat{d}(t), \Delta_{rev}(t)$ *and* $\Delta_{id}(t)$ *are all non-increasing in* t.

3.3 General lower bound

To obtain a lower bound for the total variation distance (3.1), instead of taking the supremum over all subsets of $\{0, \ldots, n-1\}$ we restrict the supremum to intervals of the form $A = \{0, \ldots, c\}$. Recognizing this as the Kolmogorov-Smirnov distance, the bound follows directly from Theorem 3.2.3.

Corollary 3.3.1 *Assume $q_1 \geq \cdots \geq q_n$ and $p_i > 0$ for $i = 1, \ldots, n$. Then the total variation distance defined in (3.1) satisfies*

$$TV(t) \geq \sup_{t' \geq t} \tilde{d}(t') = \tilde{d}(t), \tag{3.2}$$

where $\tilde{d}(t)$ is the Kolmogorov-Smirnov distance between the distributions of $S^{id}(t)$ and $S^{rev}(t)$ such that

$$\tilde{d}(t) \geq \sup \left[\frac{1}{2} \tilde{d}(t), \frac{\Delta_{rev}(t)}{n}, \frac{\Delta_{id}(t)}{n} \right]$$

and

$$\begin{aligned} 0 \leq \Delta_{rev}(t) &= \sum_{1 \leq i < j \leq n} (q_i - q_j) \frac{p_i}{p_i + p_j} e^{-(p_i + p_j)t} \\ 0 \leq \Delta_{id}(t) &= \sum_{1 \leq i < j \leq n} (q_i - q_j) \frac{p_j}{p_i + p_j} e^{-(p_i + p_j)t}. \end{aligned}$$

Proof. Since by Theorem 3.2.3, $\tilde{d}(t), \Delta_{rev}(t)$ and $\Delta_{id}(t)$ are all non-increasing in t, (3.2) follows.

□

3.4 General upper bound

The following theorem gives an upper bound for the total variation distance $TV(t)$ defined in (3.1).

Theorem 3.4.1

$$TV(t) \leq \sum_{i=1}^{n} q_i e^{-p_i t}.$$

Proof. Again we use a coupling argument similar to that in the proof of Lemma 3.2.1. Consider two lists, one started in permutation π and the other in stationarity. Let the sequence of Controller's (as well as Second User's) requests be the same for both lists. If T_i^C denotes the time of Controller's first request for record i, then record i will be in the same position in both lists at time $\geq T_i^C$. Hence, for Second User request times $t' \geq t$,

$$\begin{aligned}
&|P(S^\pi(t') \in A) - P(S^{stat}(t') \in A)| \\
&\qquad = |P(S^\pi(t') \in A, S^\pi(t') = S^{stat}(t')) \\
&\qquad\quad + P(S^\pi(t') \in A, S^\pi(t') \neq S^{stat}(t')) \\
&\qquad\quad - P(S^{stat}(t') \in A, S^\pi(t') = S^{stat}(t')) \\
&\qquad\quad - P(S^{stat}(t') \in A, S^\pi(t') \neq S^{stat}(t'))| \\
&\qquad = |P(S^\pi(t') \in A, S^\pi(t') \neq S^{stat}(t')) \\
&\qquad\quad - P(S^{stat}(t') \in A, S^\pi(t') \neq S^{stat}(t'))| \\
&\qquad \leq P(S^\pi(t') \neq S^{stat}(t')) = \sum_{i=1}^n q_i P(T_i^C > t') \\
&\qquad = \sum_{i=1}^n q_i e^{-p_i t'} \leq \sum_{i=1}^n q_i e^{-p_i t}.
\end{aligned}$$

Since this upper bound is independent of both π and A as well as of t', it is an upper bound for $TV(t)$.

□

3.5 Rates of convergence — examples

In the introduction to this chapter we asked how long we need to wait until we have a small discrepancy between the search cost due to the initial arrangement and the stationary cost for a move-to-front list with a fixed number n of items and certain request probabilities. In this section, we use Corollary 3.3.1 and Theorem 3.4.1 to answer that question.

For a single user, Fill [22, Section 3] has given rates of convergence for $TV(t)$, defined in (3.1), using single-user analogues of the bounds in Corollary 3.3.1 and Theorem 3.4.1. The request probability distributions he considered are the uniform distribution, Zipf's law, generalized Zipf's law, power law and the geometric distribution. These distributions are defined as follows:

Let $r_i, i = 1, \ldots, n$ be the probability that record i is requested.

Uniform:	$r_i \propto 1$
Generalized Zipf's law:	$r_i \propto i^{-\alpha}$, $\alpha > 0$
and in particular:	
Zipf's law:	$r_i \propto i^{-1}$
Power law:	$r_i \propto i^{\beta}, \beta > 0$
Geometric:	$r_i \propto \theta^i, 0 < \theta < 1$

What can be said in the case of two users? In this section we consider the rates of convergence for some choices of Controller and Second User request probabilities from the table above. To define what we mean by necessary and sufficient time for convergence to stationarity of Second User's search cost, we extend the definition used by Fill [22].

Intuition tells us that the more items there are in the list, the slower the convergence. For this reason, we decide to view time as a function of the number of records; $t = t(n, c)$. Now, let us define formally sufficient and necessary time for convergence.

Given triangular arrays of request probabilities $\{p_i\}_{i=1}^n, \{q_i\}_{i=1}^n$, $n \geq 1$, we say that time $t = t(n, c)$ is *sufficient* for convergence to stationarity if there exists a decreasing function $H(\cdot)$, independent of n, such that $TV(t) \leq H(c)$ for all $n \geq N(c)$ and $c > 0$. Furthermore we require that $H(c) \to 0$ as $c \to \infty$ and that for each c, $N(c) < \infty$.

If there is such a function $H(\cdot)$, this means that for any fixed c, regardless of how many records n there are (as long as they are more than $N(c)$), the discrepancy measured by $TV\ (t(n, c))$ is at most $H(c)$, see Figure 3.5.1. In the examples below, the value of $N(c)$ defining what is 'sufficiently large' appears merely as restrictions on the inequalities we use.

Conversely, we say that $t = t(n, c)$ is *necessary* for convergence to stationarity if there exists a function $h(\cdot)$, independent of n, such that $h(c) \leq TV(t)$. This means that if there are n items in the list, then up to time $t = t(n, c)$, the discrepancy $TV(t)$ is at least $h(c)$ and this bound is the same for all n, see Figure 3.5.2.

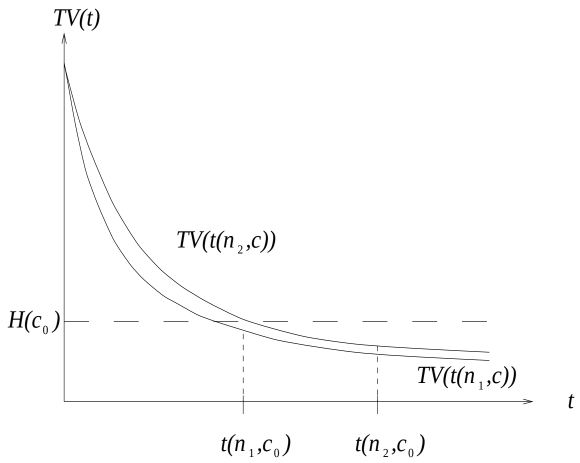


Figure 3.5.1: *Sufficient time for convergence.* Consider two lists with n_1 and n_2 records respectively (where n_1, n_2 are sufficiently large and $n_1 < n_2$). From time $t = t(n, c)$ onward, the total variation distance keeps below $H(c)$. In particular for our two lists, from time $t(n_1, c_0)$ resp. $t(n_2, c_0)$ the total variation distance is at most $H(c_0)$.

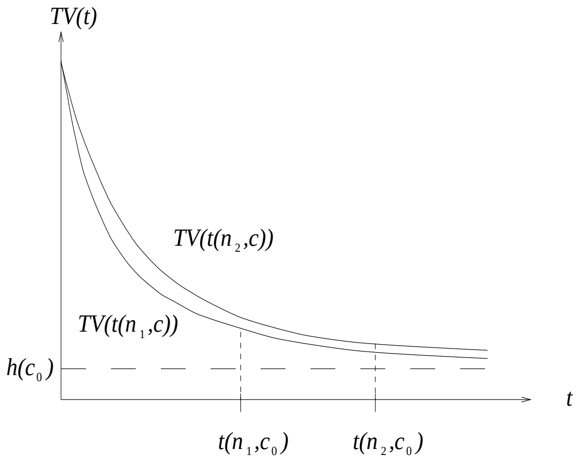


Figure 3.5.2: *Necessary time for convergence.* Consider two lists with n_1 resp. n_2 records ($n_1 < n_2$). At time $t = t(n, c)$, the total variation distance is at least $h(c)$, regardless of n. In particular for our two lists, up to time $t(n_1, c_0)$ resp. $t(n_2, c_0)$ the total variation distance is at least $h(c_0)$.

How do these definitions help us to find out how long we must wait until the discrepancy $TV(t)$ is less than a certain value? Imagine that the request probabilities $p_1, \ldots, p_n, q_1, \ldots, q_n$ are known for some (also known) number of records n. Set $H(c)$ equal to the desired $TV(t)$-value and find out what c is in this case. With n, this gives the sufficient time $t(n, c)$, which means that under these circumstances, if we wait for time $t(n, c)$, the discrepancy will certainly have fallen below the given value.

If instead we have n_1 records with the same 'structure' of the request probabilities (for instance distributed according to the power law, see the exampes below), we must instead wait for time $t(n_1, c)$ in order to guarantee that the total variation distance is less than the one specified. That is, we can see clearly how the number of records effect the rate of convergence.

Is the waiting time we get in this way too long? Set $h(c)$ equal to the same specified value of the discrepancy $TV(t)$, read off c and use it together with the known n to get the necessary time $t(n, c)$ telling us for how long the total variation distance is definitely greater than the desired discrepancy. We may now compare the sufficient waiting-time in the previous paragraph with this necessary time. If they are equal, then this is the 'true' waiting time until the total variation distance assumes this value. Otherwise the 'true' time is somewhere between the two.

Recall that throughout this chapter we choose the indices so that Second User's request probabilities satisfy $q_1 \geq q_2 \geq \cdots \geq q_n$.

3.5.1 Integral comparison

In what follows we shall make frequent use of some integral comparisons. To avoid excessive repetitiousness we collect them here.

Define

$$\begin{aligned} G_n(\beta) &:= \sum_{i=1}^{n} i^{\beta} \\ H_n(\alpha) &:= \sum_{i=1}^{n} i^{-\alpha} \\ H_n &:= \sum_{i=1}^{n} i^{-1}. \end{aligned}$$

Note that H_n is merely $H_n(1)$.

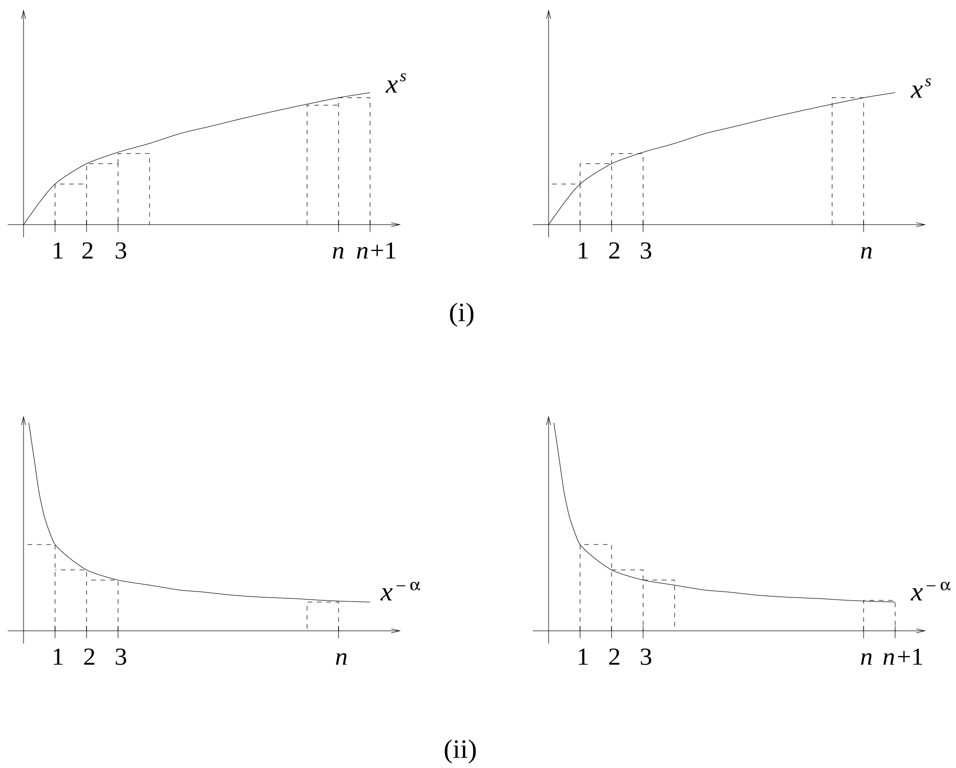


Figure 3.5.3: *Integral comparison.* Construction of upper and lower bound for (i) $G_n(\beta)$ and (ii) $H_n(\alpha)$.

Lemma 3.5.1 *Unless otherwise stated the following inequalities hold for* $n \geq 1$*:*

$$\frac{n^{\beta+1}}{\beta+1} \leq G_n(\beta) \leq \frac{\beta+2}{\beta+1} n^{\beta+1} \quad \text{for } \beta > 0, \tag{3.3}$$

$$\frac{1-2^{-(1-\alpha)}}{1-\alpha}\, n^{1-\alpha} \leq H_n(\alpha) \leq \frac{n^{1-\alpha}}{1-\alpha} \quad \text{for } 0 < \alpha < 1 \text{ and } n \geq 2 \text{ in the first inequality,} \tag{3.4}$$

$$\ln n \leq H_n \leq (\ln n)^2 \quad \text{for } n \geq 6 \text{ in the last inequality,} \tag{3.5}$$

$$\frac{1-2^{-(\alpha-1)}}{\alpha-1} \leq H_n(\alpha) \leq \frac{\alpha}{\alpha-1} \quad \text{for } \alpha > 1. \tag{3.6}$$

Proof. All these inequalities have been obtained through the same line of reasoning, see Figure 3.5.3.

$$\begin{aligned} G_n(\beta) &\leq \int_0^n x^\beta dx + n^\beta = \frac{n^{\beta+1}}{\beta+1} + n^\beta \\ &\leq \frac{n^{\beta+1}}{\beta+1}\left[1 + (\beta+1)\frac{1}{n}\right] \leq \frac{\beta+2}{\beta+1} n^{\beta+1}. \end{aligned} \tag{3.7}$$

Similarly,

$$G_n(\beta) \geq \int_0^n x^\beta dx = \frac{n^{\beta+1}}{\beta+1}. \tag{3.8}$$

For $H_n(\alpha)$, $0 < \alpha < 1$, we have

$$H_n(\alpha) \leq \int_0^n x^{-\alpha} dx = \frac{n^{1-\alpha}}{1-\alpha}, \tag{3.9}$$

and, if $n \geq 2$ (for the last inequality to hold),

$$H_n(\alpha) \geq \int_1^n x^{-\alpha} dx = \frac{n^{1-\alpha}-1}{1-\alpha} \geq \frac{1-2^{-(1-\alpha)}}{1-\alpha}\, n^{1-\alpha}. \tag{3.10}$$

If $\alpha > 1$,

$$H_n(\alpha) \leq \int_1^n x^{-\alpha} dx + 1 \leq \frac{\alpha}{\alpha-1},$$

$$H_n(\alpha) \geq \int_1^{n+1} x^{-\alpha} dx \geq \frac{1-2^{-(\alpha-1)}}{\alpha-1}.$$

Finally, for $n \geq 6$ (again for the sake of the last inequality),

$$H_n \leq \int_1^n x^{-1} dx + 1 = \ln n + 1 \leq (\ln n)^2,$$

and

$$H_n \geq \int_1^{n+1} x^{-1} dx = \ln(n+1) \geq \ln n.$$

$\square$

3.5.2 Uniformly distributed Second User-requests

For any permutation π of the list, the search cost is less than or equal to c if and only if Second User requests any of the records $\pi(1), \ldots, \pi(c+1)$. So, when $q_i = 1/n, i = 1, \ldots, n$, the probability that the search cost is at most c will simply be $(c+1)/n$. Hence, in this case, for all permutations π, $P(S^\pi(t) \leq c) - P(S^{stat}(t) \leq c) = 0$ for all $t \geq 0$ and all $c \in \{0, \ldots, n-1\}$. In fact,

$$|P(S^\pi(t) \in A) - P(S^{stat}(t) \in A)| = 0$$

for all subsets A of $\{0, \ldots, n-1\}$ and it follows that for all $t \geq 0$, the total variation distance defined in (3.1) is 0.

3.5.3 Summable weights

If the request probabilities can be written in the form

$$r_i = \frac{w_i}{\sum_{i=1}^n w_i},$$

where the weights w_i do not depend on n and $\sum_{i=1}^\infty w_i < \infty$, we say that the distribution have *summable weights*. For our examples, the geometric distribution and the generalized Zipf's law with $\alpha > 1$ satisfy this criterion.

As soon as Second User's requests have summable weights, then, regardless of Controller's request probabilities, time c is necessary for convergence to stationarity (Proposition 3.5.3.2). Time c is sufficient if the weights are summable for both Controller's and Second User's requests (Proposition 3.5.3.1). In Section 3.5.5 we find sufficient time for some examples where only one of Controller and Second User requests with summable weights.

Proposition 3.5.3.1 *Suppose that Controller's and Second User's request probabilities for $i = 1, \dots, n$ are*

$$p_i = \frac{\nu_i}{\sum_{j=1}^n \nu_j}$$

resp.

$$q_i = \frac{w_i}{\sum_{j=1}^n w_j}.$$

If $\sum_{i=1}^\infty \nu_i < \infty$ and $\sum_{i=1}^\infty w_i < \infty$ then for $t \geq t(n,c) = c$ (no restrictions on n or c except $n \geq 1,\ c \geq 0$),

$$TV(t) \leq \frac{1}{w_1} \sum_{i=1}^\infty w_i e^{-c\nu_i} \downarrow 0 \text{ as } c \uparrow \infty$$

and hence time c is sufficient for convergence to stationarity.

Proof. Without loss of generality, we may assume $\sum_{i=1}^\infty \nu_i = \sum_{i=1}^\infty w_i = 1$. By Theorem 3.4.1 we have for $t \geq t(n,c) = c$,

$$\begin{aligned} TV(t) &\leq \sum_{i=1}^n q_i e^{-p_i t} = \sum_{i=1}^n \frac{w_i}{\sum_{j=1}^n w_j} \exp\left\{ -\frac{\nu_i t}{\sum_{j=1}^n \nu_j} \right\} \\ &\leq \sum_{i=1}^\infty \frac{w_i}{\sum_{j=1}^n w_j} \exp\left\{ -\frac{\nu_i c}{\sum_{j=1}^n \nu_j} \right\} \leq \frac{1}{w_1} \sum_{i=1}^\infty w_i e^{-\nu_i c} =: H(c). \end{aligned}$$

Since

$$w_i e^{-\nu_i c} \to 0 \qquad \text{for all } i \text{ as } c \to \infty,$$

and since

$$w_i e^{-\nu_i c} \leq w_i \quad \text{where} \quad \sum_1^\infty w_i = 1 < \infty,$$

by the dominated convergence theorem we have that, $H(c) \downarrow 0$ as $c \uparrow \infty$. Hence, c is sufficient time for convergence.

$\square$

The first few records may be equally popular with Second User. Let i_0 be the label of the last such record,

$$i_0 := \min\{i \geq 1 : w_{i+1} \neq w_i\}.$$

If the weights w_i are summable then i_0 is finite. (Suppose that i_0 is infinite, that is, the records are all equally popular and have equal weights. Then the sum of an infinite number of such weights is not finite, which contradicts the summability criterion.)

Proposition 3.5.3.2 *Suppose that Second User's request probabilities for $i = 1, \dots, n$ are*

$$q_i = \frac{w_i}{\sum_{j=1}^{n} w_j}.$$

If $\sum_{i=1}^{\infty} w_i < \infty$ and $t \leq t(n,c) = c$, then for $n \geq \max\{3, i_0 + 1\}$,

$$TV(t) \geq (w_{i_0} - w_{i_0+1})e^{-c}$$

and hence time c is necessary for convergence to stationarity.

Proof. Again, without loss of generality, we may assume $\sum_{i=1}^{\infty} w_i = 1$. If K is the number of requests made by Controller in the time-interval $[0, t)$, then, for $0 < t \leq t(n,c) = c$,

$$\begin{aligned}
\tilde{d}(t) &= \max_c \left[P(S^{id}(t) \leq c) - P(S^{rev}(t) \leq c)\right] \\
&\geq P(S^{id}(t) \leq 1) - P(S^{rev}(t) \leq 1) \\
&= \sum_k \left[P(S^{id}(t) \leq 1) - P(S^{rev}(t) \leq 1) | K = k\right] P(K = k).
\end{aligned}$$

If $k = 0$, then by our choice of indices $(q_1 \geq q_2 \geq \cdots \geq q_n)$,

$$\begin{aligned}
&\left[P(S^{id}(t) \leq 1) - P(S^{rev}(t) \leq 1) | K = 0\right] \\
&\qquad \geq q_1 + q_2 - (q_n + q_{n-1}) \geq q_1 - q_n \\
&\qquad \geq \frac{w_{i_0} - w_{i_0+1}}{\sum_{j=1}^{n} w_j} = w_{i_0} - w_{i_0+1}.
\end{aligned}$$

If $k \geq 1$,

$$\begin{aligned}
&\Big[P(S^{id}(t) \leq 1) - P(S^{rev}(t) \leq 1) | K = k\Big] \\
&= p_1^k(q_1 + q_2) + \sum_{i=2}^{n} p_i^k (q_i + q_1) \\
&\quad + \mathbf{I}\{k > 1\} \sum_{\mathcal{J}} p_{i_1} \cdots p_{i_k} (q_{i_k} + q_{i_{k-1}}) \\
&\quad - \Big[p_n^k (q_n + q_{n-1}) + \sum_{i=1}^{n-1} p_i^k (q_i + q_n) \\
&\qquad + \mathbf{I}\{k > 1\} \sum_{\mathcal{J}} p_{i_1} \cdots p_{i_k} (q_{i_k} + q_{i_{k-1}}) \Big] \\
&= p_1^k (q_2 - q_n) + \sum_{i=2}^{n-1} p_i^k (q_1 - q_n) + p_n^k (q_1 - q_{n-1}) \\
&\geq (q_1 - q_n) \sum_{i=2}^{n-1} p_i^k \geq (q_1 - q_n) p_2^k \geq (w_{i_0} - w_{i_0+1}) p_2^k.
\end{aligned}$$

Here, $\mathcal{J}$ is the set of all indices $i_1, \ldots, i_k$ such that not all of them are equal. Now, since by the embedding, Controller's requests follow a Poisson process of intensity 1, we have

$$\tilde{d}(t) \geq \sum_{k \geq 0} (w_{i_0} - w_{i_0+1}) p_2^k \frac{t^k}{k!} e^{-t} = (w_{i_0} - w_{i_0+1}) e^{-t(1-p_2)} \geq (w_{i_0} - w_{i_0+1}) e^{-c}.$$

The statement follows from Corollary 3.3.1, equation (3.2).

□

As in Fill [22, Remark 3.6], we emphasize that the conditions $n \geq 3$ and $n \geq i_0 + 1$ are necessary. If $n = 2$ the permutation process is in stationarity as soon as Controller has made his first request and if $n \leq i_0$ Second User's requests are uniformly distributed.

Second User requests	necessary time	sufficient time
uniform	c	c
summable (e.g. geometric Zipf 's law, α>1)	c	cn
Zipf 's law, (α=1)	cn - $n \ln \ln n$	cn
Zipf 's law, 0<α<1	cn	cn
power law	cn	cn

Figure 3.5.4: *Rates of convergence.* Controller requests with uniformly distributed request probabilities.

3.5.4 Uniformly distributed Controller-requests

When Controller's request probabilities are uniformly distributed and Second User's requests are anything but uniform, Proposition 3.5.4.1 shows that time cn is sufficient. For uniform Second User requests we can do better than this, in fact, we already know that, according to Section 3.5.2, time c is both sufficient and necessary.

When Second User's requests have summable weights (e.g. geometric distribution, generalized Zipf's law with $\alpha > 1$) we know by Proposition 3.5.3.2 that time c is necessary for convergence to stationarity. In Example 3.5.4.2 - 3.5.4.4 we find necessary time for convergence when Second User requests according to the non-summable generalized Zipf's law with $\alpha < 1$, power law and Zipf's law. The results are collected in Figure 3.5.4. We conclude that since the necessary and sufficient time to convergence agree for uniform, Zipf's law, $0 < \alpha < 1$ and power law Second User requests, these are the 'true' rates of convergence.

Proposition 3.5.4.1 *If Controller's request probabilities are uniformly distributed, $p_i = 1/n, i = 1, \ldots, n$, and $t \geq t(n,c) = cn$ (for $n \geq 1$ and $c \geq 0$), then*

$$TV(t) \leq e^{-t/n} \leq e^{-c} \downarrow 0 \text{ as } c \uparrow \infty$$

and hence time cn is sufficient for convergence to stationarity.

Proof. By Theorem 3.4.1, for $t \geq t(c,n) = cn$,

$$TV(t) \quad \leq \quad \sum_{i=1}^{n} q_i e^{-p_i t} = e^{-t/n} \sum_{i=1}^{n} q_i = e^{-t/n} \leq e^{-c}.$$

□

Example 3.5.4.2 *If Controller's requests are uniformly distributed and Second User requests according to the generalized Zipf's law, $0 < \alpha < 1$, that is, for $i = 1, \ldots, n$,*

$$\begin{aligned} p_i &= 1/n, \\ q_i &= i^{-\alpha}/H_n(\alpha), \quad H_n(\alpha) = \sum_{i=1}^{n} i^{-\alpha}, \end{aligned}$$

then for $n \geq 4$ and $t \leq t(n,c) = cn$ ($c \geq 0$),

$$TV(t) \geq e^{-2c} \frac{1}{32}(1-\alpha)\left(2^{\alpha} - \left(\frac{4}{3}\right)^{\alpha}\right)$$

and hence time cn is necessary for convergence to stationarity.

Recall from Section 3.2 that

$$\Delta_{rev}(t) = \sum_{1 \leq i < j \leq n} (q_i - q_j) \frac{p_i}{p_i + p_j} e^{-(p_i+p_j)t}.$$

With our particular $\{p_i\}_{i=1}^n$ and $\{q_i\}_{i=1}^n$, we have

$$\Delta_{rev}(t) \quad = \quad e^{-2t/n} \frac{1}{2H_n(\alpha)} \sum_{1 \leq i < j \leq n} (i^{-\alpha} - j^{-\alpha}).$$

Summing only over a subset of $\{1 \leq i < j \leq n\}$, using the notation $\lfloor x \rfloor$ for the integer part of x,

$$\begin{aligned}
\sum_{1\le i<j\le n}(i^{-\alpha}-j^{-\alpha}) &\ge \sum_{4\le i<j\le n}(i^{-\alpha}-j^{-\alpha})\\
&\ge \sum_{i=\lfloor\frac{n}{4}\rfloor}^{\lfloor\frac{n}{2}\rfloor}\sum_{j=\lfloor\frac{3n}{4}\rfloor+1}^{n}(i^{-\alpha}-j^{-\alpha})\\
&\ge \sum_{i=\lfloor\frac{n}{4}\rfloor}^{\lfloor\frac{n}{2}\rfloor}\sum_{j=\lfloor\frac{3n}{4}\rfloor+1}^{n}\left(\lfloor\frac{n}{2}\rfloor^{-\alpha}-\left(\lfloor\frac{3n}{4}\rfloor+1\right)^{-\alpha}\right)\\
&\ge \frac{n}{4}\frac{n}{4}\left(\lfloor\frac{n}{2}\rfloor^{-\alpha}-\left(\lfloor\frac{3n}{4}\rfloor+1\right)^{-\alpha}\right)\\
&\ge \frac{n^{2-\alpha}}{16}\left(2^{\alpha}-\left(\frac{4}{3}\right)^{\alpha}\right),
\end{aligned}$$

since $i^{-\alpha}$ is decreasing in i. Hence

$$\Delta_{rev}(t)\ge e^{-2t/n}\frac{1}{32H_n(\alpha)}n^{2-\alpha}\left(2^{\alpha}-\left(\frac{4}{3}\right)^{\alpha}\right) \tag{3.11}$$

and by the integral comparison (3.4),

$$\frac{n^{1-\alpha}}{H_n(\alpha)}\ge 1-\alpha \text{ for } 0<\alpha<1,$$

we have

$$\frac{\Delta_{rev}(t)}{n}\ge\frac{e^{-2t/n}}{32}(1-\alpha)\left(2^{\alpha}-\left(\frac{4}{3}\right)^{\alpha}\right).$$

By Corollary 3.3.1,

$$\begin{aligned}
TV(t) &\ge \frac{\Delta_{rev}(t)}{n}\ge\frac{e^{-2t/n}}{32}(1-\alpha)\left(2^{\alpha}-\left(\frac{4}{3}\right)^{\alpha}\right)\\
&\ge \frac{e^{-2c}}{32}(1-\alpha)\left(2^{\alpha}-\left(\frac{4}{3}\right)^{\alpha}\right),
\end{aligned}$$

for $t\le t(n,c)=cn$.

$\square$

Remark. Considering $\Delta_{id}(t)$ instead of $\Delta_{rev}(t)$ gives exactly the same since for this particular case they are identical.

Example 3.5.4.3 *If Controller's requests are uniformly distributed and Second User requests according to power law, that is, for $\beta > 0$ and $i = 1, \ldots, n$,*

$$\begin{aligned} p_i &= 1/n, \\ q_i &= (n-i+1)^\beta / G_n(\beta), \quad G_n(\beta) = \sum_{i=1}^{n} i^\beta, \end{aligned}$$

then for $n \geq 8$ and $t \leq t(n,c) = cn$ $(c \geq 0)$,

$$TV(t) \geq \frac{\beta+1}{32(\beta+2)} \left(\left(\frac{1}{2}\right)^\beta - \left(\frac{1}{4}\right)^\beta \right) e^{-2c}$$

and hence time cn is necessary for convergence to stationarity.

Proceeding as in Example 3.5.4.2 we get for $n \geq 8$

$$\begin{aligned} \Delta_{rev}(t) &= \frac{e^{-2t/n}}{2G_n(\beta)} \sum_{1 \leq i < j \leq n} \left((n-i+1)^\beta - (n-j+1)^\beta \right) \\ &\geq \frac{e^{-2t/n}}{2G_n(\beta)} \sum_{i=\lfloor \frac{n}{4} \rfloor - 1}^{\lfloor \frac{n}{2} \rfloor - 1} \sum_{j=\lfloor \frac{3n}{4} \rfloor - 2}^{n-3} \left((n-i+1)^\beta - (n-j+1)^\beta \right) \\ &\geq \frac{e^{-2t/n}}{2G_n(\beta)} \frac{n}{4} \frac{n}{4} \left(\left(n - \lfloor \frac{n}{2} \rfloor \right)^\beta - \left(n - \lfloor \frac{3n}{4} \rfloor - 1 \right)^\beta \right) \\ &\geq \frac{e^{-2t/n}}{2G_n(\beta)} \frac{n}{4} \frac{n}{4} \left(\left(\frac{n}{2}\right)^\beta - \left(\frac{n}{4}\right)^\beta \right) \\ &= \frac{e^{-2t/n}}{32 G_n(\beta)} n^{\beta+2} \left(\left(\frac{1}{2}\right)^\beta - \left(\frac{1}{4}\right)^\beta \right). \end{aligned}$$

By the integral comparison (3.3) we have that

$$\frac{n^{\beta+1}}{G_n(\beta)} \geq \frac{\beta+1}{\beta+2}$$

and for $t \le t(n,c) = cn$,

$$\begin{aligned}\frac{\Delta_{rev}(t)}{n} &\geq \frac{e^{-2t/n}}{32}\frac{\beta+1}{\beta+2}\left(\left(\frac{1}{2}\right)^{\beta}-\left(\frac{1}{4}\right)^{\beta}\right)\\ &\geq \frac{e^{-2c}}{32}\frac{\beta+1}{\beta+2}\left(\left(\frac{1}{2}\right)^{\beta}-\left(\frac{1}{4}\right)^{\beta}\right).\end{aligned}$$

Again the result follows from Corollary 3.3.1.

□

Example 3.5.4.4 *If Controller's requests are uniformly distributed and Second User requests according to Zipf's law, that is, for $i = 1, \dots, n$,*

$$\begin{aligned} p_i &= 1/n, \\ q_i &= i^{-1}/H_n, \quad H_n = \sum_{i=1}^{n} i^{-1}, \end{aligned}$$

then for $n \ge 6$ and $t \le t(n,c) = cn - n\ln\ln n$ $(c \ge 0)$,

$$TV(t) \geq \frac{e^{-2c}}{48}$$

and hence time $cn - n\ln\ln n$ is necessary for convergence to stationarity.

These request probabilities imply $\alpha = 1$ in equation (3.11) we have

$$\Delta_{rev}(t) \geq \frac{ne^{-2t/n}}{48H_n}.$$

Hence, if $t \le t(n,c) = cn - n\ln\ln n$, $n \ge 6$, using the integral comparison (3.5) we get

$$\frac{\Delta_{rev}(t)}{n} \geq \frac{e^{-(2t/n+\ln H_n)}}{48} \geq \frac{e^{-(2t/n+2\ln\ln n)}}{48} \geq \frac{e^{-2c}}{48}$$

and by Corollary 3.3.1 we are done.

□

	Second User requests		
	power	*geometric*	
Controller requests		*similar preferences*	*opposite preferences*
Zipf 's law, $0<\alpha<1$	cn	$cn^{3/2}$	$cn^{3/2}$
Zipf 's law, $(\alpha=1)$	$cn(\ln n)^2$	$c(\ln n)^2$	$\frac{c(\ln n)^2}{1-\theta^n}$
Zipf 's law, $\alpha>1$	cn^α	c*	c*

* By Proposition 3.5.3.1.

Figure 3.5.5: *Sufficient time to convergence.* Zipf's law, $\alpha > 1$ and the geometric distribution have summable weights.

3.5.5 Permutation dependence — sufficient time

By our choice of indices we have $q_1 \geq q_2 \geq \cdots \geq q_n$. However, the relative order between the p_i's is arbitrary. By Lemma 3.2.1 the search cost is minimal for $p_1 \geq p_2 \geq \cdots \geq p_n > 0$ and maximal for $0 < p_1 \leq p_2 \leq \cdots \leq p_n$. We refer to these cases as *similar preferences* and *opposite preferences* respectively. In this section, we give for two examples using Lemma 3.5.5.1. In Example 3.5.5.2 - 3.5.5.3 we determine sufficient time for convergence for some combinations of request probabilities with both summable and non-summable weights. The results are summarized in Figure 3.5.5.

Lemma 3.5.5.1 *Assuming $p_i > 0, i = 1, \ldots, n$, then for $t > 0$*

$$TV(t) \leq \frac{e^{-1}}{t} \sum_{i=1}^{n} \frac{q_i}{p_i}.$$

Proof. By Theorem 3.4.1 we have

$$TV(t) \leq \sum_{i=1}^{n} q_i e^{-p_i t} = \sum_{i=1}^{n} \frac{q_i}{p_i t} p_i t e^{-p_i t} \leq \frac{e^{-1}}{t} \sum_{i=1}^{n} \frac{q_i}{p_i}$$

since $\max_{x \geq 0}(xe^{-x}) = e^{-1}$.

$\square$

Example 3.5.5.2 *If Controller requests according to the generalized Zipf's law, $\alpha > 0$, and Second User according to power law, $\beta > 0$, that is, for $i = 1, \ldots, n$,*

$$\begin{aligned} p_i &= i^{-\alpha}/H_n(\alpha), \\ q_i &= (n-i+1)^{\beta}/G_n(\beta), \end{aligned}$$

for similar preferences and

$$\begin{aligned} p_i &= (n-i+1)^{-\alpha}/H_n(\alpha), \\ q_i &= (n-i+1)^{\beta}/G_n(\beta), \end{aligned}$$

for opposite, where $H_n(\alpha) = \sum_{i=1}^{n} i^{-\alpha}$ and $G_n(\beta) = \sum_{i=1}^{n} i^{\beta}$. Then for $n \geq 6$ the time sufficient for convergence to stationarity is

(i) cn *if* $0 < \alpha < 1$,
(ii) $cn(\ln n)^2$ *if* $\alpha = 1$,
(iii) cn^α *if* $\alpha > 1$.

We show the result only for the case of similar preferences. For opposite preferences, the calculations are completely analogous and are therefore omitted.

By Lemma 3.5.5.1 we have

$$TV(t) \leq \frac{e^{-1}}{t}\frac{H_n(\alpha)}{G_n(\beta)}\sum_{i=1}^{n}(n-i+1)^\beta i^\alpha.$$

Applying Cauchy-Schwarz' inequality to the sum we see that

$$\begin{aligned} TV(t) &\leq \frac{e^{-1}}{t}\frac{H_n(\alpha)}{G_n(\beta)}\left[\sum_{i=1}^{n}(n-i+1)^{2\beta}\right]^{\frac{1}{2}}\left[\sum_{i=1}^{n} i^{2\alpha}\right]^{\frac{1}{2}} \\ &= \frac{e^{-1}}{t}\frac{H_n(\alpha)}{G_n(\beta)}[G_n(2\beta)]^{\frac{1}{2}}[G_n(2\alpha)]^{\frac{1}{2}}. \end{aligned} \tag{3.12}$$

By the integral comparison (3.3), we have

$$\begin{aligned} TV(t) &\leq \frac{e^{-1}}{t}\frac{H_n(\alpha)}{n^{\beta+1}/(\beta+1)}\left[\frac{n^{2\beta+1}}{2\beta+1}(2\beta+2)\right]^{\frac{1}{2}}\left[\frac{n^{2\alpha+1}}{2\alpha+1}(2\alpha+2)\right]^{\frac{1}{2}} \\ &= \frac{e^{-1}}{t}\frac{(\beta+1)}{\sqrt{2\beta+1}\sqrt{2\alpha+1}}H_n(\alpha)\frac{n^{\beta+\alpha+1}}{n^{\beta+1}}\sqrt{2(\beta+1)}\sqrt{2(\alpha+1)} \\ &= \frac{2e^{-1}}{t}H_n(\alpha)n^\alpha(\beta+1)\frac{\sqrt{\beta+1}\sqrt{\alpha+1}}{\sqrt{2\beta+1}\sqrt{2\alpha+1}}. \end{aligned}$$

Now, by the integral comparisons (3.4) - (3.6) we have for $0 < \alpha < 1$

$$TV(t) \leq n\frac{2e^{-1}}{t}\frac{(\beta+1)}{(1-\alpha)}\sqrt{\frac{(\beta+1)(\alpha+1)}{(2\beta+1)(2\alpha+1)}}, \tag{3.13}$$

for $\alpha = 1$ and $n \geq 6$

$$TV(t) \leq n(\ln n)^2\frac{2e^{-1}}{t}(\beta+1)\sqrt{\frac{2(\beta+1)}{3(2\beta+1)}}, \tag{3.14}$$

and for $\alpha > 1$

$$TV(t) \leq n^{\alpha} \frac{2e^{-1}}{t} \frac{\alpha(\beta+1)}{(\alpha-1)} \sqrt{\frac{(\beta+1)(\alpha+1)}{(2\beta+1)(2\alpha+1)}}. \tag{3.15}$$

For $0 < \alpha < 1$, $\alpha = 1$ and $\alpha > 1$, choosing $t \geq t(n,c)$, where $t(n,c)$ equals the respective time in the statement, the right hand side of the corresponding equation (3.13)-(3.15) is a function $H(c)$, independent of n, such that $H(c) \propto 1/c \downarrow 0$ as $c \uparrow \infty$ and we are done.

□

Example 3.5.5.3 *If Controller requests according to the generalized Zipf's law with $0 < \alpha \leq 1$ and Second User according to the geometric distribution, $0 < \theta < 1$, and their preferences are similar, that is, for $i = 1, \ldots, n$,*

$$\begin{aligned} p_i &= i^{-\alpha}/H_n(\alpha), \\ q_i &= \theta^i/K_n(\theta), \end{aligned}$$

where $H_n(\alpha) = \sum_{i=1}^n i^{-\alpha}$ and $K_n(\theta) = \sum_{i=1}^n \theta^i$, then for $n \geq 6$ the time sufficient for convergence to stationarity is

(i) $cn^{3/2}$ *if* $0 < \alpha < 1$,
(ii) $c(\ln n)^2$ *if* $\alpha = 1$.

If their preferences are opposite, that is, for $i = 1, \ldots, n$,

$$\begin{aligned} p_i &= (n+i-1)^{-\alpha}/H_n(\alpha), \\ q_i &= \theta^i/K_n(\theta), \end{aligned}$$

then for $n \geq 6$ the sufficient time is

(i) $cn^{3/2}$ *if* $0 < \alpha < 1$,
(ii) $c(\ln n)^2/(1-\theta^n)$ *if* $\alpha = 1$.

Recognizing in $K_n(\theta)$ the geometric series,

$$K_n(\theta) = \frac{\theta(1-\theta^n)}{1-\theta} \tag{3.16}$$

and by Lemma 3.5.5.1 we have

$$TV(t) \leq \frac{e^{-1}}{t}\frac{H_n(\alpha)(1-\theta)}{\theta(1-\theta^n)}\sum_{i=1}^{n}\theta^i i^\alpha. \tag{3.17}$$

When $0 < \alpha < 1$, we use Cauchy-Schwarz' inequality.

$$\sum_{i=1}^{n}\theta^i i^\alpha \leq \sqrt{\sum_{i=1}^{n}\theta^i}\sqrt{\sum_{i=1}^{n} i^\alpha} = \sqrt{\frac{\theta^2(1-\theta^{2n})}{1-\theta^2}G_n(2\alpha)}.$$

By the integral comparison (3.3),

$$\begin{aligned}\sum_{i=1}^{n}\theta^i i^\alpha &\leq \theta\sqrt{\frac{(1-\theta^{2n})}{(1-\theta^2)}\frac{(2\alpha+2)}{(2\alpha+1)}n^{2\alpha+1}} \\ &\leq n^{\alpha+1/2}\theta\sqrt{2\frac{1}{(1-\theta^2)}\frac{(\alpha+1)}{(2\alpha+1)}}.\end{aligned}$$

Hence, from (3.17) by the integral comparison (3.4) we have that

$$\begin{aligned}TV(t) &\leq n^{3/2}\frac{e^{-1}(1-\theta)}{t(1-\alpha)(1-\theta^n)}\sqrt{2\frac{1}{(1-\theta^2)}\frac{(\alpha+1)}{(2\alpha+1)}} \\ &\leq n^{3/2}\frac{e^{-1}}{t(1-\alpha)}\sqrt{2\frac{1}{(1-\theta^2)}\frac{(\alpha+1)}{(2\alpha+1)}}\end{aligned}$$

since $(1-\theta)/(1-\theta^n) \leq 1$ for $0 < \theta < 1$.
If $\alpha = 1$ we can determine the sum exactly;

$$\sum_{i=1}^{n} i\theta^i = \frac{\theta(1-\theta^n - n(\theta^n - \theta^{n+1}))}{(1-\theta)^2} \leq \frac{1-\theta^n}{(1-\theta)^2}. \tag{3.18}$$

Now, using the integral comparison (3.5) for H_n we get

$$TV(t) \leq \frac{e^{-1}}{t\theta(1-\theta)}(\ln n)^2$$

provided that $n \geq 6$.

For $t \geq t(n,c)$, where $t(n,c)$ is taken to be $c(\ln n)^2$ and $cn^{3/2}$ for $\alpha = 1$ and $0 < \alpha < 1$ respectively, $TV(t) \leq H(c) \propto 1/c \downarrow 0$ as $c \uparrow \infty$. Hence, these are the sufficient times.

Analogous calculations give the results for opposite preferences.

□

Chapter 4

Time-dependent requests

So far we have considered only time-independent request probabilities but what if the popularities of the records change in time? One of the advantages of self-organizing schemes such as the move-to-front rule is that they adapt to changes in the request pattern and therefore are especially suited for situations where the request probabilities may change in time. However, up to now no results on this subject seem to have been published.

This chapter is devoted to Second User's search cost for move-to-front with time-dependent request probabilities. Again, we study the cost due to a request by Second User at the current instant. Our model, see Sections 4.1 and 4.2, allows Controller to have changed his taste in the past as deterministic functions of time. We still assume the requests to be made independently and hence, as before, previous actions by Second User do not affect the present since the cost is determined only by the current request and the list arrangement at this moment.

As before, all results for Second User's search cost reduce to the standard single-user move-to-front analogues when Controller and Second User are taken to have identical tastes. These are given as corollaries and since allowing time-dependence is a new angle to the study of self-organizing search, so are these single-user results.

First, in Section 4.1, we consider the search cost when there are a fixed number of records in the list. We show how to extend the results from Chapter 2 to this new situation of time-dependence and carry the generalization through for some of them.

In the remainder of the chapter we study the search cost as the number of records n tends to infinity. In Theorem 4.3.1 we give the long-term asymptotic distribution (as $n \to \infty$) of the fraction of records we need to check before we find the one requested by Second User. By 'long-term' we understand that the influence of the initial order of the records in the list is negligible and hence we can regard the permutation process to be governed only by move-to-front.

In Section 4.4 we take into account the transient 'cold start' effects of the initial permutation of the list. The long-term behaviour of course follows, but to make the presentation easier to digest, we save the complication of transient behaviour depending on how the list starts until we have become acquainted with the long-term performance.

In Section 4.5 we use Theorem 4.3.1 to derive results about stochastic ordering and in Section 4.6 we consider some examples.

4.1 Finitely many records

All the results in Chapter 2 may be generalized to the case of time-dependent request probabilities. The generalization is straightforward and hence we confine ourselves to deriving the generating function for Second User's long-term search cost (Theorem 4.1.3) and the expression for the probability function at time t, given the initial permutation of the list (Theorem 4.1.2).

Recall the embedding in Section 2.2, Figure 2.2.1. Now that Controller's popularities are no longer constant in time, the intensities of the Poisson processes modelling the requests must also be time-dependent.

Let Controller request record i, $i = 1, \ldots, n$, at the time-points of an independent Poisson process evolving in backward time with deterministic, time-dependent intensity ${}^{t}p_i(t^*) > 0$ (except possibly at finitely many instants t^* where the intensity may be 0), where at each backward-time instant[1] t^*, the intensities satisfy $\sum_{i=1}^{n} {}^{t}p_i(t^*) = 1$. The idea behind this notation is to guide the association path along a certain track:

At the current instant t, 'now', we reset the *backward*-time, that is, $t^* := 0$. Hence, t defines the *starting* point of the backward-time and the superindex t of the intensities is put *before* the intensity function as a reminder that this is the point in time where the evolution of the process (in backward time) begins. This way, confusion about whether t might be something other than an index is hopefully avoided. Looking back, the Poisson

[1] For clarity, all instants in backward-time will be given the superscript $*$.

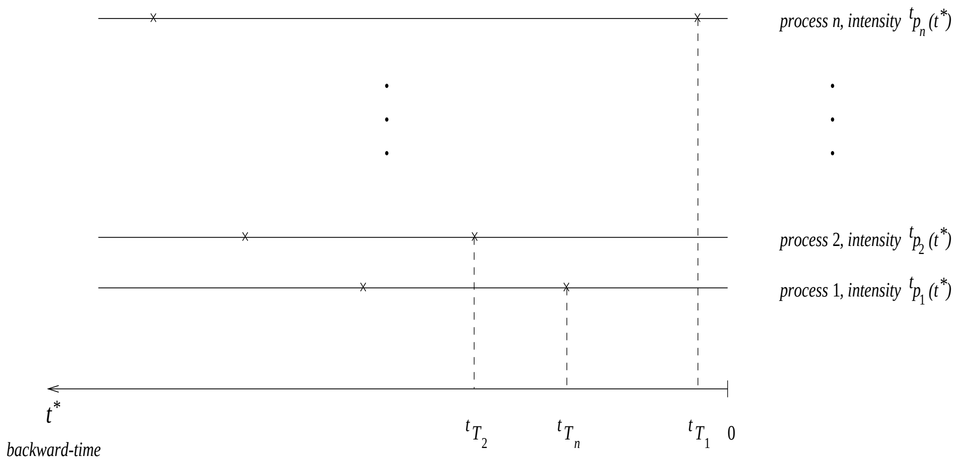


Figure 4.1.1: *Poisson embedding.* Controller requests i at time-points of a Poisson process with intensity $^tp_i(t^*)$. ^tT_i is the time elapsed since Controller last requested record i.

processes model Controller's requests in the past, hence the intensity is a function of the backward-time t^*.

A Second User request made 'now' is for record i with probability ^tq_i, where $\sum_{i=1}^n {}^tq_i = 1$ regardless of t. Second User's taste may also change in time, that is, we allow $^{t_1}q_i \neq {}^{t_2}q_i$ for $t_1 \neq t_2$, but since Second User's current taste is sufficient to determine the current search cost, we may think of $^tq_1, \ldots, {}^tq_n$ as a set of constants. The superindex t is a reminder that at a different instant, Second User's request probabilities may be an entirely different set of constants.

Let ^tT_i denote the backward waiting time until the 'first' occurrence in process i, Figure 4.1.1. The purpose of the superindex t is again to remind us that the distribution depends on where 'now', and consequently the starting point of the backward-time, is. We have

$$P({}^tT_i > u^*) = \exp\left\{ -\int_0^{u^*} {}^tp_i(t^*)dt^* \right\}$$

with density

$$f_{^tT_i}(u^*) = {}^tp_i(u^*)\exp\Big\{-\int_0^{u^*} {}^tp_i(t^*)dt^*\Big\}.$$

First, we need a generalized version of Proposition 2.3.1.1. With the notation from Section 2.3.1, we define indicator random variables $\mathbf{1}_{ij}(t)$, $i, j = 1, \ldots, n$, indicating whether or not i precedes j in the list at time t and search cost[2] $S_i^\pi(t)$ for record i at time t when the list is initially in permutation π. Since the initial list arrangement is known, the indicators $\mathbf{1}_{ij}(0)$ are deterministic — $\mathbf{1}_{ij}(0) = 1$ if and only if i precedes j in the permutation π.

Proposition 4.1.1

$$\begin{aligned} P(S_i^\pi(t) = k) &= \int_0^t P_{u^*}(S_i^\pi(t) = k)\ {}^tp_i(u^*)\exp\Big\{-\int_0^{u^*} {}^tp_i(t^*)dt^*\Big\}du^* \\ &\quad + P_t(S_i^\pi(t) = k)\exp\Big\{-\int_0^t {}^tp_i(t^*)dt^*\Big\}, \end{aligned}$$

where $P_{u^*}(S_i^\pi(t) = k)$ is the probability function of the distribution

$$\begin{cases} \oplus_{j\neq i} Bern\Big(1 - \exp\Big\{-\int_0^{u^*} {}^tp_j(t^*)dt^*\Big\}\Big) & \text{if } u^* \in (0, t) \\ \oplus_{j\neq i} Bern\Big(1 - \mathbf{1}_{ij}(0)\exp\Big\{-\int_0^{t} {}^tp_j(t^*)dt^*\Big\}\Big) & \text{if } u^* = t. \end{cases}$$

Proof. This is a straightforward generalization of the proof of Proposition 2.3.1.1. Again, let ${}^tL_i$ be the time elapsed since Controller's last request for record i, $i = 1, \ldots, n$. Then, ${}^tL_i$ is distributed as $\min\{{}^tT_i, t\}$. When conditioning on ${}^tL_i = u^*$, we have two cases:
If $0 < u^* < t$,

$$P_{u^*}(\mathbf{1}_{ji}(t) = 1) = P({}^tL_j < u^*) = 1 - \exp\Big\{-\int_0^{u^*} {}^tp_j(t^*)dt^*\Big\}.$$

[2] Recall that the search cost depends on the number n of records although this dependence is suppressed in the notation.

If $u^* = t$,

$$P_t(\mathbf{1}_{ji}(t) = 1) = 1 - \exp\Big\{ - \int_0^t {}^t p_j(t^*) dt^* \Big\} + \mathbf{1}_{ji}(0) \exp\Big\{ - \int_0^t {}^t p_j(t^*) dt^* \Big\},$$

and we are done.

$\square$

Theorem 4.1.2 *For a list started in permutation π the probability function for Second User's search cost at time t is*

$$P(S^\pi(t) = k) = \sum_{i=1}^n {}^t q_i P(S_i^\pi(t) = k),$$

where the distribution of $S_i^\pi(t)$ is as in Proposition 4.1.1.

In the previous chapter where the request probabilities did not change in time, the search cost eventually reached stationarity. Now, however, as the popularities keep changing, so does the search cost distribution. Hence, the long-term search cost $S^{long\text{-}term}(t)$ depends heavily on t in that the time-point t defines the starting point of the backward-time. Since the intensities are time-dependent, we may get very different results for different choices of backward-time starting points. There is one great simplification though which we achieve by studying only the long-term behaviour — the list will have 'forgotten' it's initial arrangement and hence the dependence on π will have worn off.

Theorem 4.1.3 *The generating function for Second User's long-term search cost is*

$$E\left(z^{S^{long\text{-}term}(t)}\right) = \int_0^\infty e^{-u^*} \left[\sum_{i=1}^n \frac{{}^t q_i \; {}^t p_i(u^*)}{1 + \left(\exp\left\{ \int_0^{u^*} {}^t p_i(t^*) dt^* \right\} - 1 \right) z} \right] \times \left[\prod_{j=1}^n \left(1 + \left(\exp\left\{ \int_0^{u^*} {}^t p_j(t^*) dt^* \right\} - 1 \right) z \right) \right] du^*.$$

Proof. As we are studying the long-term behaviour, the clock has been running long enough to ensure that the 'memory' of the initial permutation has been lost and hence the search cost distribution is the one we obtain by letting u^* in the right-hand side of Proposition 4.1.1 range from 0 to infinity. Do not confuse the upper bound on u^*, labeled t in Proposition 4.1.1, with the current instant t — any point in time great enough to satisfy the long-term criterion. We have that $P_{u^*}(S_i^{long\text{-}term}(t) = k)$ is the probability function of

$$\oplus_{j \neq i} Bern\Bigg(1 - \exp\Bigg\{ -\int_0^{u^*} {}^t p_j(t^*)dt^* \Bigg\}\Bigg) \qquad u^* \in (0, \infty).$$

Conditioning on ${}^tT_i = u^*$, we get

$$\begin{aligned}
&E\left(z^{S^{long\text{-}term}(t)}\right) \\
&= \sum_i {}^t q_i \int_0^\infty {}^t p_i(u^*) \exp\Bigg\{ -\int_0^{u^*} {}^t p_i(t^*)dt^* \Bigg\} \\
&\quad \times \prod_{j \neq i} \Bigg[\exp\Bigg\{ -\int_0^{u^*} {}^t p_j(t^*)dt^* \Bigg\} \\
&\qquad + \Bigg(1 - \exp\Bigg\{ -\int_0^{u^*} {}^t p_j(t^*)dt^* \Bigg\}\Bigg) z \Bigg] du^* \\
&= \int_0^\infty \sum_i \left[{}^t q_i \; {}^t p_i(u^*) \frac{\prod_j \left[1 + \left(\exp\left\{ \int_0^{u^*} {}^t p_j(t^*)dt^* \right\} - 1 \right) z \right]}{1 + \left(\exp\left\{ \int_0^{u^*} {}^t p_i(t^*)dt^* \right\} - 1 \right) z} \right] \\
&\quad \times \exp\left\{ -\int_0^{u^*} \sum_j {}^t p_j(t^*)dt^* \right\} du^*
\end{aligned}$$

$$= \int_0^\infty e^{-u^*} \left[\sum_i \frac{{}^t q_i \; {}^t p_i(u^*)}{1 + \left(\exp\left\{ \int_0^{u^*} {}^t p_i(t^*) dt^* \right\} - 1 \right) z} \right] \times \left[\prod_j \left(1 + \left(\exp\left\{ \int_0^{u^*} {}^t p_j(t^*) dt^* \right\} - 1 \right) z \right) \right] du^*$$

since for every $t^* \in [0, \infty)$, $\sum_{j=1}^n {}^t p_j(t^*) = 1$.

$\square$

From Theorem 4.1.2 and Theorem 4.1.3 new results for standard single-user move-to-front with time-dependent request probabilities follow when we let Second User be identical to Controller.

Corollary 4.1.4 *For a list started in permutation π the probability function a single user's search cost at time t is*

$$P(S^\pi(t) = k) = \sum_{i=1}^n {}^t p_i(0) P(S_i^\pi(t) = k),$$

where the distribution of $S_i^\pi(t)$ is as in Proposition 4.1.1.

Corollary 4.1.5 *The generating function for the long-term search cost for a single user is*

$$E\left(z^{S^{long\text{-}term}(t)} \right) = \int_0^\infty e^{-u^*} \left[\sum_i \frac{{}^t p_i(0) \; {}^t p_i(u^*)}{1 + \left(\exp\left\{ \int_0^{u^*} {}^t p_i(t^*) dt^* \right\} - 1 \right) z} \right] \times \left[\prod_j \left(1 + \left(\exp\left\{ \int_0^{u^*} {}^t p_j(t^*) dt^* \right\} - 1 \right) z \right) \right] du^*.$$

4.2 Embedding in a two-dimensional Poisson process

In the rest of this chapter, we shall study Second User's search cost when the number of records in the list tends to infinity. In order to do so, we use a slightly different embedding model. Intuitively, we start from the model in Section 4.1, cf. Figure 4.1.1. Now, instead of having an independent Poisson process modelling Controller's requests for each record, we let a two-dimensional Poisson process govern the accesses. For this purpose, we need to 'continuize' the requests — let us say that Controller may call any real number between 0 and n and we interpret a call for any number in $(k-1, k]$ as a request for record k.

As we shall let n tend to infinity, we need to rescale time and 'space' (the requests axis) in order to get natural limits. Let Controller call any real number in $(0, 1]$ and interpret a call for a number in $((k-1)/n, k/n]$ as a request for record k. As n increases, the slices defining requests for the different records (Figure 4.2.1) become thinner and we need to wait longer in order to register an occurrence in a certain slice. In other words, as 'space' is rescaled, we also need to rescale time. Let nt be the genuine, unrescaled time elapsed since the clock was started. Then t is the rescaled current time-point.

Formally, let a two-dimensional Poisson process with deterministic time-dependent intensity ${}^{t}p(x, t^*)$ model Controller's requests as in Figure 4.2.1. Here, x is the rescaled, continuized record space and t^* is the rescaled backward-time. Again, the /rescaled/ backward-time is reset at the time-point of the current Second User request. At this instant (still conditional on that Second User makes a request at all), Second User requests record k with probability

$$ {}^{nt}q_k(n) := \int\limits_{(k-1)/n}^{k/n} {}^{t}q(x)dx, \qquad k = 1, \ldots, n, \tag{4.1} $$

where (regardless of t) ${}^{t}q(x) > 0$ for all (except finitely many) $x \in (0, 1]$ and $\int_0^1 {}^{t}q(x)dx = 1$. Interpret the left-hand side of (4.1) as the request probability in unrescaled time and 'space' while the right-hand side is the corresponding quantity after scaling.

Define Controller's request intensity for record k

$$ {}^{nt}p_k(t^*,n) := n \int\limits_{(k-1)/n}^{k/n} {}^{t}p(x,t^*)dx, \qquad k = 1,\ldots,n, \tag{4.2} $$

where for each (except finitely many) $x \in (0,1]$, ${}^{t}p(x,t^*) > 0$ (except for a finite number of instants t^*) and for each t^*, $\int_0^1 {}^{t}p(x,t^*)dx = 1$. Heuristically, this definition is motivated as follows: Interpret the left-hand side of (4.2) as the intensity of Controller requesting record k at (genuine) backward-time nt^*. Or, differently put, the probability that Controller requests record k in the backward-time interval $(nt^*, nt^* + d(nt^*)]$ is given by ${}^{nt}p_k(t^*,n)d(nt^*)$.

Now, rescale space and time. The probability that Controller calls a number in $((k-1)/n, k/n]$ in the rescaled backward-time interval $(t^*, t^*+dt^*]$ is then given by $n\int_{(k-1)/n}^{k/n} {}^{t}p(x,t^*)dxdt^*$, since $d(nt^*) = ndt^*$. Hence, the multiplicative factor n appears on the right-hand side as a consequence of the scaling. We return to this expression in the next section when we give sufficient regularity conditions on the intensity ${}^{t}p(x,t^*)$.

Conditional on that Second User requests record k at the current instant nt, let ${}^{nt}T_k$ be the rescaled time since the last occurrence of an event in the slice $x \in ((k-1)/n, k/n]$ in Figure 4.2.1. Then, provided that the clock has been running long enough (the event did not occur before the clock was started), we may interpret ${}^{nt}T_k$ as the (rescaled) time elapsed since Controller last requested record k.

Under this model,

$$ P({}^{nt}T_k > v^*) = \exp\Big\{ -\int\limits_0^{v^*} {}^{nt}p_k(t^*,n)dt^* \Big\} \tag{4.3} $$

with density

$$ f_{{}^{nt}T_k}(v^*) = {}^{nt}p_k(v^*,n)\exp\Big\{ -\int\limits_0^{v^*} {}^{nt}p_k(t^*,n)dt^* \Big\}. \tag{4.4} $$

(Note that since we reason in terms of backward time, v^* is positive.)

As the number of records n tends to infinity, so does the search cost except for degenerate or at least very skew request distributions[3]. For this

[3]Such as for instance summable weights distributions, cf. Section 3.5.3.

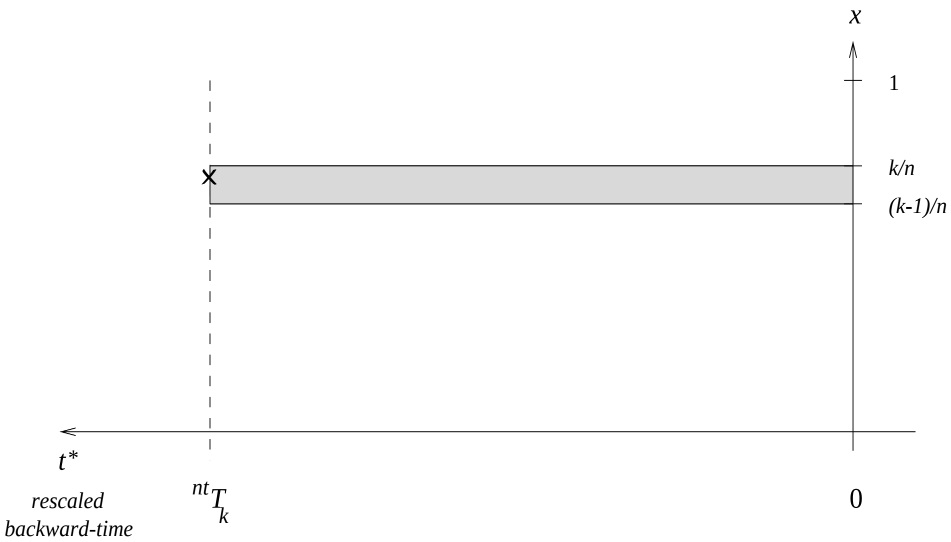


Figure 4.2.1: *Embedding in two-dimensional Poisson process.* At backward-time 0 ('now') Second User requests record k. $^{nt}T_k$ is the rescaled (that is, $n \cdot {}^{nt}T_k$ is the genuine, unrescaled) time elapsed since Controller last requested this record. In terms of the embedding, $^{nt}T_k$ is the waiting time until the first occurrence (in rescaled backward-time) in the shaded slice in a two-dimensional Poisson process with intensity $^{t}p(x, t^*)$.

reason, we shall study the search cost divided by the number of records, that is to say, the *fraction* of records we need to check before the one requested by Second User is found.

4.3 Asymptotic long-term distribution

The main result of this section is Theorem 4.3.1 where we give the asymptotic long-term distribution of the search cost fraction. Some examples are given in Section 4.6.

In the proof of Theorem 4.3.1 we need a law of large numbers. This result, Lemma 4.3.3, has been detached and is given before the actual proof of Theorem 4.3.1.

We make the following regularity assumptions: For a record $\lceil nx \rceil$,

$$ {}^{nt}p_{\lceil nx \rceil}(t^*, n) = n \int_{(\lceil nx \rceil - 1)/n}^{\lceil nx \rceil/n} {}^{t}p(y, t^*)dy \longrightarrow {}^{t}p(x, t^*) \tag{4.5} $$

and similarly

$$ n\ {}^{nt}q_{\lceil nx \rceil}(n) = n \int_{(\lceil nx \rceil - 1)/n}^{\lceil nx \rceil/n} {}^{t}q(y)dy \longrightarrow {}^{t}q(x) \tag{4.6} $$

as $n \to \infty$, where for each t^*, ${}^{t}p(x, t^*)$ and ${}^{t}q(x)$ are probability density functions.

Remark. In particular, suppose that ${}^{t}Q(x)$ is differentiable such that ${}^{t}q(x) = \frac{d}{dx}{}^{t}Q(x)$. Then

$$ n \int_{(\lceil nx \rceil - 1)/n}^{\lceil nx \rceil/n} {}^{t}q(y)dy = \frac{{}^{t}Q\left(\frac{\lceil nx \rceil}{n}\right) - {}^{t}Q\left(\frac{\lceil nx \rceil - 1}{n}\right)}{\frac{1}{n}} \longrightarrow \frac{d}{dx}{}^{t}Q(x) = {}^{t}q(x) $$

as $n \to \infty$, and hence the condition (4.6) is satisfied.

Theorem 4.3.1 *Assume that the regularity conditions (4.5) and (4.6) hold. Then, the long-term distribution function for the search cost due to a Second User request at time nt as a fraction of the number of records satisfies*

$$P(S^{long\text{-}term}(nt)/n \le s) \longrightarrow 1 - \int_0^1 {}^t q(x) \exp\left\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \right\} dx \quad (4.7)$$

as $n \to \infty$ where $v^*(s)$ is such that

$$\int_0^1 \exp\left\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \right\} dx = 1 - s. \quad (4.8)$$

The analogous result for single-user move-to-front follows immediately:

Corollary 4.3.2 *Assume that the regularity condition (4.5) holds. Then, the long-term distribution function for the search cost as a fraction of the number of records due to a request at time nt for single-user move-to-front satisfies*

$$P(S^{long\text{-}term}(nt)/n \le s) \longrightarrow 1 - \int_0^1 {}^t p(x, 0) \exp\left\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \right\} dx$$

as $n \to \infty$ where $v^*(s)$ is defined by (4.8).

Let $S_k^{long\text{-}term}(nt)$ denote Second User's (long-term) cost of searching record k, $k = 1, \ldots, n$, at time nt. With this notation, we now claim the following:

Given that at time nt Second User requests record $\lceil nx \rceil$ and that time nv^* has elapsed since Controller's last request for this record (that is, conditional on ${}^{nt}T_{\lceil nx \rceil} = v^*$), $S_{\lceil nx \rceil}^{long\text{-}term}(nt)/n$ converges in distribution to a point-mass[4] in

$$1 - \int_0^1 \exp\left\{ - \int_0^{v^*} {}^t p(y, t^*) dt^* \right\} dy.$$

[4]Or equivalently, the corresponding random variables converge in probability.

Lemma 4.3.3 *Assume that the regularity conditions (4.5) and (4.6) hold. Then, for every $x \in (0,1]$ and $v^* \in (0,\infty)$,*

$$\mathcal{L}\left(\frac{S_{\lceil nx\rceil}^{\textit{long-term}}(nt)}{n} - \left[1 - \int_0^1 \exp\left\{-\int_0^{v^*} {}^t p(y,t^*)dt^*\right\}dy\right] \,\middle|\, {}^{nt}T_{\lceil nx\rceil} = v^*\right) \longrightarrow \delta_0$$

as $n \to \infty$, where δ_0 denotes a one-point distribution in 0.

Proof. Introduce indicators $I_i,\ i = 1,\dots,n$. $I_i = 1$ if record $\lceil nx\rceil$ precedes record i at the time of the current request, that is, if i has *not* been requested by Controller in the rescaled backward-time interval $(0, v^*)$; $I_i = 0$ otherwise. By (4.3) we have

$$E\left[S_{\lceil nx\rceil}^{\textit{long-term}}(nt)/n \,\middle|\, {}^{nt}T_{\lceil nx\rceil} = v^*\right] = E\left[\frac{1}{n}\sum_{\substack{i=1\\ i\neq\lceil nx\rceil}}^{n}(1 - I_i)\right]$$

$$= \frac{n-1}{n} - \frac{1}{n}\sum_{\substack{i=1\\ i\neq\lceil nx\rceil}}^{n} \exp\left\{-\int_0^{v^*} {}^{nt}p_i(t^*,n)dt^*\right\}$$

$$= \frac{n-1}{n} - \int_0^1 \mathbf{I}\left\{y \notin \left(\frac{\lceil nx\rceil - 1}{n}, \frac{\lceil nx\rceil}{n}\right]\right\}$$

$$\times \exp\left\{-\int_0^{v^*} {}^{nt}p_{\lceil ny\rceil}(t^*,n)dt^*\right\}dy \tag{4.9}$$

$$\longrightarrow 1 - \int_0^1 \exp\left\{-\int_0^{v^*} {}^t p(y,t^*)dt^*\right\}dy \tag{4.10}$$

as $n \to \infty$, where in (4.9) we have changed the sum into an integral and made the substitution $y := i/n$. The limit (4.10) follows by the regularity condition (4.5).

It remains to show that $\mathrm{Var}[S_{\lceil nx\rceil}^{\textit{long-term}}(nt)/n \mid {}^{nt}T_{\lceil nx\rceil} = v^*] \longrightarrow 0$. Since the indicators are independent (by the embedding),

$$\begin{aligned}
&\operatorname{Var}\left[S^{\textit{long-term}}_{\lceil nx\rceil}(nt)/n \;\middle|\; {}^{nt}T_{\lceil nx\rceil}=v^*\right]\\
&= \operatorname{Var}\left[\frac{1}{n}\sum_{\substack{i=1\\ i\neq\lceil nx\rceil}}^{n}(1-I_i)\right] = \frac{1}{n^2}\sum_{\substack{i=1\\ i\neq\lceil nx\rceil}}^{n}\operatorname{Var}[I_i]\\
&= \frac{1}{n^2}\sum_{\substack{i=1\\ i\neq\lceil nx\rceil}}^{n}\exp\left\{-\int_0^{v^*}{}^{nt}p_i(t^*,n)dt^*\right\}\left(1-\exp\left\{-\int_0^{v^*}{}^{nt}p_i(t^*,n)dt^*\right\}\right)\\
&\leq \frac{1}{n^2}\sum_{\substack{i=1\\ i\neq\lceil nx\rceil}}^{n}\exp\left\{-\int_0^{v^*}{}^{nt}p_i(t^*,n)dt^*\right\} \qquad (4.11)\\
&= \frac{1}{n}\int_0^1 \mathbf{I}\left\{y\notin\left(\frac{\lceil nx\rceil-1}{n},\frac{\lceil nx\rceil}{n}\right]\right\}\exp\left\{-\int_0^{v^*}{}^{nt}p_{\lceil ny\rceil}(t^*,n)dt^*\right\}dy. \qquad (4.12)
\end{aligned}$$

Again, due to the regularity condition (4.5), as $n\to\infty$ (4.12) behaves like

$$\frac{1}{n}\int_0^1 \exp\left\{-\int_0^{v^*}{}^{t}p(y,t^*)dt^*\right\}dy \longrightarrow 0.$$

□

Proof of Theorem 4.3.1. Conditioning twice, first with respect to the record (k) requested by Second User at time t and secondly with respect to the rescaled time (v^*) elapsed since Controller last requested this record, we determine the distribution function straightforwardly:

$$\begin{aligned}
&P(S^{\textit{long-term}}(nt)/n\leq s)\\
&= \sum_{k=1}^{n}{}^{nt}q_k(n)\int_0^\infty P(S_k^{\textit{long-term}}(nt)/n\leq s|\;{}^{nt}T_k=v^*)f_{{}^{nt}T_k}(v^*)dv^*
\end{aligned}$$

$$= \sum_{k=1}^{n} {}^{nt}q_k(n) \int_0^\infty P(S_k^{long\text{-}term}(nt)/n \le s|\ {}^{nt}T_k = v^*)$$
$$\times {}^{nt}p_k(v^*, n) \exp\Big\{ - \int_0^{v^*} {}^{nt}p_k(t^*, n) dt^* \Big\} dv^*,$$

where the distribution of ${}^{nt}T_k$ is known from the embedding, cf. (4.4). Changing the sum into an integral and substituting $x := k/n$ we get

$$P(S^{long\text{-}term}(nt)/n \le s)$$
$$= \int_0^1 n\ {}^{nt}q_{\lceil nx \rceil}(n) \int_0^\infty P(S_{\lceil nx \rceil}^{long\text{-}term}(nt)/n \le s|^{nt}T_{\lceil nx \rceil} = v^*)$$
$$\times {}^{nt}p_{\lceil nx \rceil}(v^*, n) \exp\Big\{ - \int_0^{v^*} {}^{nt}p_{\lceil nx \rceil}(t^*, n) dt^* \Big\} dv^* dx$$
$$\longrightarrow \int_0^1 {}^{t}q(x) \int_0^\infty \mathbf{I}\Big\{ 1 - \int_0^1 \exp\Big\{ - \int_0^{v^*} {}^{t}p(y, t^*) dt^* \Big\} dy \le s \Big\}$$
$$\times {}^{t}p(x, v^*) \exp\Big\{ - \int_0^{v^*} {}^{t}p(x, t^*) dt^* \Big\} dv^* dx, \tag{4.13}$$

as $n \to \infty$, by the law of large numbers, Lemma 4.3.3, and by the regularity assumptions (4.5) and (4.6).

Let us simplify this expression. Since

$$1 - \int_0^1 \exp\Big\{ - \int_0^{v^*} {}^{t}p(y, t^*) dt^* \Big\} dy$$

is increasing in v^*, the indicator is 1 only for small enough values of v^*, cf. Figure 4.3.1. Let $v^*(s)$ be the limit value, that is, the greatest value of v^* such that the inequality is satisfied. In other words, define $v^*(s)$ such that

$$\int_0^1 \exp\Big\{ - \int_0^{v^*(s)} {}^{t}p(y, t^*) dt^* \Big\} dy = 1 - s.$$

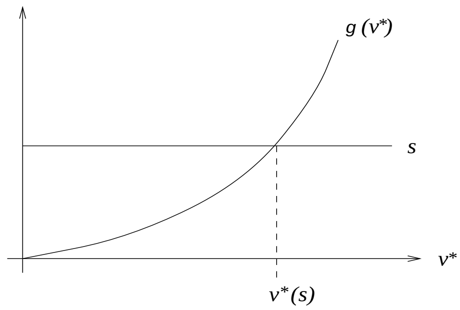


Figure 4.3.1: $v^*(s) : g(v^*) := 1 - \int_0^1 \exp\left\{ - \int_0^{v^*} {}^t p(y, t^*) dt^* \right\} dy.$

Then, for $v^* > v^*(s)$ the indicator is 0. Rewriting (4.13) as the integral merely over those v^* such that the indicator is 1, we have,

$$
\begin{aligned}
& P(S^{long\text{-}term}(nt)/n \leq s) \\
\longrightarrow \quad & \int_0^1 {}^t q(x) \int_0^{v^*(s)} {}^t p(x, v^*) \exp\left\{ - \int_0^{v^*} {}^t p(x, t^*) dt^* \right\} dv^* dx \\
= \quad & \int_0^1 {}^t q(x) \left(1 - \exp\left\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \right\} \right) dx \\
= \quad & 1 - \int_0^1 {}^t q(x) \exp\left\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \right\} dx
\end{aligned}
$$

and we are done.

□

4.4 General asymptotic cost distribution

We now extend the distribution of the asymptotic search cost fraction in Theorem 4.3.1 to include cold start effects — the short-term, transient behaviour, depending on the initial list arrangement.

Suppose that the starting permutation π of the records in the list is known. Then, for each record, we know which ones it is initially preceded by. Hence, for any $x \in (0,1]$, we can construct a set

$$A_x^\pi := \{y \in (0,1] : \lceil nx \rceil \text{ precedes } \lceil ny \rceil \text{ in permutation } \pi\}. \tag{4.14}$$

(Hence $y \in A_x^\pi$ if $\lceil ny \rceil$ is initially *after* $\lceil nx \rceil$, which motivates the choice of notation.) The reason why we choose to label the records in this way is again to prepare for the rescaling needed for the asymptotics as $n \to \infty$.

We study the asymptotic distribution of the search cost as a fraction of n at time nt as n tends to infinity.

Theorem 4.4.1 *Assume that the regularity conditions (4.5) and (4.6) hold and that the initial permutation of the list is π. Then, the distribution function for the search cost as a fraction of the number of records due to a Second User request at time nt satisfies*

$$\begin{aligned} &P(S^\pi(nt)/n \le s) \longrightarrow \\ &\quad 1 - \int_0^1 {}^t q(x) \exp\Big\{ - \int_0^{\min\{t, v^*(s)\}} {}^t p(x, t^*) dt^* \Big\} dx \\ &\quad + \mathbf{I}\{t < v^*(s)\} \int_{x \notin A^\pi_{x(s)}} {}^t q(x) \exp\Big\{ - \int_0^t {}^t p(x, t^*) dt^* \Big\} dx \end{aligned} \tag{4.15}$$

as $n \to \infty$ where $v^*(s)$ is such that

$$\int_0^1 \exp\Big\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \Big\} dx = 1 - s \tag{4.16}$$

and A_x^π defines the records initially after $\lceil nx \rceil$ in the list, cf. (4.14) and the limit $A^\pi_{x(s)}$ is given by

$$\int_{x \in A^\pi_{x(s)}} \exp\Big\{ - \int_0^t {}^t p(x, t^*) dt^* \Big\} dx = 1 - s. \tag{4.17}$$

Remark. When $t \geq v^*(s)$, we recapture the long-term asymptotic expression of Theorem 4.3.1. Hence, for each s, $v^*(s)$ is the (rescaled) waiting time required for the influence of the initial arrangement to wear off. See also pp. 127 – 128.

As before, the single-user analogue follows immediately.

Corollary 4.4.2 *Assume that the regularity condition (4.5) holds. Then, the distribution function for the search cost as a fraction of the number of records due to a request at time nt for single-user move-to-front satisfies*

$$
\begin{aligned}
P(S^\pi(nt)/n \leq s) \longrightarrow & \\
& 1 - \int_0^1 {}^t p(x,0) \exp\left\{ - \int_0^{\min\{t, v^*(s)\}} {}^t p(x,t^*)dt^* \right\} dx \\
& + \mathbf{I}\{t < v^*(s)\} \int_{x \notin A^\pi_{x(s)}} {}^t p(x,0) \exp\left\{ - \int_0^t {}^t p(x,t^*)dt^* \right\} dx
\end{aligned}
$$

as $n \to \infty$ where $v^*(s)$ and $A^\pi_{x(s)}$ are defined by (4.16) and (4.17) respectively.

The difference between just considering the long-term behaviour and taking into account also the cold start effects is perhaps most easily seen in comparing Figure 4.2.1 and Figure 4.4.1. In the former figure, the backward-time may go on forever while in the latter, there is a limit which the backward-time cannot exceed. Still letting 'now' define backward-time 0, this limit is given by the time the move-to-front process has been running. Hence, t is as far back in (rescaled) time that we can possibly go.

Now, introduce the random variables ${}^{nt}L_i$, $i = 1, \ldots, n$, denoting the rescaled time elapsed since Controller's last request for record i. Then, ${}^{nt}L_i$ is distributed as $\min\{{}^{nt}T_i, t\}$, where ${}^{nt}T_i$, $i = 1, \ldots, n$, was defined in Section 4.2.

The proof of Theorem 4.4.1 is a direct generalization of the one for Theorem 4.3.1. Hence, first of all we need a generalized version of Lemma 4.3.3 including the transient behaviour.

With notation analogous to the one used before, $S^\pi_k(nt)$ is Second User's cost of searching record k at genuine time nt. Given that Second User requests record $\lceil nx \rceil$ at rescaled time t and that rescaled time v^* has elapsed since Controller last requested this record, taking into account that there is

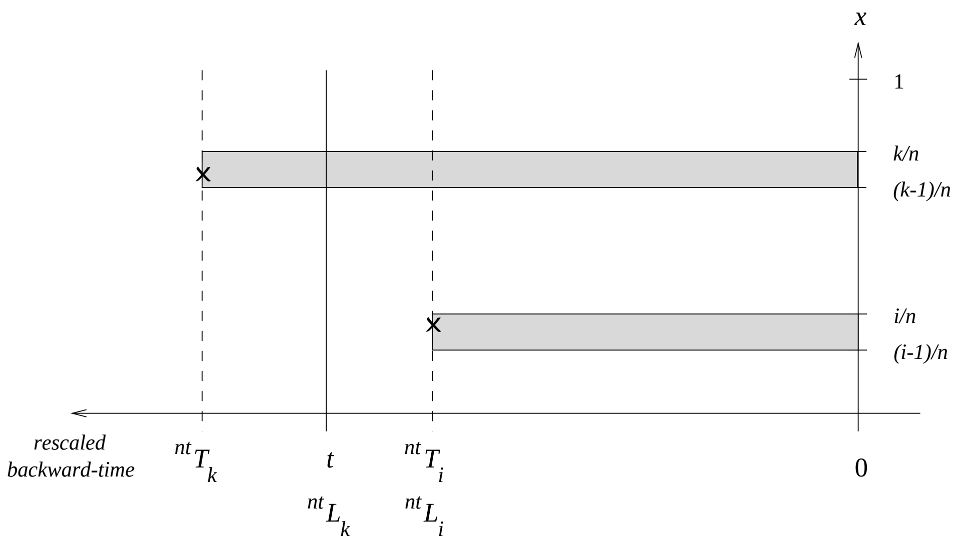


Figure 4.4.1: *Embedding in two-dimensional Poisson process.* At (forward) time nt ('now', corresponding to /rescaled/ backward-time $t^* = 0$) Second User makes a request. Controller requests according to a two-dimensional Poisson process with intensity ${}^{t}p(x, t^*)$. In terms of the embedding, ${}^{nt}T_k$ is the waiting time until the first occurrence (in rescaled backward-time) in the shaded slice. ${}^{nt}L_i$ is the 'actual' (rescaled) time elapsed since i was last requested; if Controller has not requested this record since the clock was started (genuine) time nt ago, ${}^{nt}L_i$ is set to t, otherwise, ${}^{nt}L_i$ coincides with ${}^{nt}T_i$.

now a limit to the backward-time, the law of large numbers gives the new lemma: Conditional on ${}^{nt}L_{\lceil nx\rceil} = v^*$, $S^{\pi}_{\lceil nx\rceil}(nt)/n$ converges in probability;

Lemma 4.4.3 *Assume that the regularity conditions (4.5) and (4.6) hold and that the clock has been running for time nt. Then, for every $x \in (0,1]$ and $v^* \in (0,\infty)$,*
if $0 < v^* < t$,

$$\mathcal{L}\left(\frac{S^{\pi}_{\lceil nx\rceil}(nt)}{n} - \left[1 - \int_0^1 \exp\left\{-\int_0^{v^*} {}^{t}p(y,t^*)dt^*\right\}dy\right] \,\Bigg|\, {}^{nt}L_{\lceil nx\rceil} = v^*\right) \longrightarrow \delta_0$$

and if $v^* = t$,

$$\mathcal{L}\left(\frac{S^{\pi}_{\lceil nx\rceil}(nt)}{n} - \left[1 - \int_{y\in Ax} \exp\left\{-\int_0^{t} {}^{t}p(y,t^*)dt^*\right\}dy\right] \,\Bigg|\, {}^{nt}L_{\lceil nx\rceil} = t\right) \longrightarrow \delta_0$$

as $n \to \infty$, where δ_0 denotes a one-point distribution in 0.

Proof. Let $\mathbf{1}_{i,j}(0) = 1$ if record i precedes record j in the initial list permutation, otherwise $\mathbf{1}_{i,j}(0) = 0$. Introduce indicators similar to those in the proof of Lemma 4.3.3: $I_i = 1$ if $\lceil nx\rceil$ precedes i at the time of the current request. Conditional on ${}^{nt}L_{\lceil nx\rceil} = v^*$, the event that Controller has /not/ requested record i splits into two cases:

If $0 < v^* < t$, Controller has requested $\lceil nx\rceil$ at least once since the start. Therefore, $\lceil nx\rceil$ is currently preceding i if and only if Controller has *not* requested i since the last request for $\lceil nx\rceil$, that is, in the (rescaled) backward time interval $(0, v^*)$. By (4.3),

$$P(I_i = 1) = \exp\left\{-\int_0^{v^*} {}^{nt}p_i(t^*,n)dt^*\right\}.$$

As usual, the search cost is given by the number of records preceding the requested one $\lceil nx\rceil$. The present situation is identical to that in the proof of Lemma 4.3.3 and hence

$$E\left[S^{\pi}_{\lceil nx\rceil}(nt)/n \,\Big|\, {}^{nt}L_{\lceil nx\rceil} = v^*\right] \longrightarrow 1 - \int_0^1 \exp\left\{-\int_0^{v^*} {}^{t}p(y,t^*)dt^*\right\}dy$$

as $n \to \infty$ while

$$\mathrm{Var}\left[S^{\pi}_{\lceil nx\rceil}(nt)/n \;\middle|\; {}^{nt}L_{\lceil nx\rceil} = v^*\right] \longrightarrow 0$$

and the result in the statement follows.

If $v^* = t$, Controller has not requested $\lceil nx\rceil$ since the start and $\lceil nx\rceil$ precedes i if this was the case already at the start and Controller has not requested i.

$$P(I_i = 1) = \mathbf{1}_{\lceil nx\rceil, i}(0) \exp\left\{ - \int_0^t {}^{nt}p_i(t^*, n)dt^*\right\}.$$

Hence, we get

$$E\left[S^{\pi}_{\lceil nx\rceil}(nt)/n \;\middle|\; {}^{nt}L_{\lceil nx\rceil} = t\right] = E\left[\frac{1}{n}\sum_{\substack{i=1\\ i\neq \lceil nx\rceil}}^{n}(1-I_i)\right]$$

$$= \frac{n-1}{n} - \int_0^1 \mathbf{I}\left\{y \notin \left(\frac{\lceil nx\rceil - 1}{n}, \frac{\lceil nx\rceil}{n}\right]\right\}$$

$$\times \mathbf{1}_{\lceil nx\rceil, \lceil ny\rceil}(0) \exp\left\{ - \int_0^t {}^{nt}p_{\lceil ny\rceil}(t^*, n)dt^*\right\}dy$$

$$\longrightarrow 1 - \int_{y\in A^{\pi}_x} \exp\left\{ - \int_0^t {}^{t}p(y, t^*)dt^*\right\}dy$$

as $n \to \infty$ by the regularity condition (4.5) and by (4.14).

As for the variance,

$$\mathrm{Var}\left[S^{\pi}_{\lceil nx\rceil}(nt)/n \;\middle|\; {}^{nt}L_{\lceil nx\rceil} = t\right]$$

$$= \frac{1}{n^2}\sum_{\substack{i=1\\ i\neq \lceil nx\rceil}}^{n} \mathrm{Var}\left[I_i\right]$$

$$\begin{aligned}
&= \frac{1}{n^2} \sum_{\substack{i=1 \\ i \neq \lceil nx \rceil}}^{n} \mathbf{1}_{\lceil nx \rceil, i}(0) \exp\left\{ - \int_0^t {}^{nt}p_i(t^*, n) dt^* \right\} \\
&\qquad \times \left(1 - \mathbf{1}_{\lceil nx \rceil, i}(0) \exp\left\{ - \int_0^t {}^{nt}p_i(t^*, n) dt^* \right\} \right) \\
&\leq \frac{1}{n^2} \sum_{\substack{i=1 \\ i \neq \lceil nx \rceil}}^{n} \mathbf{1}_{\lceil nx \rceil, i}(0) \exp\left\{ - \int_0^t {}^{nt}p_i(t^*, n) dt^* \right\} \\
&\leq \frac{1}{n^2} \sum_{\substack{i=1 \\ i \neq \lceil nx \rceil}}^{n} \exp\left\{ - \int_0^t {}^{nt}p_i(t^*, n) dt^* \right\}. \qquad (4.18)
\end{aligned}$$

Apart from the upper limit of the integral, (4.18) is the expression in (4.11). Hence, by the proof of Lemma 4.3.3,

$$\mathrm{Var}[S^{\pi}_{\lceil nx \rceil}(nt)/n \mid {}^{nt}L_{\lceil nx \rceil} = t] \longrightarrow 0$$

as $n \to \infty$, and we are done.

□

Proof of Theorem 4.4.1. We modify the proof of Theorem 4.3.1. As in that proof we condition with respect to the record k requested by Second User at genuine time nt and the (genuine) time nv^* elapsed since Controller last requested k.

$$\begin{aligned}
&P(S^{\pi}(nt)/n \leq s) \\
&\quad = \sum_{k=1}^{n} {}^{nt}q_k(n) \int_0^{\infty} P(S^{\pi}_k(nt)/n \leq s | \; {}^{nt}L_k = v^*) f_{{}^{nt}L_k}(v^*) dv^* \\
&\quad = \sum_{k=1}^{n} {}^{nt}q_k(n) \int_0^{t} P(S^{\pi}_k(nt)/n \leq s | \; {}^{nt}L_k = v^*) f_{{}^{nt}L_k}(v^*) dv^* \qquad (4.19) \\
&\qquad + \sum_{k=1}^{n} {}^{nt}q_k(n) P(S^{\pi}_k(nt)/n \leq s | \; {}^{nt}L_k = t) P({}^{nt}L_k = t).
\end{aligned}$$

In (4.19), $v^* < t$ and hence the distribution of ${}^{nt}L_k$ equals that of ${}^{nt}T_k$.

$$
\begin{aligned}
&P(S_k^\pi(nt)/n \le s) \\
&\quad = \sum_{k=1}^{n} {}^{nt}q_k(n) \int_0^t P(S_k^\pi(nt)/n \le s|\ {}^{nt}L_k = v^*) f_{{}^{nt}T_k}(v^*)dv^* \\
&\qquad + \sum_{k=1}^{n} {}^{nt}q_k(n) P(S_k^\pi(nt)/n \le s|\ {}^{nt}L_k = t) P({}^{nt}T_k > t).
\end{aligned}
$$

The distribution of ${}^{nt}T_k$ is known from (4.3) – (4.4). Hence,

$$
\begin{aligned}
&P(S^\pi(nt)/n \le s) \\
&\quad = \sum_{k=1}^{n} {}^{nt}q_k(n) \int_0^t P(S_k^\pi(nt)/n \le s|\ {}^{nt}L_k = v^*) \\
&\qquad\qquad \times {}^{nt}p_k(v^*, n) \exp\Big\{ - \int_0^{v^*} {}^{nt}p_k(t^*, n)dt^* \Big\} dv^* \\
&\qquad + \sum_{k=1}^{n} {}^{nt}q_k(n) P(S_k^\pi(nt)/n \le s|\ {}^{nt}L_k = t) \exp\Big\{ - \int_0^{t} {}^{nt}p_k(t^*, n)dt^* \Big\}.
\end{aligned}
\tag{4.20}
$$

Changing the sum into an integral and substituting $x := k/n$ we get

$$
\begin{aligned}
&P(S^\pi(nt)/n \le s) \\
&\quad = \int_0^1 n\ {}^{nt}q_{\lceil nx \rceil}(n) \int_0^t P(S_{\lceil nx \rceil}^\pi(nt)/n \le s|{}^{nt}L_{\lceil nx \rceil} = v^*) \\
&\qquad\qquad \times {}^{nt}p_{\lceil nx \rceil}(v^*, n) \exp\Big\{ - \int_0^{v^*} {}^{nt}p_{\lceil nx \rceil}(t^*, n)dt^* \Big\} dv^* dx \\
&\qquad + \int_0^1 n\ {}^{nt}q_{\lceil nx \rceil}(n) P(S_{\lceil nx \rceil}^\pi(nt)/n \le s|{}^{nt}L_{\lceil nx \rceil} = t) \\
&\qquad\qquad \times \exp\Big\{ - \int_0^{t} {}^{nt}p_{\lceil nx \rceil}(t^*, n)dt^* \Big\} dx.
\end{aligned}
$$

By the law of large numbers, Lemma 4.4.3, and the regularity assumptions

(4.5) and (4.6), we have

$$
\begin{aligned}
&P(S^\pi(nt)/n \leq s) \longrightarrow \\
&\quad \int_0^1 {}^t q(x) \int_0^t \mathbf{I}\Big\{1 - \int_0^1 \exp\Big\{-\int_0^{v^*} {}^t p(y,t^*)dt^*\Big\}dy \leq s\Big\} \\
&\qquad \times {}^t p(x,v^*) \exp\Big\{-\int_0^{v^*} {}^t p(x,t^*)dt^*\Big\}dv^* dx \\
&\quad + \int_0^1 {}^t q(x)\mathbf{I}\Big\{1 - \int_{y\in A_x^\pi} \exp\Big\{-\int_0^t {}^t p(y,t^*)dt^*\Big\}dy \leq s\Big\} \\
&\qquad \times \exp\Big\{-\int_0^t {}^t p(x,t^*)dt^*\Big\}dx
\end{aligned}
$$

as $n \to \infty$.

Recognizing the similarity between the double integral term and (4.13), we turn the indicator function into an integration area as we did in the proof of Theorem 4.3.1. Pursuing a similar line of reasoning to rewrite also the indicator function in the second integral term as an integration area, we find that the left hand side of the inequality in the argument of this indicator function,

$$1 - \int_{y\in A_x^\pi} \exp\Big\{-\int_0^t {}^t p(y,t^*)dt^*\Big\}dy \tag{4.21}$$

decreases if the integration area A_x^π is increased. So the inequality is satisfied only for sufficiently large sets A_x^π. However, by definition (4.14), A_x^π cannot exceed $(0,1]$. This means that the inequality cannot always be satisfied by a suitable choice of A_x^π for all s. In fact, the breakpoint is given by

$$\int_{y\in(0,1]} \exp\Big\{-\int_0^t {}^t p(y,t^*)dt^*\Big\}dy = 1 - s. \tag{4.22}$$

For smaller values of s, the indicator necessarily equals 0. But (4.22) is identical to the equation defining $v^*(s)$,

$$\int_0^1 \exp\Big\{-\int_0^{v^*(s)} {}^t p(y,t^*)dt^*\Big\}dy = 1-s,$$

only with t as the upper integration limit instead of $v^*(s)$. This observation allows us to express a condition on t in terms of s: For each s, the indicator is 0 for all $t > v^*(s)$, while for $t \le v^*(s)$, the indicator is 1 for integration areas $A_x^\pi \supset A_{x(s)}^\pi$, where we define $A_{x(s)}^\pi$ such that

$$\int_{y\in A_{x(s)}^\pi} \exp\Big\{-\int_0^t {}^t p(y,t^*)dt^*\Big\}dy = 1-s.$$

With these definitions we have,

$$\begin{aligned} &P(S^\pi(nt)/n \le s) \longrightarrow \\ &\quad 1-\int_0^1 {}^t q(x)\exp\Big\{-\int_0^{\min\{t,v^*(s)\}} {}^t p(x,t^*)dt^*\Big\}dv^*dx \\ &\quad +\mathbf{I}\{t\le v^*(s)\}\int_{x:A_x^\pi\supset A_{x(s)}^\pi} {}^t q(x)\exp\Big\{-\int_0^t {}^t p(x,t^*)dt^*\Big\}dx \end{aligned} \tag{4.23}$$

as $n \to \infty$.

But from the definition of A_x^π, (4.14), it follows that $\{x : A_x^\pi \supset A_{x(s)}^\pi\}$ is simply the set of $x \in (0,1]$ such that $\lceil nx\rceil$ precedes $\lceil nx(s)\rceil$ in the initial permutation and the integration area $x : A_x^\pi \supset A_{x(s)}^\pi$ in (4.23) may be simplified to $x \notin A_{x(s)}^\pi$ and

$$\begin{aligned} &P(S^\pi(nt)/n \le s) \longrightarrow \\ &\quad 1-\int_0^1 {}^t q(x)\exp\Big\{-\int_0^{\min\{t,v^*(s)\}} {}^t p(x,t^*)dt^*\Big\}dv^*dx \\ &\quad +\mathbf{I}\{t\le v^*(s)\}\int_{x\notin A_{x(s)}^\pi} {}^t q(x)\exp\Big\{-\int_0^t {}^t p(x,t^*)dt^*\Big\}dx \end{aligned}$$

If $t = v^*(s)$, then $A_{x(s)}^\pi = (0,1]$ and since integrating over the empty set

$x \notin (0,1]$ yields nothing, this case may be omitted in the indicator argument, whereby we obtain the expression in the statement.

□

4.5 Stochastic ordering

Suppose that ${}^t q_1(x)$ and ${}^t q_2(x)$ are two possible densities describing Second User's request probabilities as in (4.1). Denote the corresponding distribution functions

$$ {}^tQ_1(x) = \int_0^x {}^tq_1(y)dy \quad \text{and} \quad {}^tQ_2(x) = \int_0^x {}^tq_2(y)dy. $$

Furthermore, we make some additional regularity assumptions about the intensity ${}^tp(x,t^*)$:
(i) For almost every $t^* \in (0, v^*(s))$, ${}^tp(x,t^*)$ is differentiable with respect to x and for almost every $x \in (0,1]$, $\frac{\partial}{\partial x}{}^tp(x,t^*)$ is integrable with respect to t^*.
(ii) There exists a function $h(\cdot)$, integrable with respect to t^*, such that

$$ \sup_x \left| \frac{\partial}{\partial x}{}^tp(x,t^*) \right| \leq |h(t^*)|. $$

Then the next theorem allows us to decide which distribution yields the stochastically better long-term asymptotic search cost.

Assuming ${}^tq_1(\cdot)$ to be Second User's request density, let S_1 be a random variable such that $S^{long\text{-}term}(nt)/n \xrightarrow{\mathcal{L}} S_1$. Then the distribution of S_1 is known by Theorem 4.3.1. Similarly, if Second User requests according to density ${}^tq_2(\cdot)$, define a random variable S_2 such that $S^{long\text{-}term}(nt)/n \xrightarrow{\mathcal{L}} S_2$.

Theorem 4.5.1 *Assume that the regularity conditions (4.5) - (4.6) hold and that ${}^tp(x,t^*)$ satisfies (i) - (ii) above. If*

$$ \int_0^1 [{}^tQ_2(x) - {}^tQ_1(x)] \exp\left\{ - \int_0^{v^*(s)} {}^tp(x,t^*)dt^* \right\} \int_0^{v^*(s)} \frac{\partial}{\partial x}{}^tp(x,t^*)dt^*dx \leq 0, \tag{4.24} $$

where $v^*(s)$ is defined by (4.8), then S_1 is stochastically greater than S_2.

Remark. The transient analogue (cf. Section 4.4) of (4.24) is

$$\int_0^1 [{}^tQ_2(x) - {}^tQ_1(x)] \exp\Big\{ - \int_0^{\min\{t,v^*(s)\}} {}^tp(x,t^*)dt^*\Big\}$$
$$\times \int_0^{\min\{t,v^*(s)\}} \frac{\partial}{\partial x} {}^tp(x,t^*)dt^*dx$$
$$+\mathbf{I}\{t < v^*(s)\} \int_{x \notin A^\pi_{x(s)}} [{}^tQ_1(x) - {}^tQ_2(x)] \exp\Big\{ - \int_0^t {}^tp(x,t^*)dt^*\Big\}$$
$$\times \int_0^t \frac{\partial}{\partial x} {}^tp(x,t^*)dt^*dx$$
$$+\mathbf{I}\{t < v^*(s)\} \Big[[{}^tQ_1(x) - {}^tQ_2(x)] \exp\Big\{ - \int_0^t {}^tp(x,t^*)dt^*\Big\}\Big]_{x \notin A^\pi_{x(s)}} \le 0,$$

where $v^*(s)$ and $A^\pi_{x(s)}$ are defined by (4.16) resp. (4.17). The dependence on the initial permutation complicates the interpretation and we shall not treat this expression further. Note however that it tends to (4.24) as t tends to infinity.

Proof. By Theorem 4.3.1 and integration by parts,

$$P(S_1 \le s) - P(S_2 \le s)$$
$$= \int_0^1 [{}^tq_2(x) - {}^tq_1(x)] \exp\Big\{ - \int_0^{v^*(s)} {}^tp(x,t^*)dt^*\Big\}dx$$
$$= \Big[[{}^tQ_2(x) - {}^tQ_1(x)] \exp\Big\{ - \int_0^{v^*(s)} {}^tp(x,t^*)dt^*\Big\}\Big]_{x=0}^1$$
$$+ \int_0^1 [{}^tQ_2(x) - {}^tQ_1(x)] \exp\Big\{ - \int_0^{v^*(s)} {}^tp(x,t^*)dt^*\Big\} \frac{\partial}{\partial x} \int_0^{v^*(s)} {}^tp(x,t^*)dt^*dx$$
$$= \int_0^1 [{}^tQ_2(x) - {}^tQ_1(x)] \exp\Big\{ - \int_0^{v^*(s)} {}^tp(x,t^*)dt^*\Big\} \int_0^{v^*(s)} \frac{\partial}{\partial x} {}^tp(x,t^*)dt^*dx$$

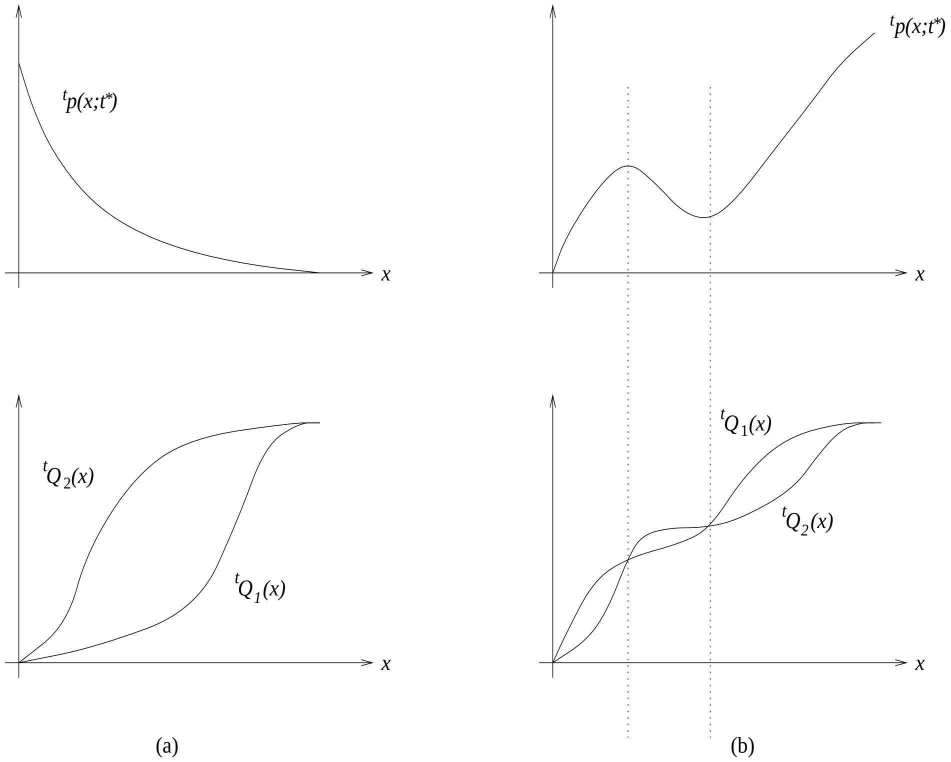


Figure 4.5.1: *Illustration of situations in which S_1 is stochastically greater than S_2.*

because the conditions (i)-(ii) allow us to change the order of integration and differentiation (cf. Appendix B) and since ${}^tQ_1(1) = {}^tQ_2(1) = 1$ and ${}^tQ_1(0) = {}^tQ_2(0) = 0$.

□

In particular we have (cf. Figure 4.5.1 (b))

Corollary 4.5.2 *Assume the same conditions as in Theorem 4.5.1. If for each $x \in (0,1]$, $[{}^tQ_2(x) - {}^tQ_1(x)]$ and $\frac{\partial}{\partial x}{}^tp(x,t^*)$ are of opposite sign for all $t^* \in (0,\infty)$, S_1 is stochastically greater than S_2.*

Proof. Since

$$0 \le \exp\Big\{ - \int_0^{v^*(s)} {}^tp(x,t^*)dt^* \Big\} \le 1,$$

(4.24) is always satisfied.

□

Earlier, in Section 3.5, when we considered only time-independent request probabilities we chose indices so that Second User's request probabilities were decreasing for increasing labels. If, analogously,

$$1 > p_1 \ge p_2 \ge \cdots \ge p_n > 0 \tag{4.25}$$

always (at any time) holds for Controller's request probabilities, we have the following (cf. Figure 4.5.1 (a)):

Corollary 4.5.3 *Suppose that for all $t^* \in (0,\infty)$, ${}^tp(x,t^*)$ is decreasing in x. If ${}^tQ_1(x) \le {}^tQ_2(x)$ for all $x \in (0,1]$, then S_1 is stochastically greater than S_2.*

In other words, if R_1 (representing Second User's request) and R_2 are random variables with distribution functions ${}^tQ_1(\cdot)$ and ${}^tQ_2(\cdot)$ respectively, then if R_1 is stochastically greater than R_2 then S_1 is stochastically greater than S_2 (and hence S_2 is the 'better' search cost).

Proof. If for all t^*, ${}^tp(x,t^*)$ is decreasing in x then $\frac{\partial}{\partial x}{}^tp(x,t^*)$ is negative. Hence, since ${}^tQ_2(x) \ge {}^tQ_1(x)$, $[{}^tQ_2(x) - {}^tQ_1(x)]\frac{\partial}{\partial x}{}^tp(x,t^*)$ is negative for all x and t^*. Consequently, Theorem 4.5.1 holds.

□

Remark. In particular, these results of course holds if Controller's request probabilities are indeed time-independent.

Remark. Intuitively, these results tell us that if Second User frequently wants records that Controller seldom uses, the search cost will be greater than if Second User prefers records that are popular also with Controller. If Second User always requested only Controller's currently most popular record, the minimal search cost would be attained.

4.6 Examples

Earlier in our discussion of the move-to-front rule we let certain (time-independent) request probability distributions serve as examples (cf. Section 3.5). We now claim that of these, Theorem 4.3.1 (and Theorem 4.4.1) applies to the uniform distribution, generalized Zipf's law with $0 < \alpha < 1$ and power law. First, we consider the uniform distribution. Next, we turn to Zipf's and power law. Introducing slightly modified versions of these distributions, we give (explicitly) the limiting distribution of the cost fraction as the number of records tends to infinity. Finally, we consider the limiting distribution function when Controller's preferences change once in time.

Uniform distribution

As far as the uniform distribution is concerned, we immediately have that as soon as at least one of Controller and Second User requests with uniform probabilities, the limiting distribution of the asymptotic long-term search cost fraction is again uniform on the unit interval since $S^{long\text{-}term}(nt)/n$ takes values in $\{0, 1/n, 2/n, \ldots, (n-1)/n\}$, each with probability $1/n$. This result follows when we take ${}^tp(x, t^*) \equiv 1$ or ${}^tq(x) \equiv 1$ in Theorem 4.3.1 and in this case the regularity condition (4.1) is obviously satisfied since $n\frac{1}{n} = n\int_{(k-1)/n}^{k/n} 1dy = 1$.

In fact, for uniform Second User request probabilities the asymptotic cost fraction is immediately uniform regardless of Controller's requests (cf. Section 3.5.2) and hence the distribution of $S^{\pi}(nt)/n$ converges to the uniform on the unit interval as we take ${}^tq(x) \equiv 1$ in Theorem 4.4.1. When Controller has uniform request probabilities, the initial list permutation at first affects the search cost until stationarity has been reached. Hence, the limiting distribution of $S^{\pi}(nt)/n$ in this case is not necessarily uniform.

Generalized Zipf's law with $0 < \alpha < 1$ and power law

The generalized Zipf's law with $\alpha > 1$ and the geometric distribution belong to the category of distributions with summable weights (cf. Section 3.5.3). In those cases the preferences for the records are highly unbalanced and the regularity condition (4.6) is not satisfied. As the number of records n becomes large, some records are very popular while most hardly ever get requested. Hence, the limiting distribution for the search cost fraction is degenerate and the result of Theorem 4.3.1 does not apply.

In Proposition 4.6.1 we give the long-term behaviour. If one wishes to include cold-start effects, the derivation is modified straightforwardly — just use the general limiting distribution (4.15) instead of the long-term (4.7).

Before we can apply Theorem 4.3.1, the regularity condition (4.6) must be satisfied. In the case of generalized Zipf's law, $0 < \alpha < 1$, the request probability for record $\lceil nx \rceil$ is given by

$$p_{\lceil nx \rceil} = \frac{\lceil nx \rceil^{-\alpha}}{H_n(\alpha)}.$$

Integral comparison (3.10) - (3.9) gives

$$\frac{n^{1-\alpha}-1}{1-\alpha} \leq H_n(\alpha) \leq \frac{n^{1-\alpha}}{1-\alpha}$$

and so

$$\frac{1-\alpha}{n^{1-\alpha}} n \lceil nx \rceil^{-\alpha} \leq np_{\lceil nx \rceil} \leq \frac{1-\alpha}{n^{1-\alpha}-1} n \lceil nx \rceil^{-\alpha}$$

where both the right-hand and the left-hand side $\longrightarrow (1-\alpha)x^{-\alpha}$ as $n \to \infty$. Hence,

$$np_{\lceil nx \rceil} \longrightarrow (1-\alpha)x^{-\alpha}.$$

Here, $np_{\lceil nx \rceil}$ is not exactly equal to $n \int_{(\lceil nx \rceil -1)/n}^{\lceil nx \rceil /n} (1-\alpha) y^{-\alpha} dy$, cf. (4.1) and (4.6). However, since we are only interested in asymptotics we are content that they are almost equal for large n:

$$\begin{aligned}
& n \int\limits_{(\lceil nx \rceil -1)/n}^{\lceil nx \rceil /n} (1-\alpha) y^{-\alpha} dy \\
&= n \left[\left(\frac{\lceil nx \rceil}{n} \right)^{1-\alpha} - \left(\frac{\lceil nx \rceil}{n} - \frac{1}{n} \right)^{1-\alpha} \right] \\
&= \frac{\left(\frac{\lceil nx \rceil}{n} \right)^{1-\alpha} - \left(\frac{\lceil nx \rceil}{n} - \frac{1}{n} \right)^{1-\alpha}}{\frac{1}{n}} \\
&\approx \left. \frac{d}{dy} y^{1-\alpha} \right|_{y=\frac{\lceil nx \rceil}{n}} \\
&= (1-\alpha) \left(\frac{\lceil nx \rceil}{n} \right)^{-\alpha} \\
&= n(1-\alpha) \frac{\lceil nx \rceil^{-\alpha}}{n^{1-\alpha}}
\end{aligned}$$

$$\approx n\frac{\lceil nx\rceil^{-\alpha}}{H_n(\alpha)} = np_{\lceil nx\rceil}$$

by definition of the derivative and since for large n, $1/H_n(\alpha) \approx (1-\alpha)/n^{1-\alpha}$.

In the light of this, we define the *modified Zipf's law*, $0 < \alpha < 1$:

$$p_k(n) := \int\limits_{(k-1)/n}^{k/n} (1-\alpha)x^{-\alpha}dx, \quad k = 1, \dots, n. \tag{4.26}$$

In case of opposite requests, it is easily shown by identical argumentation that

$$np_{\lceil nx\rceil} = n\frac{(n-\lceil nx\rceil)^{-\alpha}}{H_n(\alpha)} \longrightarrow (1-\alpha)(1-x)^{-\alpha}$$

which is fit to be the density generating the request probabilities for large n.

For power law,

$$p_{\lceil nx\rceil} = \frac{\lceil nx\rceil^{\beta}}{G_n(\beta)}.$$

Integral comparison (3.7) - (3.8) gives

$$\frac{n^{\beta+1}}{\beta+1} \leq G_n(\beta) \leq \frac{n^{\beta+1}-1}{\beta+1} + n^{\beta}$$

and we get

$$\frac{\beta+1}{n^{\beta+1}-1+(\beta+1)n^{\beta}}n\lceil nx\rceil^{\beta} \leq np_{\lceil nx\rceil} \leq \frac{\beta+1}{n^{\beta+1}}n\lceil nx\rceil^{\beta}$$

where both sides of the inequality $\longrightarrow (\beta+1)x^{\beta}$ as $n \to \infty$ and hence

$$np_{\lceil nx\rceil} \longrightarrow (\beta+1)x^{\beta}.$$

Reasoning as above, it follows that for large n, $np_{\lceil nx\rceil}$ is almost equal to $n\int_{(\lceil nx\rceil-1)/n}^{\lceil nx\rceil/n}(\beta+1)y^{\beta}dy$. Hence, we define the *modified power law*, $\beta > 0$:

$$p_k(n) := \int\limits_{(k-1)/n}^{k/n} (\beta+1)x^{\beta}dx, \quad k = 1, \dots, n. \tag{4.27}$$

If we have opposite requests, then

$$p_{\lceil nx\rceil} = \frac{(n-\lceil nx\rceil)^{\beta}}{G_n(\beta)}$$

and the same method is used to find that

$$np_{\lceil nx \rceil} \longrightarrow (\beta+1)(1-x)^{\beta},$$

which we take to be the density in this case in analogy with the reasoning above.

Proposition 4.6.1 *If Controller requests according to the modified Zipf's law (defined in (4.26)) with parameter α_p, $0 < \alpha_p < 1$, and Second User according to either the modified Zipf's law with parameter α_q, $0 < \alpha_q < 1$, or the modified power law (defined in (4.27)) with parameter $\beta_q > 0$, then, provided that their preferences are similar,*

$$P(S^{long\text{-}term}(nt)/n \leq s) \longrightarrow$$
$$1-(1-\alpha_q)\int_0^1 x^{-\alpha_q} \exp\Big\{-v^*(s)(1-\alpha_p)x^{-\alpha_p}\Big\}dx$$

resp.

$$P(S^{long\text{-}term}(nt)/n \leq s) \longrightarrow$$
$$1-(\beta_q+1)\int_0^1 x^{\beta_q} \exp\Big\{-v^*(s)(1-\alpha_p)x^{-\alpha_p}\Big\}dx,$$

as $n \to \infty$, where $v^*(s)$ is such that

$$\int_0^1 \exp\Big\{-v^*(s)(1-\alpha_p)y^{-\alpha_p}\Big\}dy = 1-s.$$

If instead Controller requests according to the modified power law with parameter $\beta_p > 0$ while Second User requests according to either the modified Zipf's law with parameter α_q, $0 < \alpha_q < 1$, or the modified power law with parameter $\beta_q > 0$, still with similar preferences, then

$$P(S^{long\text{-}term}(nt)/n \leq s) \longrightarrow$$
$$1-(1-\alpha_q)\int_0^1 x^{-\alpha_q} \exp\Big\{-v^*(s)(\beta_p+1)x^{\beta_p}\Big\}dx$$

resp.

$$P(S^{long\text{-}term}(nt)/n \leq s) \longrightarrow$$
$$1-(\beta_q+1)\int_0^1 x^{\beta_q} \exp\Big\{-v^*(s)(\beta_p+1)x^{\beta_p}\Big\}dx,$$

as $n \to \infty$*, where* $v^*(s)$ *is such that*

$$\int_0^1 \exp\Big\{-v^*(s)(\beta_p+1)x^{\beta_p}\Big\}dx = 1-s.$$

If the preferences are opposite, replace one x *in the limit above corresponding to the right Controller and Second User distributions with* $1-x$*.*

Remark. We stress that in Proposition 4.6.1, Controller does not change his preferences in time, while Second User might have had a different 'taste' on previous requests. As we have remarked before, requests by Second User in the past do not affect the search cost of the current Second User request.

Proof. The modified Zipf's law and modified power law probability distributions were defined to make regularity condition (4.6) hold and consequently, Theorem 4.3.1 may be applied to find the limiting distributions for $S^{long\text{-}term}(nt)/n$ as $n \to \infty$.

These request distributions include no dependence on time, and substituting the distributions $p(x)$ for ${}^tp(x,t^*)$ and $q(x)$ for ${}^tq(x)$ representing Controller's and Second User's requests respectively (using the integrands in the right-hand side of (4.26), (4.27) or their opposite-preferences analogues depending on which distribution is currently in question), the results follow immediately from

$$P(S^{long\text{-}term}(nt)/n \leq s) \longrightarrow 1-\int_0^1 q(x)e^{-v^*(s)p(x)}dx$$

as $n \to \infty$ where $v^*(s)$ is such that

$$\int_0^1 e^{-v^*(s)p(x)}dx = 1-s.$$

□

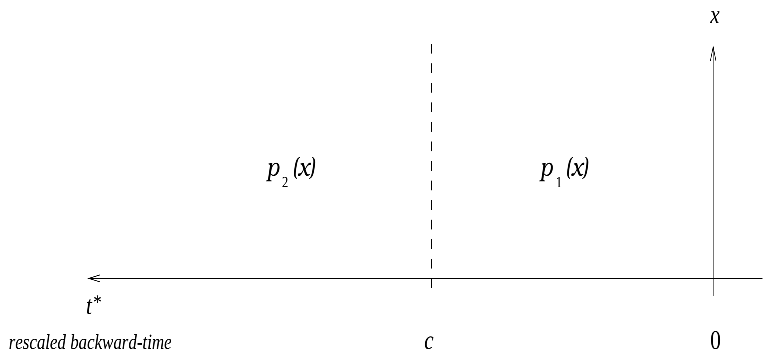


Figure 4.6.1: *Time-dependent Controller requests.* When $t^* \in [0,c)$, Controller requests according to a Poisson process with intensity $p_1(x)$. When $t^* \in [c,\infty)$ the intensity is $p_2(x)$.

Time-dependence — a simple example

When the time dependence is the simplest imaginable, Controller's taste only changes once. Looking back in time starting now (at time nt), Controller's request density changed at rescaled backward-time c (genuine time nc ago) from $p_1(x)$ to $p_2(x)$, cf. Figure 4.6.1. By Theorem 4.3.1, the distribution of Second User's search cost fraction satisfies

$$
\begin{aligned}
&P(S^{\textit{long-term}}(nt)/n \le s) \\
&\quad \longrightarrow \quad 1 - \int_0^1 q(x) \exp\Big\{ - \Big[\int_0^{v^*(s)} (p_1(x)\mathbf{I}\{t^* \in [0,c)\} \\
&\qquad\qquad\qquad\qquad + p_2(x)\mathbf{I}\{t^* \in [c,\infty)\})dt^* \Big]\Big\} dx \\
&\quad = \quad 1 - \int_0^1 q(x) \exp\Big\{ - [\min\{c, v^*(s)\} p_1(x) \\
&\qquad\qquad\qquad + (v^*(s) - \min\{c, v^*(s)\}) p_2(x)]\Big\} dx
\end{aligned}
$$

as $n \to \infty$ where $v^*(s)$ is such that

$$\int_0^1 \exp\left\{ -\left[\min\{c, v^*(s)\}p_1(x) + (v^*(s) - \min\{c, v^*(s)\})p_2(x)\right]\right\} dx = 1 - s.$$

Let for instance Second User request according to the modified Zipf's law (4.26) with $\alpha = 0.5$ and let Controller request according to the modified power law (4.27) with $\beta = 1$. If Controller's preferences are similar to those of Second User from 'now' until rescaled backward-time c when he reverses his preferences, that is, if

$$\begin{aligned} q(x) &= (1-\alpha)x^{-\alpha} = 1/2\sqrt{x} \\ p_1(x) &= (\beta+1)(1-x)^\beta = 2(1-x) \\ p_2(x) &= (\beta+1)x^\beta = 2x, \end{aligned}$$

then we have the limiting distribution function in Figure 4.6.2.

If the preferences are opposite until rescaled backward-time c (with the same underlying request probability distributions) when they are reversed into similar preferences, that is, if

$$\begin{aligned} q(x) &= 1/2\sqrt{x} \\ p_1(x) &= 2x \\ p_2(x) &= 2(1-x), \end{aligned}$$

then we have the limiting distribution function in Figure 4.6.3.

We may also take the opportunity to illustrate Theorem 3.2.1. The asymptote corresponding to modified Zipf's law ($\alpha = 0.5$) Second User requests and modified power law ($\beta = 1$) Controller requests with similar preferences is clearly above the asymptote corresponding to the same distributions but with opposite preferences, cf. Figure 4.6.4. Hence, the search cost fraction corresponding to the former distribution with similar preferences is stochastically less than the search cost fraction of the latter one with opposite preferences.

These asymptotes are intuitively natural. If c is very small then $p_1(\cdot)$ has not yet had time to induce any significant rearrangements of the records and hence $p_2(\cdot)$ will prevail. If on the other hand c is large, then $p_1(\cdot)$ has governed so long that almost all records have been requested since the change at time cn and hence the effects of the $p_2(\cdot)$ rule will have faded away.

When $p_1(\cdot)$ begins to take over, the first records to be rearranged according to the new conditions (specified by $p_1(\cdot)$) are the most popular ones. Hence, the items belonging to the beginning of the list will be the first to find their new 'correct' locations while less frequently requested records need more time to get requested and therefore need longer to adjust.

As an effect, as c increases, the limiting distribution function for the search cost fraction will converge 'from below with respect to s'. The probability that the search cost fraction is less than some small s is determined by what records are currently at the beginning of the list (in its turn determined by Controller's requests) and by the argument above this rapidly agrees with the case of Controller's requests having been indefinitely governed by the density $p_1(\cdot)$. The greater the value of s, the longer $p_1(\cdot)$ needs to have been in control for the necessary agreement between the current arrangement of the records in the list and what it would have looked like had Controller always had the same taste.

Conversely, as c decreases, the impopular records at the rear of the list are released and arranges themselves according to $p_2(\cdot)$ as they used to before Controller changed his preferences.

We have in fact already encountered this phenomenon as we studied the effect of the initial permutation in Theorem 4.4.1. Although we do not know the actual list arrangement at the switching-time (rescaled backward-time c), we know from Theorem 4.4.1 that it influences the cost fraction distribution function at the point s for another $v^*(s)$ rescaled units of time, that is, we can consider the current distribution as the one at (rescaled) time c in a list governed by the Controller request distribution $p_1(\cdot)$ that was started in some permutation given by the previously ruling Controller request distribution $p_2(\cdot)$. More concretely, in our current example(s), we have that given a certain value of c, the limiting distribution function will have 'forgotten' the list permutation at the switching-time for all s such that $c \geq v^*(s)$. Here, $v^*(s)$ is given by (cf. (4.16))

$$\int_0^1 e^{-p_1(x)v^*(s)}dx = 1 - s. \tag{4.28}$$

In the example in Figure 4.6.2 (and Figure 4.6.5), $p_1(x) = 2(1-x)$ and (4.28) reduces to

$$\frac{1 - e^{-2v^*(s)}}{2v^*(s)} = 1 - s.$$

The same holds for $p_1(x) = 2x$ in Figure 4.6.3. Taking $v^*(s) := c$, we have the s-values for which the limiting distribution has converged to the asymptote answering to the case where $p_1(\cdot)$ has always been ruling:

$$s : s \leq 1 - \frac{1 - e^{-2c}}{e^{2c}}.$$

In particular, for $c = 0.01,\ 0.1,\ 0.5,\ 1,\ 5,\ 10,\ 30$ the breakpoints are $s = 0.00993,\ 0.00937,\ 0.368,\ 0.568,\ 0.900,\ 0.950,\ 0.983$.

Remark. In Fill [22], the Laplace transform is used to show that for generalized Zipf's law $(0 < \alpha < 1)$ and power law $(\beta > 0)$ request probabilities, the search cost fraction $S^{long\text{-}term}(nt)/n$ for a single user converges in distribution. Some properties of the limiting random variable are given in an appendix, for instance, for power law request probabilities, there is a rather complicated density. Let us show that this follows from our Theorem 4.3.1.

Differentiating the limiting distribution function in Theorem 4.3.1 once with respect to s, we get the density $f(s)$:

$$f(s) = \frac{dv^*(s)}{ds} \int_0^1 {}^t q(x) {}^t p(x, v^*(s)) \exp\Big\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \Big\} dx \tag{4.29}$$

since $v^*(s)$ does not depend on the integration variable x. As before, $v^*(s)$ is such that

$$\int_0^1 \exp\Big\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \Big\} dx = 1 - s. \tag{4.30}$$

Further, differentiating (4.30) with respect to s we have

$$\frac{dv^*(s)}{ds} \int_0^1 {}^t p(x, v^*(s)) \exp\Big\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \Big\} dx = 1,$$

i.e.,

$$\frac{dv^*(s)}{ds} = \left(\int_0^1 {}^t p(x, v^*(s)) \exp\Big\{ - \int_0^{v^*(s)} {}^t p(x, t^*) dt^* \Big\} dx \right)^{-1}. \tag{4.31}$$

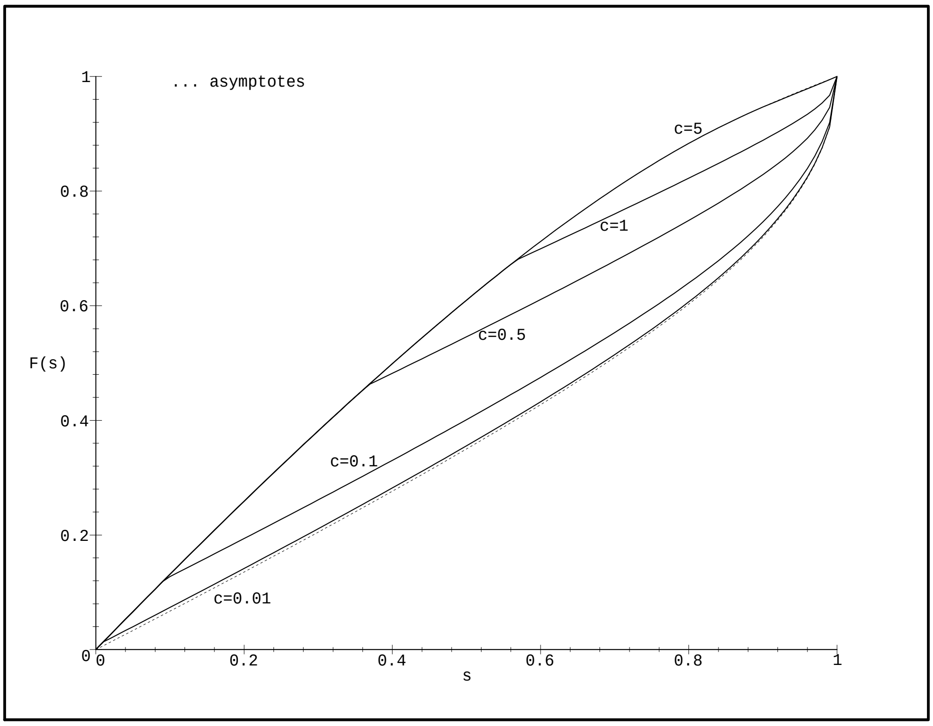


Figure 4.6.2: *Limiting distribution function,* $P(S^{long\text{-}term}(nt)/n \leq s) \to F(s)$ *as* $n \to \infty$*.* Similar preferences until rescaled backward-time c, then opposite; $q(x) = 1/2\sqrt{x}$, $p_1(x) = 2(1-x)$, $p_2(x) = 2x$.

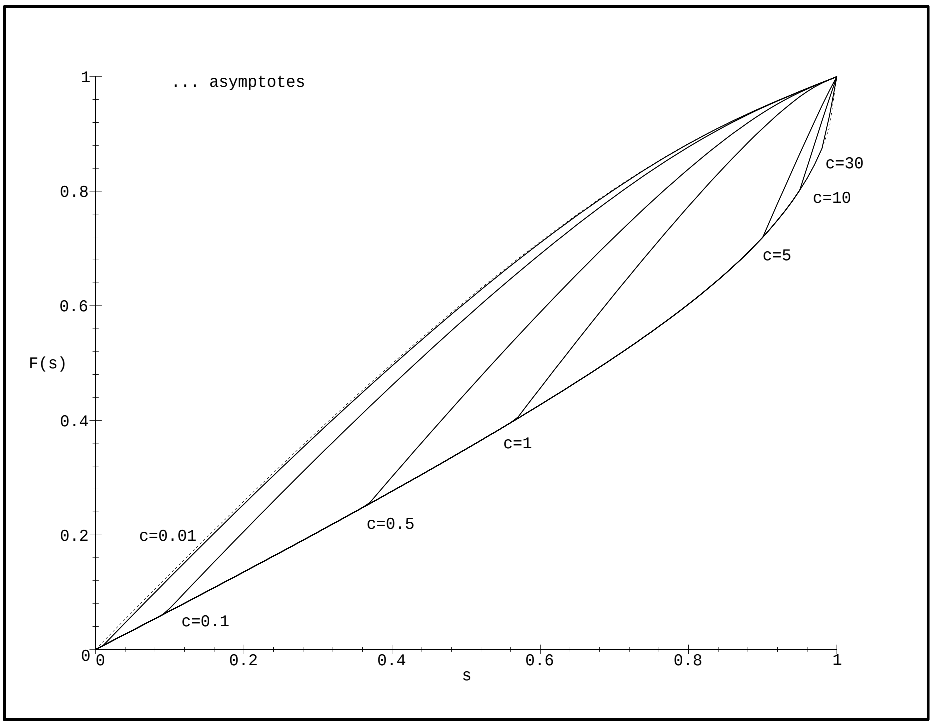


Figure 4.6.3: *Limiting distribution function,* $P(S^{long\text{-}term}(nt)/n \leq s) \to F(s)$ *as* $n \to \infty$*.* Opposite preferences until rescaled backward-time c, then similar; $q(x) = 1/2\sqrt{x}$, $p_1(x) = 2x$, $p_2(x) = 2(1-x)$.

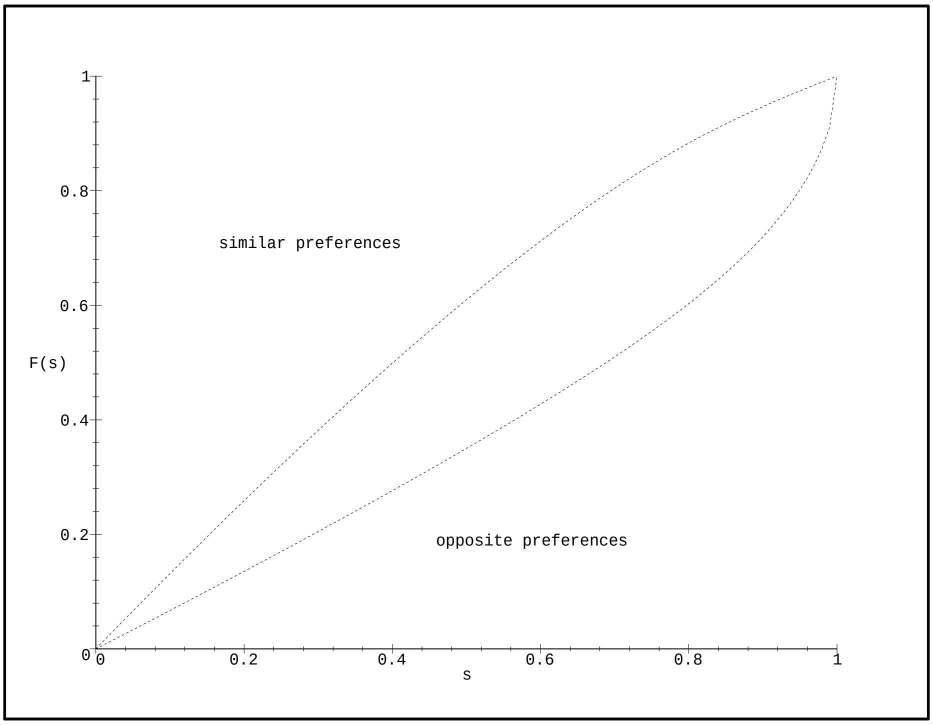


Figure 4.6.4: *Asymptotes,* $P(S^{long\text{-}term}(nt)/n \leq s) \to F(s)$ *as* $n \to \infty$.
Time-independent request probability densities.
Similar preferences: $q(x) = 1/2\sqrt{x}$, $p(x) = 2(1-x)$.
Opposite preferences: $q(x) = 1/2\sqrt{x}$, $p(x) = 2x$.

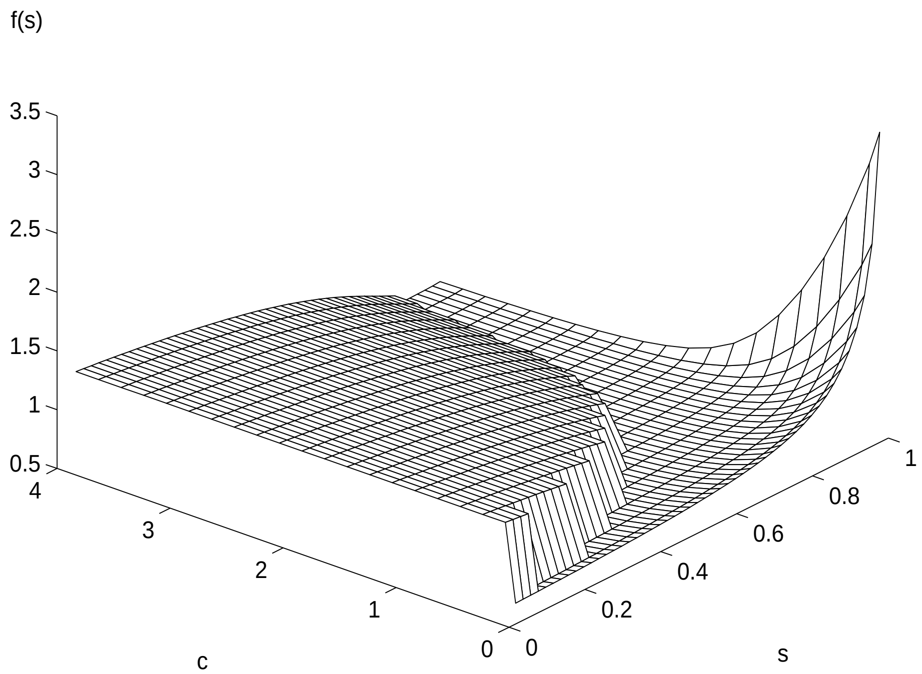


Figure 4.6.5: *Density $f(s)$ for the asymptotic search cost fraction for some c-values.* Similar preferences until rescaled backward-time c, then opposite; $q(x) = 1/2\sqrt{x}$, $p_1(x) = 2(1-x)$, $p_2(x) = 2x$.

Together, (4.29) and (4.31) yield

$$f(s) = \frac{\int_0^1 {}^tq(x)\,{}^tp(x, v^*(s)) \exp\left\{ - \int_0^{v^*(s)} {}^tp(x, t^*)dt^* \right\} dx}{\int_0^1 {}^tp(x, v^*(s)) \exp\left\{ - \int_0^{v^*(s)} {}^tp(x, t^*)dt^* \right\} dx}, \tag{4.32}$$

where $v^*(s)$ is still given by (4.30). This is the general long-term limiting density for the search cost fraction as the number of records tends to infinity.

In Figure 4.6.5, (4.32) has been used to obtain the density for the example above where Second User requests according to the modified Zipf's law, $\alpha = 0.5$, while Controller requests according to the modified power law, $\beta = 1$, with preferences similar to Second User's until (rescaled) backward-time c where they switch to opposite.

Now, if in particular Controller and Second User both request according to the same power law distribution, we should recapture Fill's result. Hence, let ${}^tq(x) := {}^tp(x, t^*) := (1 + \beta)x^\beta$. Then

$$f(s) = (\beta + 1)\frac{\int_0^1 x^{2\beta} \exp\{ - (\beta + 1)v^*(s)x^\beta\}dx}{\int_0^1 x^\beta \exp\{ - (\beta + 1)v^*(s)x^\beta\}dx}, \tag{4.33}$$

where $v^*(s)$ is such that

$$\int_0^1 \exp\{ - (\beta + 1)v^*(s)x^\beta\}dx = 1 - s. \tag{4.34}$$

We compare these expressions with Fill [22, equation (A.6)][5]:

$$f(b) = (1 + a)\frac{\int_0^1 \exp\{-u(b)r\}r^{1/a+1}dr}{\int_0^1 \exp\{-u(b)r\}r^{1/a}dr}, \tag{4.35}$$

where $u(b)$ is such that

$$b = 1 - \frac{1}{a}\int_0^1 \exp\{-u(b)r\}r^{1/a-1}dr. \tag{4.36}$$

[5]To avoid confusion some of the symbols used in Fill [22] have been changed.

Letting $r := x^\beta$ and $a := \beta$, (4.35) and (4.36) become

$$f(b) = (1+\beta)\frac{\int_0^1 x^{2\beta}\exp\{-u(b)x^\beta\}dx}{\int_0^1 x^{\beta}\exp\{-u(b)x^\beta\}dx}$$

resp.

$$b = 1 - \int_0^1 \exp\{-u(b)x^\beta\}dx,$$

which is identical to (4.33) resp. (4.34) if we let $u(b) := (\beta+1)v^*(s)$ and $b := s$.

Similarly, we may obtain the density for the limiting search cost fraction when Controller and Second User both request according to the same modified Zipf's law distribution, $0 < \alpha < 1$, ${}^tq(x) := {}^tp(x,t^*) := (1-\alpha)x^{-\alpha}$. Substituting these request densities into our general limiting search cost fraction density (4.32), we have

$$f(s) = (1-\alpha)\ \frac{\int_0^1 x^{-2\alpha}\exp\{-(1-\alpha)v^*(s)x^{-\alpha}\}dx}{\int_0^1 x^{-\alpha}\exp\{-(1-\alpha)v^*(s)x^{-\alpha}\}dx}, \tag{4.37}$$

where $v^*(s)$ is such that

$$\int_0^1 \exp\{-(1-\alpha)v^*(s)x^{-\alpha}\}dx = 1-s. \tag{4.38}$$

The explicit density (4.37)-(4.38) is not given in the appendix of Fill [22], however some asymptotics are. We show that these results too follow from Theorem 4.3.1, in this case via (4.37) and (4.38).

As $s \to 0$ in (4.38), so does $v^*(s)$. Hence, as $s \to 0$,

$$f(s) \longrightarrow (1-\alpha)\ \frac{\int_0^1 x^{-2\alpha}dx}{\int_0^1 x^{-\alpha}dx} = \frac{(1-\alpha)^2}{1-2\alpha}\left(\frac{1-\lim_{x\to 0} x^{1-2\alpha}}{1-\lim_{x\to 0} x^{1-\alpha}}\right).$$

Since when $y \to 0$, $y^{1-\gamma} \longrightarrow 0$ for $\gamma < 1$ and $y^{1-\gamma} \longrightarrow \infty$ for $\gamma > 1$, we have the following two cases:

$$f(s) \longrightarrow \begin{cases} (1-\alpha)^2/(1-2\alpha) & \text{if } 0 < \alpha < 1/2, \\ \infty & \text{if } 1/2 < \alpha < 1. \end{cases}$$

If $\alpha = 1/2$, we have

$$f(s) \longrightarrow \frac{1}{2}\frac{\int_0^1 x^{-1}dx}{\int_0^1 x^{-1/2}dx} = -\frac{1}{4}\lim_{x\to 0} \ln x = \infty.$$

As $s \to 1$ in (4.38), $v^*(s) \to \infty$.

From (4.37), substituting $y := (1-\alpha)v^*(s)x^{-\alpha}$, we have

$$\frac{f(s)}{1-\alpha} = \frac{1}{(1-\alpha)v^*(s)}\left[\frac{\int_{(1-\alpha)v^*(s)}^{\infty} y^{1-1/\alpha}e^{-y}dy}{\int_{(1-\alpha)v^*(s)}^{\infty} y^{-1/\alpha}e^{-y}dy}\right].$$

Integration by parts yields

$$\begin{aligned}
& \frac{1}{(1-\alpha)v^*(s)}\left[\frac{[(1-\alpha)v^*(s)]^{1-1/\alpha}\, e^{-(1-\alpha)v^*(s)}}{\int_{(1-\alpha)v^*(s)}^{\infty} y^{-1/\alpha}e^{-y}dy}\right. \\
& \qquad \left. + \frac{(1-1/\alpha)\int_{(1-\alpha)v^*(s)}^{\infty} y^{-1/\alpha}e^{-y}dy}{\int_{(1-\alpha)v^*(s)}^{\infty} y^{-1/\alpha}e^{-y}dy}\right] \\
= \; & \frac{1}{(1-\alpha)v^*(s)}\left[\frac{[(1-\alpha)v^*(s)]^{1-1/\alpha}\, e^{-(1-\alpha)v^*(s)}}{[(1-\alpha)v^*(s)]^{-1/\alpha}\, e^{-(1-\alpha)v^*(s)}\,(1+o(1))}\right] \\
& \qquad + \frac{1-1/\alpha}{(1-\alpha)v^*(s)} \\
\longrightarrow \; & 1
\end{aligned}$$

as $v^*(s) \to \infty$, or equivalently,

$$f(s) \longrightarrow 1 - \alpha$$

as $s \to 1$, which agrees with Fill [22].

$\square$

Part II

Move-to-root for two users

I know I hung on that
windswept tree,
Swung there for nine long
nights,
...
Bound to the tree
That no man knows
Wither the roots of it run.

...

Well-being I won
And wisdom too.
I grew and took joy in my
growth;
From a word to a word
I was led to a word,
from a deed to another deed.

Havamal
The Poetic Edda

We now turn our attention to the *move-to-root* rule, differing from move-to-front in that the items are no longer kept in linear order in a list but are instead arranged in a binary search tree.

As in our previous discussion of the move-to-front rule, we introduce a Second User who gets to read but not rearrange the information stored in the tree.

We derive expectation, variance and expressions for the generating function for Second User's search cost and draw the corresponding conclusions for the single-user case. The basic model is that of (unknown) constant request probabilities and independent requests but, as we remark and exemplify in Section 6.3, the Poisson embedding provides means to handle also time-dependent requests.

Particular interest has been payed to the case of equal request probabilities. This case was conjectured to be the worst in terms of search cost. However, a numerical counter example was given by Dobrow and Fill [17] supporting the suspicion that this is in fact not so. We investigate the worst case further and find (numerically) that this occurs, roughly put, when the request probabilities for the first $(1, 2, \ldots)$ and the last $(n, n-1, \ldots)$ records are slightly greater than the ones for those in the middle, cf. Section 6.1.3. Nevertheless, these probabilities are very close to uniformly distributed and so the uniform case serves as a good approximation of the worst case.

Chapter 5

Introduction to move-to-root

The move-to-root rule is the tree analogue of move-to-front where instead of in a list, the items are kept at the nodes of a binary search tree (cf. Section 5.1.1). Despite the close relationship to move-to-front, not nearly as much is known about move-to-root. This is mainly due to the complexity of the tree structure making it much more difficult to analyze the impact of move-to-root on the search cost.

In this chapter we define the move-to-root rule for two users and give the important Observation 5.2.1 which is the key to our further studies of Second User's search cost in Chapter 6.

5.1 Binary search trees

5.1.1 Definitions and terminology

When discussing tree structures it is customary to use a certain terminology. In general, a *tree* is an acyclic connected graph, the vertices of which are called *nodes*. Any node may be taken as the *root*. The rooted tree is a directed graph and there is a unique path from the root to an arbitrary node j. Every node $i \neq j$ on this path is an *ancestor* of j, the closest ancestor being the *parent* node etc. The *subtree* with root i consists of i and all its descendants.

A *binary tree* is an ordered tree in which each node has at most two *children* — one left and one right child. A node with no children is called a *leaf* or a *terminal node*. The tree is *ordered* in the sense that a left branch descending from a node is considered different from a right branch descending from the same node, cf. Figure 5.1.1.

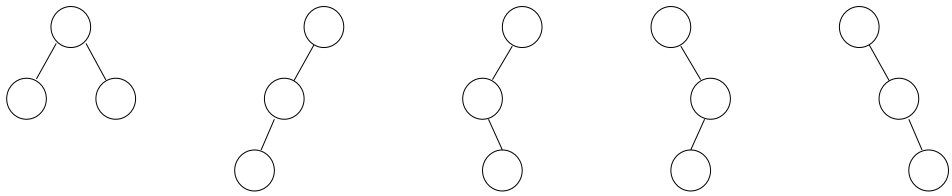


Figure 5.1.1: *Ordered binary tree.* There are five possible ordered (rooted) binary trees. If we pay no attention to order the last four trees are the same structure.

Binary trees are frequently used as information storage structures in computers. Normally, the information is divided up into *records*, each having an identifying *key* for use in searching (as in Section 2.1). A *binary search tree* is a binary tree in which all records with smaller key values are stored at nodes in the left subtree while all records with larger or equal key values are in the right subtree, and so *inorder* traversal (visit the root after visiting the left subtree but before visiting the right subtree) yields the records in ascending order of their keys. There are $C_n = \frac{1}{n+1}\binom{2n}{n}$ possible ways to construct a binary search tree with n nodes (cf. e.g. Knuth [43, p. 434]), that is, the number of binary search trees with n nodes equals the Catalan number.

5.1.2 Constructing a binary search tree

Given a sequence of records, a binary search tree is constructed applying the following algorithm to each of them (in order of their appearance in the sequence):

(i) If there is no root — insert the record as root.
(ii) If the key value is less than that of the root — insert the record in the left subtree (repeat from (i) but now consider only the left subtree).
(iii) If the key value is greater than that of the root — insert the record in the right subtree.

For an example, see Figure 5.2.2.

Note that the tree does not only depend on the number of items but also on the *order* in which they were inserted. Note also that several different permutations give rise to identical trees.

5.2 Move-to-root

The basic idea of the *move-to-root* rule is analogous to that of move-to-front (cf. Section 2.1): When a record is requested, the record is moved from its current position in the tree to be replaced at the root and the tree is then modified in order to maintain binary search tree properties.

This is the significant difference compared to move-to-front. When dealing with lists there is no internal structure that needs to be taken into consideration. In the present case however, merely moving the requested record to the root (without further restructuring) would damage the search tree. So in defining the move-to-root operation, one must specify the rearrangements of all the nodes involved.

The move-to-root process can be defined as a series of *simple exchanges* between nodes, performed until the requested item has been moved to the root of the tree, cf. Allen and Munro [1].

Let a be the requested record. Then the following algorithm defines the simple exchange, see Figure 5.2.1:

(i) If a is the root — do nothing.
(ii) If a is the left child of its parent r, modify the subtree whose root is r (the resulting tree will be the same as the original except for this subtree):
- Rotate a up to r so that a becomes the root of the subtree.
- The old left subtree of a is left unchanged in relation to a.
- The old right subtree of a becomes the left subtree of r.
- The old right subtree of r keeps its relation to r.

(iii) If a is the right child of r — perform the analogous transformation.

As an example, in Figure 1.11.1 we constructed a binary search tree consisting of eight nodes with key values $1, 2, \ldots, 8$ respectively. The same tree is now used to illustrate move-to-root in Figure 5.2.2.

From the definition of move-to-root we see that when a is requested, a is the only record which *becomes* an ancestor of any record; r on the other hand, *ceases* to be an ancestor of a and all the records contained in the subtree τ_a, that is, r ceases to be an ancestor of the requested record a and all those records with key values more 'extreme' than that of a. Hence, the only way we can make r cease to be an ancestor of a record m is to request m itself or a record between m and r. This key observation first appeared in Allen and Munro [1, Lemma 3.2].

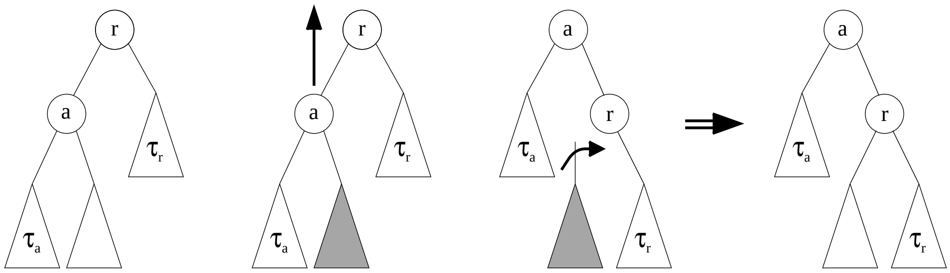


Figure 5.2.1: *Simple exchange.* Performing a simple exchange (a is the requested record) according to (ii).

Observation 5.2.1 (Allen and Munro, 1978) *Suppose that record i and j have been requested at least once each in a binary search tree modified according to move-to-root. Let $i < j$. Then i is an ancestor of j if and only if no request for $i+1, \ldots, j$ has occurred after the most recent request for i. Similarly, j is an ancestor of i if and only if no request for $i, \ldots, j-1$ has occurred after the most recent request for j.*

Above, the requested record has been moved to the root by successive simple exchanges *bottom-up.* Alternatively, we can unhook nodes and their (with respect to the requested record's key value) 'extreme' subtrees on our way down and, depending on whether the key value is less or greater than the requested key, attach them to one of two temporary trees. This *top-down* version of move-to-root (cf. Stephenson [72]) operates according to the following algorithm (a is requested, r is the root, cf. Figure 5.2.3):

(i) If a is in the left subtree, attach the root and its right subtree to the bottom left of the temporary right tree. Repeat from (i) now considering the remaining tree, that is, the left subtree containing a (i.e., τ_a in Figure 5.2.3).
(ii) If a is in the right subtree, perform the analogous operation.
(iii) If a is the root, insert the temporary trees so that the root of the temporary right tree becomes a's right child while the temporary left tree becomes a's left child.

The tree resulting from top-down move-to-root is identical to the one obtained with the bottom-up sequence of simple exchanges and Observation 5.2.1 follows. Top-down move-to-root is more efficient in the sense that each

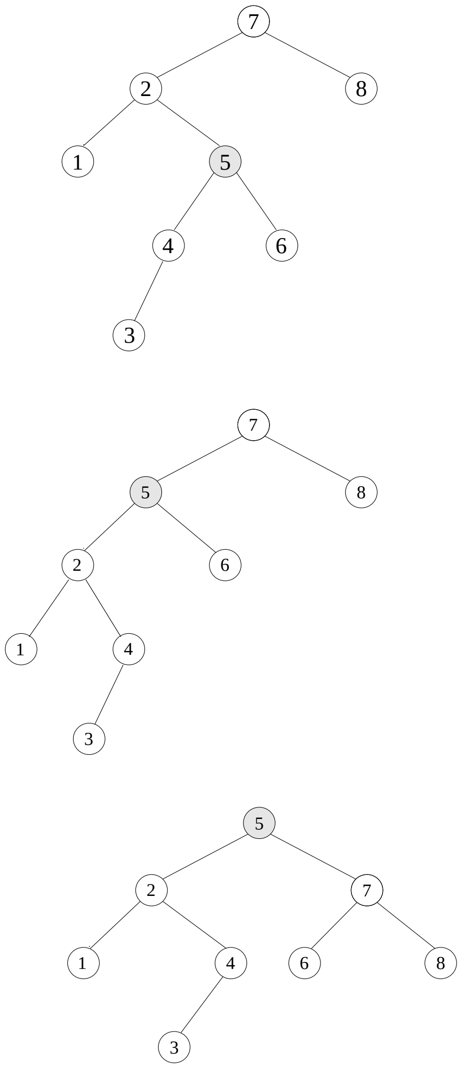


Figure 5.2.2: *Move-to-root.* Moving 5 to the root by successive simple exchanges in a tree initially constructed from the permutation 7, 8, 2, 1, 5, 6, 4, 3. Note that the same tree results from other permutations, e.g. 7, 2, 1, 5, 4, 3, 6, 8. In this example $7, 2$ and 5 are the ancestors of 4.

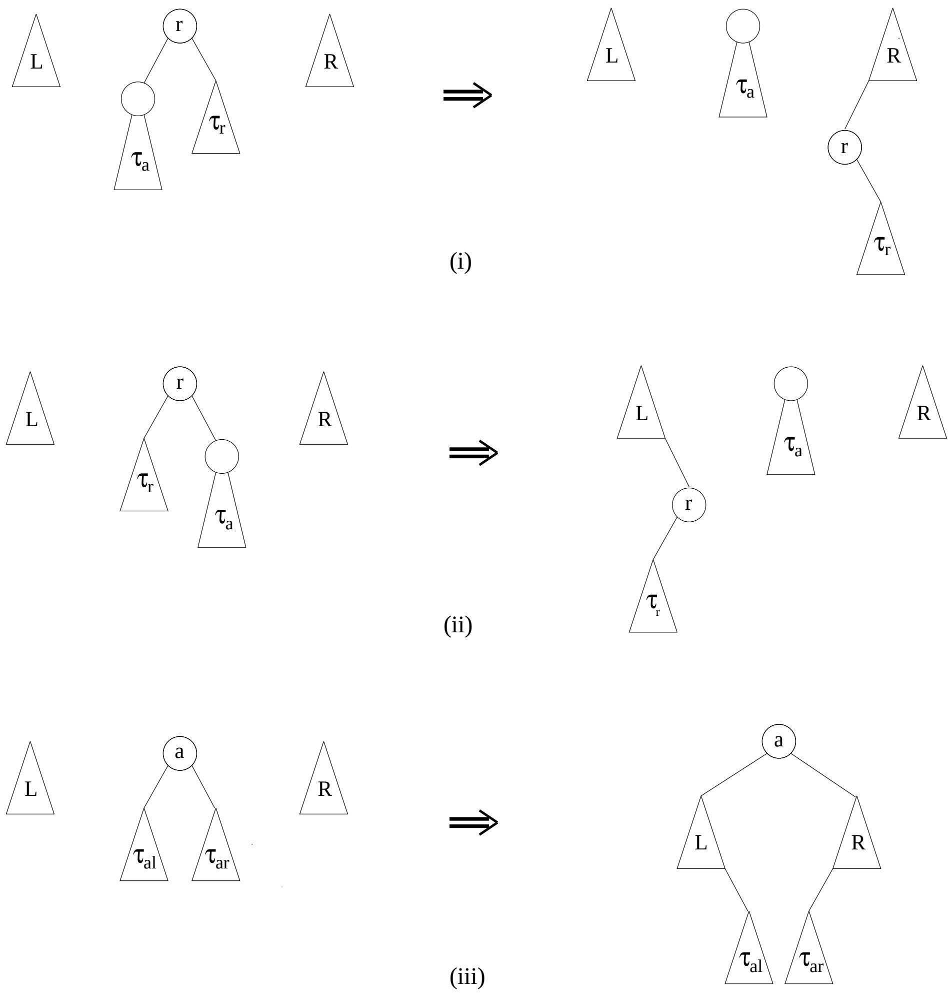


Figure 5.2.3: *Top-down move-to-root.* In (i) and (ii), the requested record a is somewhere in τ_a.

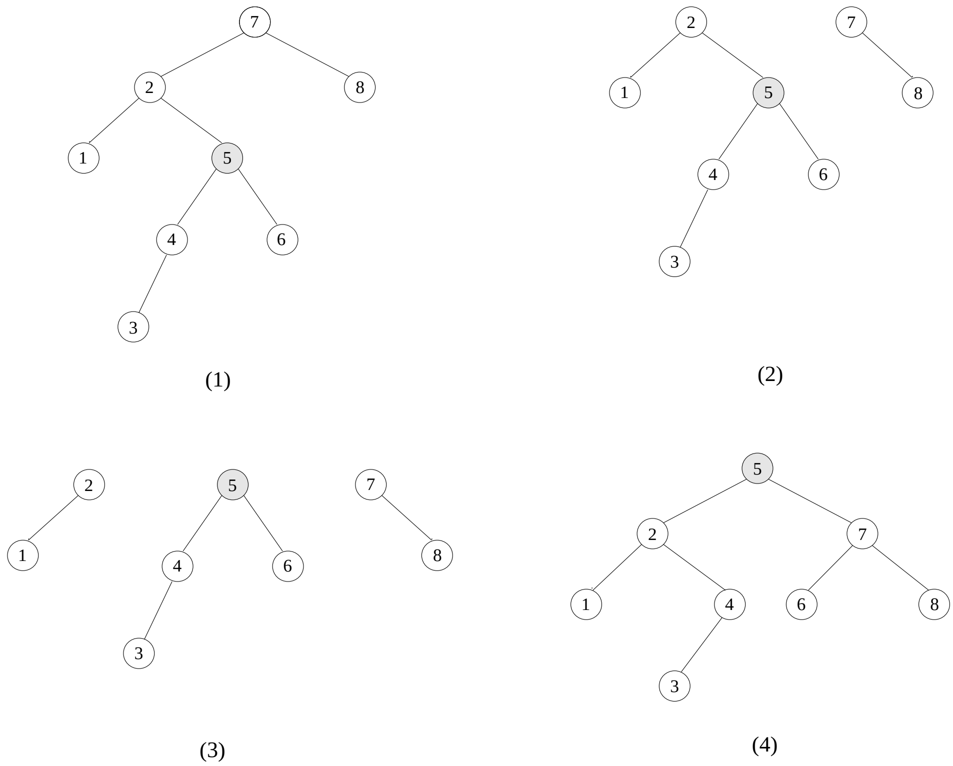


Figure 5.2.4: *Moving 5 to the root using top-down move-to-root.*

'unhooking' (step (i) or (ii) above) requires three assignments when implemented — one less than a simple exchange which needs four. Furthermore, with bottom-up move-to-root we first climb down the path to the requested node and then back up again in order to move it to the root, while with top-down we are finished once we find the requested node. The advantage of top-down move-to-root — all the restructuring is done on the way down — is also the drawback of this method. If requests are made relatively infrequently, there is plenty of time to restructure the tree between the requests and we would prefer bottom-up move-to-root since its access time is shorter.

5.3 Two users

As in the previous discussion of the move-to-front rule, we extend the model to include two users, cf. Section 2.1.

Imagine the records $1, 2, \ldots, n$ arranged in a binary search tree modified according to move-to-root. There is a high-priority 'Controller' requesting records independently at the rate of one per unit of time, each request being for record i with probability $p_i > 0$, $\sum_{i=1}^{n} p_i = 1$. When a requested record has been found and read, it is moved to the root of the tree.

There is also a low-priority 'Second User' requesting the records at a possibly different rate λ and different request probabilities q_i, $\sum_{i=1}^{n} q_i = 1$. Second User's requests are independent of previous requests, both those made by Second User and those made by Controller. Because of the lower priority, Second User is not allowed to rearrange the tree to suit his preferences.

We presume that whichever the requested record, it may always be found in the tree — there are no blocking times and no missing records.

As when we studied move-to-front, we shall use the equivalence of this model and the one obtained by the Poisson embedding in Section 2.2: Define n independent Poisson processes with intensities p_i, $i = 1, \ldots, n$ and let Controller request record i at the time-points of the one with intensity p_i. Let T_i be the time elapsed since Controller last requested record i. A similar construction may be applied to Second User's requests.

We shall study *Second User's search cost*, defined as the number of comparisons needed to find a record requested by Second User (that is, given that Second User makes a request, we consider the search cost at this time). Since this cost depends only on the current tree, in turn determined solely by Controller's requests, our results will not depend on Second User's request rate λ. The Poisson embedding nevertheless provides means to handle also unconditional properties.

Chapter 6

Search cost distribution

What is Second User's cost of searching?

Assuming unknown constant request probabilities and independent requests, we derive the expected cost at time t for a tree initially of shape τ (Theorem 6.1.1.1). Because of its complexity, the variance has been detached and is found in Appendix A. The analogous single-user expectation (and variance) is, to the best of my knowledge, a new result too.

Next, we investigate the stationary search cost — the number of comparisons needed when enough time has elapsed to ensure that the move-to-root is fully responsible for the organization of the records in the tree, independently of the initial tree. The expected stationary search cost (and the variance) follows if we let time tend to infinity in the formulas including the transient behaviour but one may also, as we do in Section 6.1.2, give a proof of combinatorial flavour. Taking Controller's and Second User's request probabilities to be identical, we recapture the known expressions for single-user move-to-root. In particular, the expectation and variance for the special case of uniform request probabilities follow.

Dwelling for a moment on single-user move-to-front, what are the 'worst' request probabilities? For move-to-front, these are the uniformly distributed ones and it used to be conjectured that this was also the case for move-to-root. In Section 6.1.3 however, we find numerically that the expected search cost is maximized by request probabilities close to but *not* equal to uniformly distributed ones. Instead the 'worst' requests have slightly larger probabilities of requesting records in the beginning and the end (indices $1, 2, \ldots$ resp. $n, n-1, \ldots$) than those in the middle.

Turning to the problem of describing Second User's search cost distribution, in Section 6.2 we derive two expressions for the generating function — one in recursive and one in integral form. We also solve the recursion (in two different ways) for equal request probabilities, recapturing the known explicit generating function and probability function for this case.

Finally, in Section 6.3, we show how to extend the results to the case of time-dependent request probabilities.

6.1 Expectation and variance

6.1.1 Transient cost

Imagine a binary search tree with n nodes rearranged according to move-to-root. Initially, at time 0, the shape of the tree is τ.

Let $S^\tau(t)$ be Second User's search cost at time t, that is, due to a Second User request at this time, $S^\tau(t)$ is the number of comparisons needed to locate the requested record. In analogy with our procedure in Section 2.3.1, introduce the indicator random variables

$$\mathbf{1}_i^j(t) = \begin{cases} 1 & \text{if } j \text{ is an ancestor of } i \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

Given that Second User requests record i at time t, let $S_i^\tau(t)$ denote the search cost. Since the search cost is measured by the number of required comparisons and since there is a unique path from the root to each node, $S_i^\tau(t) = 1 +$ the number of ancestors of i at time t, equals the length of this path:

$$S_i^\tau(t) = 1 + \sum_{j:j\neq i} \mathbf{1}_i^j(t).$$

The probability function for the search cost $S^\tau(t)$ at time t is

$$P(S^\tau(t) = k) = \sum_{i=1}^{n} q_i P(S_i^\tau(t) = k).$$

Since the probability function $P(S_i^\tau(t) = k)$ is much more complicated[1] than its move-to-front analogue, we refrain from giving a move-to-root equivalent of Proposition 2.3.1.1. Instead, for the stationary search cost, we derive the generating function in Section 6.2.

[1] The reason for this is that we now need to condition on L_j as opposed to L_i, cf. the proof of Theorem 6.1.1.1.

Theorem 6.1.1.1 *For a tree, initially in shape τ, the expectation of Second User's search cost at time t is*

$$E(S^\tau(t)) = 1 \quad + \sum_{1\le i<j\le n} \left[\frac{q_i p_j + p_i q_j}{p_i + \cdots + p_j} (1 - \exp\{-(p_i + \cdots + p_j)t\}) \right.$$
$$\left. + (q_i \mathbf{1}_i^j(0) + q_j \mathbf{1}_j^i(0)) \exp\{-(p_i + \cdots + p_j)t\} \right].$$

Remark. Note that $\mathbf{1}_i^j(0)$ is a deterministic indicator equal to one if and only if j is an ancestor of i in the initial tree τ.

Remark. Compare this expected search cost to the move-to-front one in Theorem 2.3.1.2.

The corresponding single-user result follows immediately:

Corollary 6.1.1.2 *For a tree, initially in shape τ, the expectation of a single user's search cost at time t is*

$$E(S^\tau(t)) = 1 \quad + \sum_{1\le i<j\le n} \left[\frac{2p_i p_j}{p_i + \cdots + p_j} (1 - \exp\{-(p_i + \cdots + p_j)t\}) \right.$$
$$\left. + (p_i \mathbf{1}_i^j(0) + p_j \mathbf{1}_j^i(0)) \exp\{-(p_i + \cdots + p_j)t\} \right].$$

Proof of Theorem 6.1.1.1. As in Section 2.3.1, let L_i denote the time elapsed since Controller's last request for record i, $i = 1, \ldots, n$. Since the move-to-root process has only been running for time t, this is the limit for how far back in time we can go. Hence, with the notation from the Poisson embedding (cf. Section 2.2), L_i is distributed as $\min\{T_i, t\}$.

Now, condition on $L_j = u^*$, $j \ne i$. The event that j is an ancestor of i splits into two cases:
(i) If $0 \le u^* < t$, Controller has requested j at least once before time t and j is an ancestor of i if and only if he has requested j after his most recent requests for the records between i and j (including i itself). Since by the embedding, the L_i's are independent, we have when $j < i$,

$$\begin{aligned} P_{u^*}(\mathbf{1}_i^j(t) = 1) &= P(L_{j+1} > u^*)P(L_{j+2} > u^*) \cdots P(L_i > u^*) \\ &= \exp\{-(p_{j+1} + \cdots + p_i)u^*\} \end{aligned} \tag{6.1}$$

and when $j > i$

$$\begin{aligned} P_{u^*}(\mathbf{1}_i^j(t)=1) &= P(L_i > u^*)P(L_{i+1} > u^*)\cdots P(L_{j-1} > u^*) \\ &= \exp\{-(p_i + \cdots + p_{j-1})u^*\}. \qquad (6.2) \end{aligned}$$

(ii) If $u^* = t$, Controller has never requested j. In this case j is an ancestor of i if it was an ancestor in the initial tree τ *and* if Controller has not requested any of the records between i and j (including i). Hence, when $j < i$,

$$\begin{aligned} P_{u^*}(\mathbf{1}_i^j(t)=1) &= \mathbf{1}_i^j(0)P(L_{j+1} = t)P(L_{j+2} = t)\cdots P(L_i = t) \\ &= \mathbf{1}_i^j(0)\exp\{-(p_{j+1} + \cdots + p_i)t\} \end{aligned}$$

and when $j > i$

$$\begin{aligned} P_{u^*}(\mathbf{1}_i^j(t)=1) &= \mathbf{1}_i^j(0)P(L_i = t)P(L_{i+1} = t)\cdots P(L_{j-1} = t) \\ &= \mathbf{1}_i^j(0)\exp\{-(p_i + \cdots + p_{j-1})t\}. \end{aligned}$$

Using (i) and (ii) we have

$$\begin{aligned} E(S_i^\tau(t)) = 1 \quad &+ \sum_{j:j\neq i} P(\mathbf{1}_i^j(t)=1) \\ = 1 \quad &+ \sum_{j:j<i}\Bigg[\int_0^t \exp\{-(p_{j+1}+\cdots+p_i)u^*\}p_j e^{-p_j u^*}\,du^* \\ &\qquad + \mathbf{1}_i^j(0)\exp\{-(p_{j+1}+\cdots+p_i)t\}e^{-p_j t}\Bigg] \\ &+ \sum_{j:j>i}\Bigg[\int_0^t \exp\{-(p_i+\cdots+p_{j-1})u^*\}p_j e^{-p_j u^*}\,du^* \\ &\qquad + \mathbf{1}_i^j(0)\exp\{-(p_i+\cdots+p_{j-1})t\}e^{-p_j t}\Bigg] \\ = 1 \quad &+ \sum_{j:j<i}\Bigg[\frac{p_j}{p_j+\cdots+p_i}(1-\exp\{-(p_j+\cdots+p_i)t\}) \\ &\qquad + \mathbf{1}_i^j(0)\exp\{-(p_j+\cdots+p_i)t\}\Bigg] \\ &+ \sum_{j:j>i}\Bigg[\frac{p_j}{p_i+\cdots+p_j}(1-\exp\{-(p_i+\cdots+p_j)t\}) \\ &\qquad + \mathbf{1}_i^j(0)\exp\{-(p_i+\cdots+p_j)t\}\Bigg]. \end{aligned}$$

Hence,

$$\begin{aligned} E(S^\tau(t)) &= \sum_{i=1}^{n} q_i E(S_i^\tau(t)) \\ = 1 &+ \sum_{i,j:i<j} \left[\frac{q_i p_j + p_i q_j}{p_i + \cdots + p_j} (1 - \exp\{-(p_i + \cdots + p_j)t\}) \right. \\ &\left. + (q_i \mathbf{1}_i^j(0) + q_j \mathbf{1}_j^i(0)) \exp\{-(p_i + \cdots + p_j)t\} \right]. \end{aligned}$$

□

Remark. The embedding also provides a key to the variance. Because of the complexity of this expression, it is presented and proved in Appendix A.

6.1.2 Stationary cost

In stationarity, enough time has elapsed since the tree was started to ensure that move-to-root is solely responsible for the shape of the tree — the initial tree arrangement has been 'forgotten'. Let S^{stat} denote Second User's search cost.

Theorem 6.1.2.1 *Second User's expected search cost is*

$$E\left(S^{stat}\right) = 1 + \sum_{1 \le i < j \le n} \frac{q_i p_j + p_i q_j}{p_i + \cdots + p_j} \tag{6.3}$$

and the variance

$$\begin{aligned} \operatorname{Var}\left(S^{stat}\right) &= \left(E\left(S^{stat}\right) - 1\right)\left(2 - E\left(S^{stat}\right)\right) \\ &+2 \sum_{1 \le i < j < k \le n} \frac{1}{p_i + \cdots + p_k} \left(\frac{(q_i p_j + p_i q_j) p_k}{p_i + \cdots + p_j} + \frac{p_i (q_j p_k + p_j q_k)}{p_j + \cdots + p_k} \right). \end{aligned}$$

If Second User's request probabilities are identical to those of Controller, the known single-user results follow immediately.

Corollary 6.1.2.2 (Allen and Munro, 1978) *The expected search cost for a single user is*

$$E\left(S^{stat}\right) = 1 + 2 \sum_{1 \le i < j \le n} \frac{p_i p_j}{p_i + \cdots + p_j} \tag{6.4}$$

and the variance

$$\begin{aligned} \mathrm{Var}\left(S^{stat}\right) &= \left(E\left(S^{stat}\right) - 1\right)\left(2 - E\left(S^{stat}\right)\right) \\ &+ \ 4 \sum_{1 \le j < i < k \le n} \frac{p_i p_j p_k}{p_j + \cdots + p_k}\left(\frac{1}{p_j + \cdots + p_i} + \frac{1}{p_i + \cdots + p_k}\right). \end{aligned}$$

Theorem 6.1.2.1 follows when $t \to \infty$ in Theorem 6.1.1.1 resp. Theorem A.1 in Appendix A. The proof we give below, however, is one of 'combinatorial flavour' which does not use the Poisson embedding.

Proof of Theorem 6.1.2.1. Reasoning as in the previous section, let $S_i^{stat} = 1 +$ the number of ancestors of i be the search cost needed to locate record i and introduce indicators such that

$$S_i^{stat} = 1 + \sum_{j: j \ne i} \mathbf{1}_i^j, \qquad \text{where } \mathbf{1}_i^j = \begin{cases} 1 & \text{if } j \text{ is an ancestor of } i, \\ 0 & \text{otherwise.} \end{cases}$$

If $j > i$, then by Observation 5.2.1 j is an ancestor of i if and only if Controller has not requested any of $i, \ldots, j-1$ after his last request for j and hence

$$P(\mathbf{1}_i^j = 1) = \frac{p_j}{p_i + \cdots + p_j}.$$

Similarly, if $j < i$, then j is an ancestor of i if and only if Controller has not requested any of $j+1, \ldots, i$ after his last request for j and we have

$$P(\mathbf{1}_i^j = 1) = \frac{p_j}{p_j + \cdots + p_i}.$$

Consequently,

$$\begin{aligned} E\left(S^{stat}\right) &= \sum_i q_i E(S_i^{stat}) = \sum_i q_i \left(1 + \sum_{j: j \ne i} P(\mathbf{1}_i^j = 1)\right) \\ &= 1 + \sum_{1 \le i < j \le n} \frac{q_i p_j + p_i q_j}{p_i + \cdots + p_j}. \end{aligned}$$

The variance is determined analogously, using that $\left(\mathbf{1}_i^j\right)^2 = \mathbf{1}_i^j$.

$$
\begin{aligned}
&E\left((S^{stat})^2\right)\\
&= \sum_i q_i E\Big[\big(1+\sum_{j:j\neq i}\mathbf{1}_i^j\big)^2\Big]\\
&= 1+\sum_i q_i E\Big[2\sum_{j:j\neq i}\mathbf{1}_i^j+\sum_{j:j\neq i}\big(\mathbf{1}_i^j\big)^2+\sum_{j:j\neq i}\sum_{\substack{k:k\neq j\\ k\neq i}}\mathbf{1}_i^j\mathbf{1}_i^k\Big]\\
&= 1+3(E(S^{stat})-1)+2\sum_i q_i\sum_{j:j\neq i}\sum_{\substack{k:k>j\\ k\neq i}}P(\mathbf{1}_i^j\mathbf{1}_i^k=1)
\end{aligned}
$$

It remains to determine the last term of $E\left((S^{stat})^2\right)$. We do this below as we prove Lemma 6.1.2.3.

Lemma 6.1.2.3

$$
\begin{aligned}
&\sum_i p_i\sum_{j:j\neq i}\sum_{\substack{k:k>j\\ k\neq i}}P(\mathbf{1}_i^j\mathbf{1}_i^k=1)\\
&= \sum_{1\le i<j<k\le n}\frac{1}{p_i+\cdots+p_k}\left(\frac{(q_ip_j+p_iq_j)p_k}{p_i+\cdots+p_j}+\frac{p_i(q_jp_k+p_jq_k)}{p_j+\cdots+p_k}\right).
\end{aligned}
$$

By this lemma we get

$$
\begin{aligned}
&\operatorname{Var}\left(S^{stat}\right)=E\left((S^{stat})^2\right)-\left[E\left(S^{stat}\right)\right]^2\\
&= \left(E\left(S^{stat}\right)-1\right)\left(2-E\left(S^{stat}\right)\right)\\
&\quad +2\sum_{1\le i<j<k\le n}\frac{1}{p_i+\cdots+p_k}\left(\frac{(q_ip_j+p_iq_j)p_k}{p_i+\cdots+p_j}+\frac{p_i(q_jp_k+p_jq_k)}{p_j+\cdots+p_k}\right).
\end{aligned}
$$

□

Proof of Lemma 6.1.2.3.
(i) If $j<k<i$, $\mathbf{1}_i^j\mathbf{1}_i^k=1$ if and only if $\mathbf{1}_i^j=\mathbf{1}_i^k=1$, that is, if and only if j and k are both ancestors of i. The only possible shape of a tree in which these ancestry requirements are satisfied is shown in Figure 6.1.1.

Since $j<k$, j must be an ancestor of k which on its part must be an ancestor of i — so we demand that Controller has not requested any

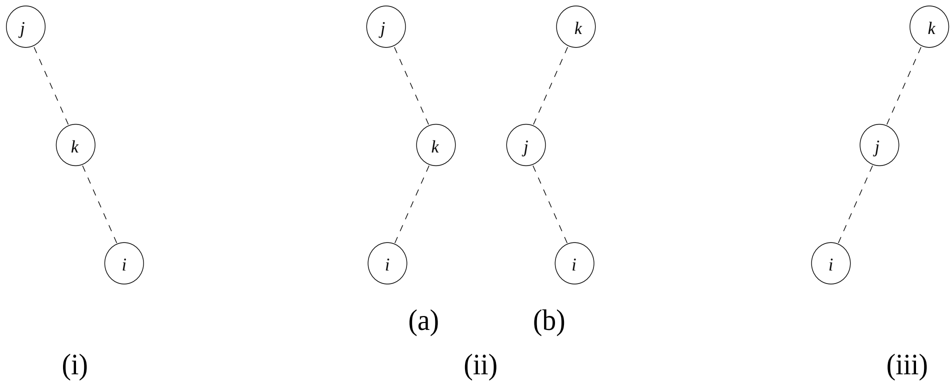


Figure 6.1.1: *Possible trees.*

of $j+1,\cdots,k$ after his last request for j and nor has he requested any of $k+1,\cdots,i$ after his last request for k. In other words, first request only k from the set $\{k,\ldots,i\}$ and then move j to the root *without ruining the ancestry between k and i*, that is, request only j from $\{j,\ldots,i\}$. The desired probability is the product of these two:

$$P(\mathbf{1}_i^j\mathbf{1}_i^k=1)=\frac{p_k}{(p_k+\cdots+p_i)}\frac{p_j}{(p_j+\cdots+p_i)}.$$

(ii) If $j<i<k$, j and k are both ancestors of i if and only if neither any of $j+1,\ldots,i$ has been requested by Controller after his last request for j, nor has he requested any of the $i,\ldots,k-1$ after his last request for k. This time there are two possible trees, Figure 6.1.1. Either request k alone from $\{i,\ldots,k\}$ to make k an ancestor of i and request j from $\{j,\ldots,k\}$. This way the tree (a) in Figure 6.1.1 is obtained. Or request only j from $\{j,\ldots,i\}$ and k from $\{j,\ldots,k\}$ to obtain the tree (b). Sum the two cases to get the probability

$$P(\mathbf{1}_i^j\mathbf{1}_i^k=1)=\frac{p_jp_k}{p_j+\cdots+p_k}\left(\frac{1}{p_i+\cdots+p_k}+\frac{1}{p_j+\cdots+p_i}\right).$$

(iii) If $i<j<k$, the close resemblance with (i) immediately gives

$$P(\mathbf{1}_i^j\mathbf{1}_i^k=1)=\frac{p_jp_k}{(p_i+\cdots+p_j)(p_i+\cdots+p_k)}.$$

Together, (i)-(iii) give

$$
\begin{aligned}
&\sum_i q_i \sum_{j:j\neq i} \sum_{\substack{k:k>j\\k\neq i}} P(\mathbf{1}_i^j \mathbf{1}_i^k = 1) \\
&= \sum_i \sum_{j:j<i} \sum_{\substack{k:k>j\\k<i}} \frac{q_i p_j p_k}{(p_j + \cdots + p_i)(p_k + \cdots + p_i)} \\
&\quad + \sum_i \sum_{j:j<i} \sum_{k:k>i} \frac{q_i p_j p_k}{p_j + \cdots + p_k} \left(\frac{1}{p_j + \cdots + p_i} + \frac{1}{p_i + \cdots + p_k} \right) \\
&\quad + \sum_i \sum_{j:j>i} \sum_{k:k>j} \frac{q_i p_j p_k}{(p_i + \cdots + p_j)(p_i + \cdots + p_k)} \\
&= \sum_{1\le i<j<k\le n} \frac{1}{p_i + \cdots + p_k} \left(\frac{(q_i p_j + p_i q_j) p_k}{p_i + \cdots + p_j} + \frac{p_i (q_j p_k + p_j q_k)}{p_j + \cdots + p_k} \right).
\end{aligned}
$$

□

From Theorem 6.1.2.1 the known (cf. Section 1.11.2.4) formulas for the expectation and variance in the special case arising when all the request probabilities are equal follow. According to Knuth [43], the expectation was first given by Windley [78], Booth and Colin [9] and Hibbard [34] who seem to have developed the method of tree insertion independently. The expression for the variance seems to have been given first by Knott in an unpublished paper, cf. Section 1.11.2.4.

Corollary 6.1.2.4 *If $p_i = q_i = 1/n$, $i = 1, \ldots, n$, then the expected search cost resp. variance are*

$$
\begin{aligned}
E\left(S^{stat}\right) &= 2\left(1 + \frac{1}{n}\right) H_n - 3 \qquad (6.5)\\
\mathrm{Var}\left(S^{stat}\right) &= \left(2 + \frac{10}{n}\right) H_n - 4\left(1 + \frac{1}{n}\right)\left(\frac{{H_n}^2}{n} + H_n(2)\right) + 4
\end{aligned}
$$

where H_n and $H_n(2)$, the n:th harmonic number of order one resp. two, were defined on p. 74.

Proof. Tedious but straightforward.

□

6.1.3 Worst case for single-user move-to-root

Suppose that a single user considers the records to be equally popular. Then, no impopular records can be rotated down the tree leaving the frequently requested ones close to the root. It might therefore seem like a reasonable guess that for a single user, uniform request probabilities give the maximum search cost.

For single-user move-to-front, Fill and Holst [23] showed that the expected (stationary) search cost is a Schur-concave[2] function and hence the maximum is achieved for uniform request probabilities. Unfortunately, however, since the expected (stationary) search cost for move-to-root is not symmetric with respect to the p_i's, it does not possess this feature. In fact, Dobrow and Fill [17, Section 3.1] produce a numerical example supporting the suspicion that uniform request probabilities do in fact *not* maximize the expectation. What set of p_i's does?

A numerical solution to the optimization problem of determining the most unfortunate choice of request probabilities,

$$\max_{p_1,\dots,p_n} \; 1 + 2 \sum_{1\le i<j\le n} \frac{p_i p_j}{p_i + \cdots + p_j},$$

or equivalently

$$\min_{p_1,\dots,p_n} \; - \sum_{1\le i<j\le n} \frac{p_i p_j}{p_i + \cdots + p_j},$$

using the procedure *NPSOL*, shows that these p_i's are symmetric and U-shaped (Figure 6.1.2). *NPSOL*, (cf. Gill, Murray and Saunders [27]) solves the nonlinear programming problem using a sequential quadratic programming (SQP) method. In short, such a method solves a sequence of quadratic programming subproblems (quadratic objective function and linear constraints). The solution to these subproblems converges to a solution of the original problem. For a thorough description, see e.g. Fletcher [26, pp. 304–317] or Gill, Murray and Wright [28, pp. 237–242].

The numerical results reveal that the maximum expected search cost, even initially being very close to the one answering to the uniform case, gets closer and closer to the uniform expected search cost as n increases (in Figure 6.1.4 they can hardly be distinguished), whereas the distribution of the 'worst' request probabilities, indeed being very close to the uniform, still shows no tendency to converge to a uniform distribution (Figure 6.1.2).

[2]For details on the concept of Schur-concavity, see for instance Marshall and Olkin [54, Chapter 2, Section B].

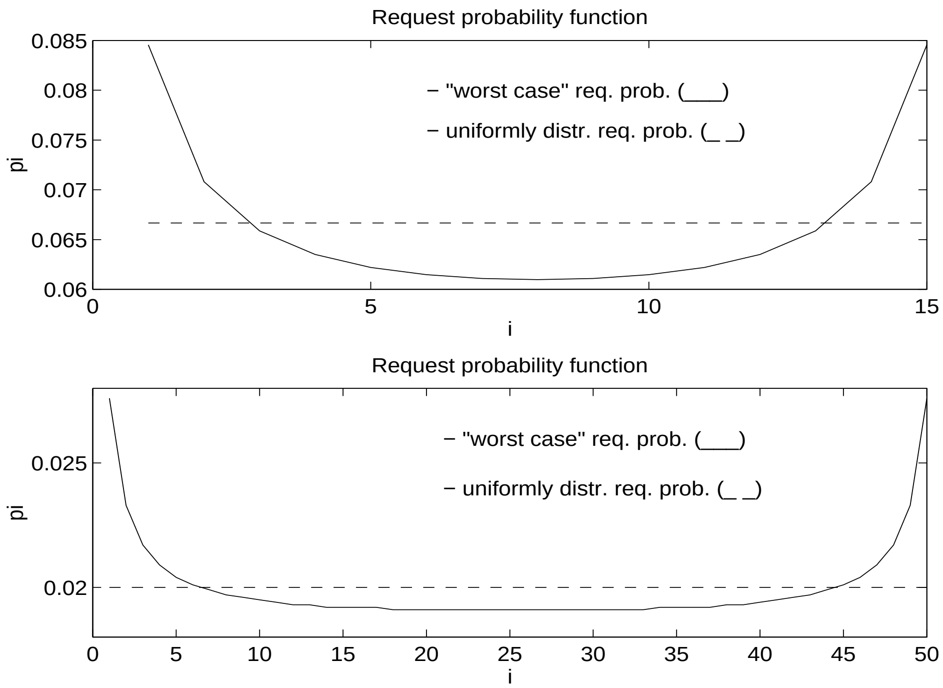


Figure 6.1.2: *Request probabilities.* Uniform and worst case request probabilities (p_i's) when $n = 15$ resp. $n = 50$. (The points of the probability function have been joined.)

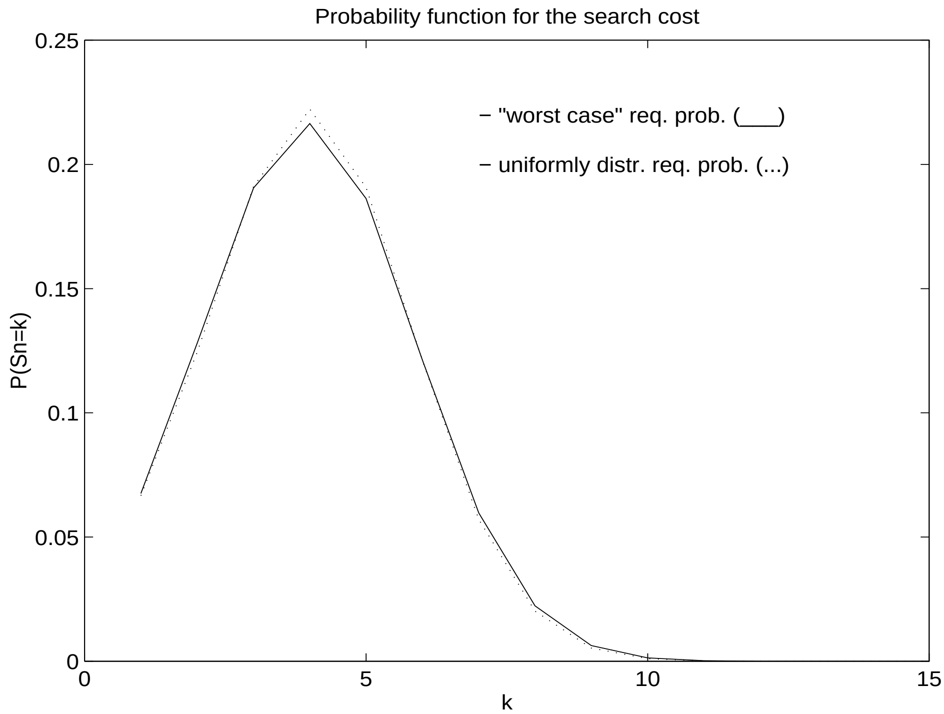


Figure 6.1.3: *Search cost distribution.* Probability function for the search cost due to the request probabilities $n = 15$ in the previous figure. (The points of the probability function have been joined.)

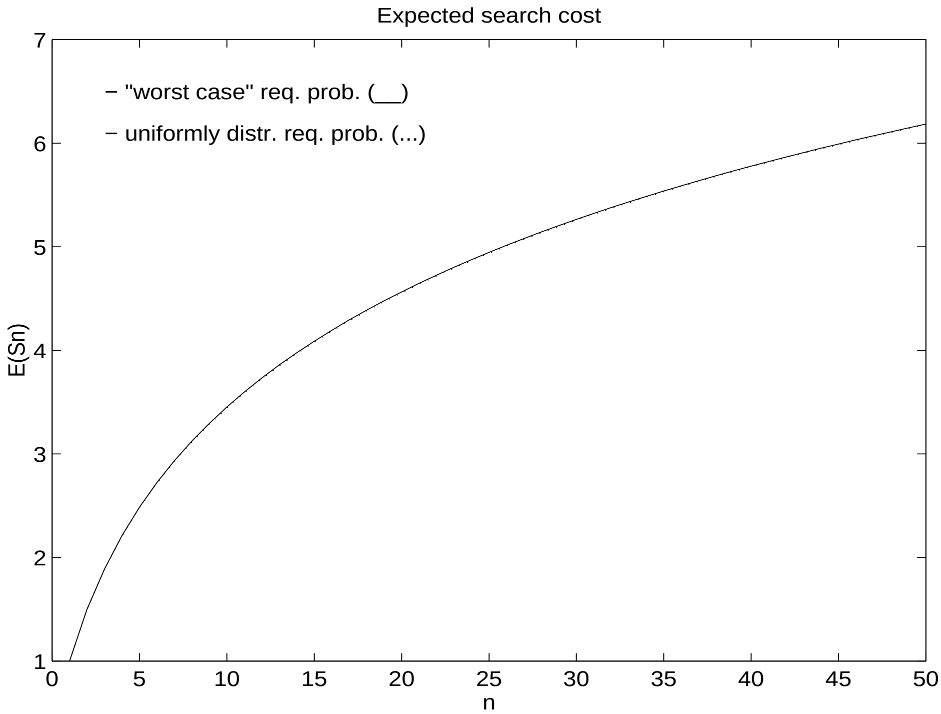


Figure 6.1.4: *Expected search cost.* $E\left(S^{stat}\right)$ as a function of n for worst case and uniform request probabilities.

However, the similarity in search cost behaviour between the two cases suggests that uniform probabilities provide a good approximation of the worst case and this result justifies the importance of the uniform case. In Section 6.2.3 we study the search cost due to uniform request probabilities further.

Remark. Figure 6.1.2 has been generated using Matlab implementations of *NPSOL* while for Figure 6.1.3 the recursion for the generating function (Corollary 6.2.4) and its special case solution (Corollary 6.2.2) for the probability function when the request probabilities are uniform have been used. Finally, the equations (6.4) and (6.5) for the expected search cost have been used for Figure 6.1.4.

Remark. Although the maximum expected cost at first glance might seem to converge to the uniform one, this cannot be the case (Bodell [7], cf. Figure 6.1.4 above). Since we are now dealing with the most unfortunate choice of request probabilities, the corresponding expectation is greater than those arising from any other set of request probabilities. In particular, it is greater than the expectation obtained when the request probabilities are all equal (Corollary 6.1.2.4)

$$2(1+\frac{1}{n})H_n - 3.$$

However, this mean is not bounded since $H_n \to \infty$ as $n \to \infty$.

6.2 Generating function

Determining the probability function for Second User's search cost by means of an analogue to Proposition 2.3.1.1 turns out not to be all that easy even if we restrict ourselves to the stationary case. With notation from Section 6.1.2, the probability function for the search cost given that Second User requests record i is

$$P(S_i^{stat} = k) \quad = \quad P\left(1 + \sum_{j:j\neq i} \mathbf{1}_i^j = k\right).$$

Conditioning on the time elapsed since Controller last requested the records, using the embedding in Section 2.2, we get

$$\begin{aligned} P(S_i^{stat} = k) &= \int \cdots \int P\left(\sum_{j:j \neq i} \mathbf{1}_i^j = k-1 \middle| T_1 = t_1^*, \cdots, T_n = t_n^* \right) \\ &\quad \times p_1 e^{-p_1 t_1^*} \cdots p_n e^{-p_n t_n^*} dt_1^* \cdots dt_n^*. \end{aligned}$$

By Observation 5.2.1 we know that if $j < i$, then

$$\mathbf{1}_i^j = 1 \text{ if } t_{j+1}^* > t_j^*, \cdots, t_i^* > t_j^* \tag{6.6}$$

and if $j > i$,

$$\mathbf{1}_i^j = 1 \text{ if } t_i^* > t_j^*, \cdots, t_{j-1}^* > t_j^*. \tag{6.7}$$

Hence,

$$P(S_i^{stat} = k) = \int \cdots \int_A \left(\prod_{j=1}^n p_j \right) \exp\Big\{ -\sum_{j=1}^n p_j t_j^* \Big\} dt_1^* \cdots dt_n^*,$$

where A is the integration area corresponding to all possible choices of t_j^*-values, $j = 1, \ldots, n$, such that $k-1$ of the conditions (6.6)-(6.7) are satisfied. Determining A may be rather troublesome and in this section we give alternative expressions for the probability distribution of Second User's search cost, namely in terms of a generating function. In fact, written as a recursive equation, the generating function proves to be rather handy even when the transient search cost is included.

If the request probabilities are equal, there are explicit expressions for the generating function and the probability function. These expressions can be obtained from the recursion (Theorem 6.2.1) which we will show in Section 6.2.3.

Alternatively, we can express the generating function in integral form as in Section 6.2.2.

6.2.1 Recurrence relation

Let $S_{k,m}^{\tau}(t)$ denote Second User's search cost at time t in a binary search tree consisting of nodes at which records $k, k+1, \ldots, m$ are stored, where the initial shape of the tree is known to have been τ. Strictly speaking, τ is a function of the present records $k, \ldots, m$ but for ease of notation we choose to write simply τ. Another reason for this decision is our current aim. We shall derive a generating function for the search cost at time t in the tree consisting of *all* n records given its initial shape. Knowing the shape of this tree, we automatically know the shape of its every subtree. Hence, denoting by τ the entire initial tree induces the intended association.

Using the notation from Section 6.1.1, in particular letting the (deterministic) indicator $\mathbf{1}_i^j(0) = 1$ if and only if j is an ancestor of i in τ, we have the following recursion for the generating function for the search cost.

Theorem 6.2.1 *The generating function for Second User's search cost at time t when τ is the initial tree, $S^\tau(t) = S^\tau_{1,n}(t)$, is given by the recursive equation*

$$\begin{cases} E(z^{S^\tau_{k,m}(t)}) = \dfrac{z}{(p_k + \cdots + p_m)(q_k + \cdots + q_m)} \\ \qquad \times \displaystyle\sum_{r=k}^{m} p_r \Bigg[1 - \bigg(1 - \frac{(p_k + \cdots + p_m)}{p_r} \prod_{l \neq r} \mathbf{1}_l^r(0)\bigg) \\ \qquad\qquad \times \exp\{-(p_k + \cdots + p_m)t\}\Bigg] \\ \qquad \times \Bigg[E(z^{S^\tau_{k,r-1}(t)}) \displaystyle\sum_{j=k}^{r-1} q_j + q_r + E(z^{S^\tau_{r+1,m}(t)}) \sum_{j=r+1}^{m} q_j\Bigg] \\ E(z^{S^\tau_{k,k}}) = z. \end{cases} \tag{6.8}$$

Remark. Note that l in equation (6.8) ranges from k to m.

If Second User's preferences are identical to Controller's we have the single-user analogue:

Corollary 6.2.2 *The generating function for a single user's search cost at time t when τ is the initial tree, $S^\tau(t) = S^\tau_{1,n}(t)$, is given by*

$$\begin{cases} E(z^{S^\tau_{k,m}(t)}) = \dfrac{z}{(p_k + \cdots + p_m)^2} \\ \qquad \times \displaystyle\sum_{r=k}^{m} p_r \Bigg[1 - \bigg(1 - \frac{(p_k + \cdots + p_m)}{p_r} \prod_{l \neq r} \mathbf{1}_l^r(0)\bigg) \\ \qquad\qquad \times \exp\{-(p_k + \cdots + p_m)t\}\Bigg] \\ \qquad \times \Bigg[E(z^{S^\tau_{k,r-1}(t)}) \displaystyle\sum_{j=k}^{r-1} p_j + p_r + E(z^{S^\tau_{r+1,m}(t)}) \sum_{j=r+1}^{m} p_j\Bigg] \\ E(z^{S^\tau_{k,k}}) = z. \end{cases}$$

Write $S_{k,m}$ for the stationary search cost in a tree containing records $k,\ldots,m$. Then, letting $t\to\infty$ in Theorem 6.2.1 we have the generating function for the stationary search cost.

Corollary 6.2.3 *The generating function for Second User's stationary search cost, $S^{stat}=S_{1,n}$, is given by*

$$\begin{cases} E(z^{S_{k,m}}) = \dfrac{z}{(p_k+\cdots+p_m)(q_k+\cdots+q_m)} \\ \qquad\qquad \times \displaystyle\sum_{r=k}^{m} p_r\left[E(z^{S_{k,r-1}})\sum_{j=k}^{r-1} q_j + q_r + E(z^{S_{r+1,m}})\sum_{j=r+1}^{m} q_j\right] \\ E(z^{S_{k,k}}) = z. \end{cases} \tag{6.9}$$

Finally, letting Second User's preferences be identical to Controller's we again have the analogous single-user result.

Corollary 6.2.4 *The generating function for a single user's stationary search cost, $S^{stat}=S_{1,n}$, is given by*

$$\begin{cases} E(z^{S_{k,m}}) = \dfrac{z}{(p_k+\cdots+p_m)^2} \\ \qquad\qquad \times \displaystyle\sum_{r=k}^{m} p_r\left[E(z^{S_{k,r-1}})\sum_{j=k}^{r-1} p_j + p_r + E(z^{S_{r+1,m}})\sum_{j=r+1}^{m} p_j\right] \\ E(z^{S_{k,k}}) = z. \end{cases}$$

Proof of Theorem 6.2.1. Consider the search cost $S^{\tau}_{k,m}(t)$ as defined above. Then, when conditioning on the root R:

$$S^{\tau}_{k,m}(t)\ \Big|\ R=r \quad = \begin{cases} 1 & \text{if the root } r \text{ is requested,} \\ 1+S^{\tau}_{k,r-1}(t) & \text{if the requested record has key value less than that of } r \text{ and therefore is to be found in the left subtree,} \\ 1+S^{\tau}_{r+1,m}(t) & \text{if the requested record has key value greater than that of } r \text{ and consequently is in the right subtree.} \end{cases}$$

Thus the generating function for $S^{\tau}_{k,m}(t)$ is

$$\begin{aligned}
E(z^{S^{\tau}_{k,m}(t)}) &= z\sum_{r=k}^{m} P(R=r) \\
&\times\Big[E(z^{S^{\tau}_{k,r-1}(t)})P(\text{req. record in left subtree}|R=r) \\
&\quad +1\cdot P(\text{req. root}|R=r) \\
&\quad +E(z^{S^{\tau}_{r+1,m}(t)})P(\text{req. record in right subtree}|R=r)\Big].
\end{aligned}$$

Because this tree consists only of records $k,\ldots,m$, the probability that record i is the one currently requested by Second User is $q_i/(q_k+\cdots+q_m)$. Further, r is the root if (and only if) it is an ancestor of every other node in the tree. So Controller's must have requested r after his most recent requests for any of the others or r was the root of the initial tree τ and none of the records has yet been requested by Controller. Again, we let L_i be the time elapsed since Controller last requested record i.

$$\begin{aligned}
P(R=r) &= P(L_r<L_k,\ldots,L_r<L_m) \\
&= \int_0^t P(u^*<L_k,\ldots,u^*<L_m|L_r=u^*)p_r e^{-p_r u^*}du^* \\
&\quad + e^{-p_r t}\prod_{l\neq r}\left(\mathbf{1}^r_l(0)e^{-p_l t}\right) \\
&= \int_0^t p_r\exp\{-(p_k+\cdots+p_m)u^*\}du^* \\
&\quad +\exp\{-(p_k+\cdots+p_m)t\}\prod_{l\neq r}\mathbf{1}^r_l(0) \\
&= \frac{p_r}{p_k+\cdots+p_m}\Bigg[1-\left(1-\frac{p_k+\cdots+p_m}{p_r}\prod_{l\neq r}\mathbf{1}^r_l(0)\right) \\
&\qquad\qquad \times\exp\{-(p_k+\cdots+p_m)t\}\Bigg].
\end{aligned}$$

Hence,

$$
\begin{aligned}
&E(z^{S^{\tau}_{k,m}(t)})\\
&= z\sum_{r=k}^{m} \frac{p_r}{p_k+\cdots+p_m}\left[1-\left(1-\frac{p_k+\cdots+p_m}{p_r}\prod_{l\neq r}\mathbf{1}^r_l(0)\right)\exp\{-(p_k+\cdots+p_m)t\}\right]\\
&\times\left[\frac{q_k+\cdots+q_{r-1}}{q_k+\cdots+q_m}E(z^{S^{\tau}_{k,r-1}(t)})+\frac{q_r}{q_k+\cdots+q_m}+\frac{q_{r+1}+\cdots+q_m}{q_k+\cdots+q_m}E(z^{S^{\tau}_{r+1,m}(t)})\right].
\end{aligned}
$$

If there is only one node in the tree, only one comparison is made. This gives the boundary condition $E(z^{S^{\tau}_{k,k}(t)}) = z$.

$\square$

6.2.2 Integral form

We now use the Poisson embedding to find an alternative expression for the generating function. Theorem 6.2.2.1 deals only with the stationary search cost but it is of course possible to include also the transient behaviour. The modifications of the proof are completely straightforward. However, since the transient expression is considerably more complicated, in this presentation we restrict ourselves to the stationary case below.

Theorem 6.2.2.1 *The generating function for Second User's stationary search cost is given by*

$$
\begin{aligned}
E\left(z^{S^{stat}}\right) &= z\left(\prod_{k=1}^{n} p_k\right)\sum_{i=1}^{n} q_i \int_0^{\infty} \exp\{-p_i t_i^*\}\\
&\times \left[\int_0^{\infty}\cdots\int_0^{\infty} z^{\sum_{j=1}^{i-1} I\{t_j^*<\min_{j+1\le k\le i} t_k^*\}} \exp\left\{-\sum_{k=1}^{i-1} p_k t_k^*\right\} dt_1^*\cdots dt_{i-1}^*\right]\\
&\times \left[\int_0^{\infty}\cdots\int_0^{\infty} z^{\sum_{j=i+1}^{n} I\{t_j^*<\min_{i\le k\le j-1} t_k^*\}} \exp\left\{-\sum_{k=i+1}^{n} p_k t_k^*\right\} dt_{i+1}^*\cdots dt_n^*\right] dt_i^*.
\end{aligned}
\tag{6.10}
$$

Remark. In Bodell [7, Section 5.2.2] the recursive nature of the integration areas is exploited in order to convert (6.10) into a 'programmable form'.

Proof. Suppose that record i is requested by Second User and consider the indicators $\mathbf{1}_i^j$. From the Poisson embedding we have

$$
\begin{array}{lcllll}
\mathbf{1}_i^1 = 1 & \Longleftrightarrow & T_1 < T_2, & T_1 < T_3, & \cdots & T_1 < T_i \\
\mathbf{1}_i^2 = 1 & \Longleftrightarrow & & T_2 < T_3, & \cdots & T_2 < T_i \\
\cdots & & & & \cdots & \\
\mathbf{1}_i^{i-2} = 1 & \Longleftrightarrow & & & T_{i-2} < T_{i-1}, & T_{i-2} < T_i \\
\mathbf{1}_i^{i-1} = 1 & \Longleftrightarrow & & & & T_{i-1} < T_i
\end{array}
$$

$$
\begin{array}{lcllll}
\mathbf{1}_i^{i+1} = 1 & \Longleftrightarrow & T_{i+1} < T_i & & & \\
\mathbf{1}_i^{i+2} = 1 & \Longleftrightarrow & T_{i+2} < T_i, & T_{i+2} < T_{i+1} & & \\
\cdots & & & & \cdots & \\
\mathbf{1}_i^{n} = 1 & \Longleftrightarrow & T_n < T_i, & T_n < T_{i+1}, & \cdots & T_n < T_{n-1}
\end{array}
$$

First condition on $T_i = t_i^*$ to dissolve the dependence between $\{\mathbf{1}_i^1, \ldots, \mathbf{1}_i^{i-1}\}$ and $\{\mathbf{1}_i^{i+1}, \ldots, \mathbf{1}_i^n\}$. Then condition on $T_{i-1} = t_{i-1}^*$ to liberate $\mathbf{1}_i^{i-1}$ from $\{\mathbf{1}_i^1, \ldots, \mathbf{1}_i^{i-2}\}$. Continue by successively conditioning on $T_{i-2} = t_{i-2}^*, \ldots, T_1 = t_1^*$. Proceed in the same way to dissolve the dependence between the remaining indicators $\mathbf{1}_i^{i+1}, \ldots, \mathbf{1}_i^n$, that is, successively condition on $T_{i+1} = t_{i+1}^*, \ldots, T_n = t_n^*$.

The generating function for Second User's search cost is

$$
E\left(z^{S^{stat}}\right) = \sum_{i=1}^{n} q_i E\left(z^{S_i^{stat}}\right) = z \sum_{i=1}^{n} q_i E\left(z^{\sum_{j:j \neq i} \mathbf{1}_i^j}\right),
$$

where $E\left(z^{\sum_{j:j \neq i} \mathbf{1}_i^j}\right)$ is determined following the scheme outlined above:

$$
E\left(z^{\sum_{j:j \neq i} \mathbf{1}_i^j}\right) = \int_0^\infty p_i \exp\{-p_i t_i^*\} E\left(z^{\sum_{j:j<i} \mathbf{1}_i^j} \,\middle|\, T_i = t_i^*\right) E\left(z^{\sum_{j:j>i} \mathbf{1}_i^j} \,\middle|\, T_i = t_i^*\right) dt_i^*.
$$

We have

$$
\begin{aligned}
E\Bigg(z^{\sum_{j:j\neq i}\mathbf{1}_i^j}\Bigg) &= \int_0^\infty p_i \exp\{-p_i t_i^*\} \int_0^\infty p_{i-1}\exp\{-p_{i-1}t_{i-1}^*\}\cdot z^{I\{t_{i-1}^*<t_i^*\}} \\
&\quad \times E\Bigg(z^{\sum_{j=1}^{i-2}\mathbf{1}_i^j}\,\Bigg|\, T_i=t_i^*, T_{i-1}=t_{i-1}^*\Bigg) E\Bigg(z^{\sum_{j:j>i}\mathbf{1}_i^j}\,\Bigg|\, T_i=t_i^*\Bigg) dt_{i-1}^* dt_i^* \\
&= \int_0^\infty p_i \exp\{-p_i t_i^*\} \int_0^\infty p_{i-1}\exp\{-p_{i-1}t_{i-1}^*\}\cdot z^{I\{t_{i-1}^*<t_i^*\}} \\
&\quad \times \int_0^\infty p_{i-2}\exp\{-p_{i-2}t_{i-2}^*\}\cdot z^{I\{t_{i-2}^*<t_{i-1}^*, t_{i-2}^*<t_i^*\}} \\
&\quad \times E\Bigg(z^{\sum_{j=1}^{i-3}\mathbf{1}_i^j}\,\Bigg|\, T_i=t_i^*, T_{i-1}=t_{i-1}^*, T_{i-2}=t_{i-2}^*\Bigg) \\
&\quad \times E\Bigg(z^{\sum_{j:j>i}\mathbf{1}_i^j}\,\Bigg|\, T_i=t_i^*\Bigg) dt_{i-2}^* dt_{i-1}^* dt_i^* \\
&= \quad \cdots \\
&= \Bigg(\prod_{k=1}^{i} p_k\Bigg) \int_0^\infty \int_0^\infty \cdots \int_0^\infty z^{\sum_{j=1}^{i-1} I\left\{t_j^* < \min_{j+1\le k\le i} t_k^*\right\}} \\
&\quad \times E\Bigg(z^{\sum_{j:j>i}\mathbf{1}_i^j}\,\Bigg|\, T_i=t_i^*\Bigg) \exp\Bigg\{-\sum_{k=1}^{i} p_k t_k^*\Bigg\} dt_1^* \cdots dt_{i-1}^* dt_i^*.
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
E\Bigg(z^{\sum_{j:j>i}\mathbf{1}_i^j}\,\Bigg|\, T_i=t_i^*\Bigg) &= \Bigg(\prod_{k=i+1}^{n} p_k\Bigg) \int_0^\infty \cdots \int_0^\infty z^{\sum_{j=i+1}^{n} I\left\{t_j^* < \min_{i\le k\le j-1} t_k^*\right\}} \\
&\quad \times \exp\Bigg\{-\sum_{k=i+1}^{n} p_k t_k^*\Bigg\} dt_{i+1}^* \cdots dt_n^*.
\end{aligned}
$$

$\square$

6.2.3 Equal request probabilities

Properties of the search cost in the special case where the records are requested with uniform probabilities have been considered in several different papers, cf. Section 1.11.2.4. Among other things, the generating function and probability function (in stationarity) have been derived. Here, we use the recursive equation (6.8) to recapture these results.

Corollary 6.2.1 (Brown and Shubert, 1984) *The generating function for the stationary search cost with uniform request probabilities is*

$$E\left(z^{S^{stat}}\right) = \frac{z}{n!n} \frac{2z(2z+1)\cdots(2z+n-1)-n!}{2z-1}. \tag{6.11}$$

In Section 6.2.3.1 we prove Corollary 6.2.1 by solving the recursive equation straightforwardly while in Section 6.2.3.2 we rewrite it into a differential equation.

From the generating function the probability function is then obtained.

Corollary 6.2.2 (Brown and Shubert, 1984) *The probability function for the stationary search cost with uniform request probabilities is*

$$P\left(S^{stat}=k\right) = \frac{2^{k-1}}{n!n} \sum_{j=k}^{n} \left[{n \atop j}\right],$$

where $\left[{n \atop j}\right]$ *is the* j*:th signless Stirling number of the first kind.*

Remark. For a paper on Stirling numbers, see for instance Knuth [44].

Proof. We need to find the coefficient of z^k in the generating function (6.11), since it equals the probability that the search cost is k.

From the definition of the Stirling numbers of the first kind,

$$2z(2z+1)\cdots(2z+n-1) = \sum_{j=1}^{n} \left[{n \atop j}\right] (2z)^j$$

and $n! = \sum_{j=1}^{n} \left[{n \atop j}\right]$. Hence

$$\begin{aligned} E(z^{S^{stat}}) &= \frac{z}{n!n} \sum_{j=1}^{n} \left[{n \atop j}\right] \left(\frac{(2z)^j - 1}{2z-1}\right) = \frac{z}{n!n} \sum_{j=1}^{n} \left[{n \atop j}\right] \sum_{k=0}^{j-1} (2z)^k \\ &= \frac{1}{n!n} \sum_{k=0}^{n-1} \sum_{j=k}^{n} \left[{n \atop j}\right] 2^k z^{k+1} = \frac{1}{n!n} \sum_{k=1}^{n} \sum_{j=k}^{n} \left[{n \atop j}\right] 2^{k-1} z^k. \end{aligned}$$

The coefficient of z^k gives $P\left(S^{stat}=k\right)$.

□

6.2.3.1 Solving a recursive equation

Proof of Corollary 6.2.1. Putting the p_i's equal to $1/n$ considerably simplifies the recursive expression (6.8). Since all the probabilities are now equal, the identities of the records in a certain subtree are no longer of importance — the number of records alone provides sufficient information. Let S_k be the search cost in a tree with k items, all requested with equal probabilities. Then $E(z^{S_{k,m}})$ can be written as $E(z^{S_{m-k+1}})$ and (6.8) becomes

$$\begin{aligned}
E\left(z^{S^{stat}}\right) &= E\left(z^{S_n}\right) \\
&= z\sum_{r=1}^{n}\frac{1}{n}\left[\frac{r-1}{n}E(z^{S_{r-1}})+\frac{1}{n}+\frac{n-r}{n}E(z^{S_{n-r}})\right] \\
&= \frac{z}{n}\left[1+\sum_{r=1}^{n}\left(\frac{r-1}{n}E(z^{S_{r-1}})+\frac{n-r}{n}E(z^{S_{n-r}})\right)\right] \\
&= \frac{z}{n}\left[1+2\sum_{r=1}^{n}\frac{r-1}{n}E(z^{S_{r-1}})\right]=\frac{z}{n}\left[1+2\sum_{i=1}^{n-1}\frac{i}{n}E(z^{S_i})\right]. \qquad (6.12)
\end{aligned}$$

Introducing

$$\begin{cases} f_n(z) := \dfrac{n}{z}E\left(z^{S_n}\right), & n\geq 1, \\ f_1(z) := 1, & \end{cases}$$

the recursion becomes

$$f_n(z)=1+2\frac{z}{n}\sum_{i=1}^{n-1}f_i(z)=1+2\frac{z}{n}(f_{n-1}(z)+\sum_{i=1}^{n-2}f_i(z)). \qquad (6.13)$$

Now substitute $n-1$ for n in the leftmost equality of (6.13) and rearrange the terms:

$$\sum_{i=1}^{n-2}f_i(z)=(f_{n-1}(z)-1)\frac{n-1}{2z}. \qquad (6.14)$$

Combine (6.13) and (6.14) to get

$$\begin{aligned} f_n(z) &= 1+\frac{2z}{n}\left[f_{n-1}(z)+(f_{n-1}(z)-1)\frac{n-1}{2z}\right] \\ &= \frac{1}{n}+\frac{1}{n}(2z+n-1)f_{n-1}(z). \end{aligned} \tag{6.15}$$

Multiply (6.15) by n and let $g_n(z) := nf_n(z)$. Then

$$\begin{cases} g_n(z) = 1+\dfrac{2z+n-1}{n-1}g_{n-1}(z), \quad n \geq 1, \\ g_1(z) = 1, \end{cases}$$

From this we have

$$\begin{aligned} g_{n+1}(z) &= 1+\frac{2z+n}{n}g_n(z) \\ &= 1+\frac{2z+n}{n}+\frac{(2z+n)}{n}\frac{(2z+n-1)}{n-1}+\cdots \\ &\quad +\frac{(2z+n)(2z+n-1)\cdots(2z+1)}{n!}. \end{aligned}$$

Since $g_n(z) = \frac{n^2}{z}E\left(z^{S_n}\right)$, this implies that

$$\begin{aligned} E(z^{S_{n+1}}) &= \frac{z}{(n+1)^2}\left(1+\sum_{k=1}^{n}\frac{(2z+n)\cdots(2z+n-k+1)}{n(n-1)\cdots(n-k+1)}\right) \\ &= \frac{z}{(n+1)^2}\left(1+\sum_{k=1}^{n}\frac{\binom{2z+n}{k}}{\binom{n}{k}}\right). \end{aligned}$$

By Lemma 6.2.3 below,

$$1+\sum_{k=1}^{n}\frac{\binom{2z+n}{k}}{\binom{n}{k}} = \frac{\binom{2z+n}{n+1}-1}{\frac{2z+n}{n+1}-1}$$

and so

$$E(z^{S_{n+1}}) = \frac{z}{(n+1)!(n+1)}\frac{2z(2z+1)\cdots(2z+n)-(n+1)!}{2z-1}.$$

$\square$

Lemma 6.2.3

$$\sum_{k=0}^{n} \frac{\binom{x}{k}}{\binom{n}{k}} = \frac{\binom{x}{n+1} - 1}{\frac{x}{n+1} - 1}$$

Proof. The statement can be rewritten as

$$1 + \sum_{k=1}^{n} \frac{x(x-1)\cdots(x-k+1)}{n(n-1)\cdots(n-k+1)} = \frac{\frac{x(x-1)\cdots(x-n)}{(n+1)!} - 1}{\frac{x}{n+1} - 1}$$

and hence if we can show this equality we are done.

Rearranging, we have

$$\begin{aligned}
&x(x-1)\cdots(x-n)\\
&= (n+1)! + (n+1)!\left[\frac{x}{n+1} - 1\right]\left[1 + \sum_{k=1}^{n} \frac{x(x-1)\cdots(x-k+1)}{n(n-1)\cdots(n-k+1)}\right]\\
&= n!x + [x-(n+1)]\sum_{k=1}^{n} \frac{n!}{n(n-1)\cdots(n-k+1)} x(x-1)\cdots(x-k+1)\\
&= n!x + [x-(n+1)]\sum_{k=1}^{n} (n-k)!x(x-1)\cdots(x-k+1)\\
&= n!x + \sum_{k=1}^{n} (n-k)!x(x-1)\cdots(x-k+1)\,[(x-k)-(n-k+1)]\\
&= n!x + \sum_{k=1}^{n} (n-k)!x(x-1)\cdots(x-k+1)(x-k)\\
&\qquad - \sum_{k=1}^{n} (n-k+1)!x(x-1)\cdots(x-k+1)\\
&= \sum_{k=0}^{n} (n-k)!x(x-1)\cdots(x-k) - \sum_{l=0}^{n-1} (n-l)!x(x-1)\cdots(x-l)\\
&= 0!x(x-1)\cdots(x-n).
\end{aligned}$$

$\square$

6.2.3.2 Solving a differential equation

Proof of Corollary 6.2.1. Again consider the recursion (6.8). As in the previous section, equation (6.12), when all p_i's are equal,

$$E\left(z^{S^{stat}}\right) = E\left(z^{S_n}\right) = \frac{z}{n}\left[1 + 2\sum_{i=1}^{n-1}\frac{i}{n}E(z^{S_i})\right],$$

where again we let S_k be the search cost in a tree with k equally popular items. Rewrite the recursion in terms of $a_n(z) := nE\left(z^{S_n}\right)$:

$$a_n(z) = z\left(1 + \frac{2}{n}\sum_{i=1}^{n-1} a_i(z)\right),$$

that is,

$$\begin{cases} na_n(z) = z\left(n + 2\sum_{i=1}^{n-1} a_i(z)\right), & n > 1, \\ a_1(z) = z. \end{cases} \tag{6.16}$$

Multiply (6.16) by s^{n-1}, then sum over $n \geq 1$:

$$\sum_{n=1}^{\infty} na_n(z)s^{n-1} = z\left[\sum_{n=1}^{\infty} ns^{n-1} + 2\sum_{n=2}^{\infty}\sum_{i=1}^{n-1} a_i(z)s^i s^{n-1-i}\right]. \tag{6.17}$$

Now, let $A(s) := \sum_{n=1}^{\infty} a_n(z)s^n$. Then the left-hand side of (6.17),

$$\sum_{n=1}^{\infty} na_n(z)s^{n-1} = \frac{d}{ds}A(s).$$

Considering the right-hand side of (6.17) we note that $\sum_{n=1}^{\infty} ns^{n-1}$ is an arithmetic-geometric series, equaling

$$\frac{1}{1-s} + \frac{s}{(1-s)^2} = \frac{1}{(1-s)^2}.$$

Changing the order of summation in the last term of (6.17)

$$\sum_{n=2}^{\infty}\sum_{i=1}^{n-1} a_i(z)s^i s^{n-1-i} = \sum_{i=1}^{\infty} a_i(z)s^i \sum_{n=i}^{\infty} s^{n-i} = A(s)\frac{1}{1-s}.$$

Hence, (6.17) can be written as a differential equation for the generating function $A(s)$ for $a_n(z)$, namely,

$$\begin{cases} A'(s) - \dfrac{2z}{1-s}A(s) = \dfrac{z}{(1-s)^2} \\ A(0) = 0. \end{cases} \tag{6.18}$$

Recall that $a_n(z) = nE\left(z^{S_n}\right)$. Consequently,

$$A(s) = \sum_{n=1}^{\infty} nE\left(z^{S_n}\right) s^n$$

and the solution of the differential equation yields the generating function for the search cost. (6.18) is a linear first order differential equation. Its solution is given by

$$A(s)\exp\left\{\int \frac{-2z}{1-s}ds\right\} = \int \left[\exp\left\{\int \frac{-2z}{1-s}ds\right\}\right] \frac{z}{(1-s)^2}ds + C,$$

i.e.,

$$(1-s)^{2z}A(s) = \frac{-z(1-s)^{2z-1}}{2z-1} + C \quad \text{if } z \neq \tfrac{1}{2}.$$

The boundary condition $A(0)=0$ determines the constant; $C=z/(2z-1)$ and

$$A(s) = \frac{z}{2z-1}\left(\frac{1}{(1-s)^{2z}} - \frac{1}{1-s}\right).$$

From this we obtain the generating function from the fact that the coefficient of s^n in the Taylor expansion of $A(s)$ equals $a_n(z)$, so

$$nE\left(z^{S_n}\right) = \frac{1}{n!}\frac{d^n}{ds^n}A(0).$$

Differentiating n times with respect to s,

$$A^{(n)}(s) = \\ \frac{z}{2z-1}\Big(2z(2z+1)\cdots(2z+n-1)(1-s)^{-2z-n} - n!(1-s)^{-(n+1)}\Big).$$

Hence, once again,

$$E\left(z^{S_n}\right) = \frac{z}{n!n}\,\frac{2z(2z+1)\cdots(2z+n-1) - n!}{2z-1}.$$

$\square$

6.3 Time-dependent requests

In Chapter 4 we considered Second User's cost of searching in a list maintained by the move-to-front rule. Assuming finitely many records ($n < \infty$) as in Section 4.1, suppose that Controller requests record i, $i = 1, \ldots, n$, at the time-points of a Poisson process with intensity ${}^t p_i(t^*)$, where at each instant t^* in backward time, the intensities satisfy $\sum_{i=1}^n {}^t p_i(t^*) = 1$. Second User can be thought of to request at time-points of other Poisson processes, but since we are only interested in the search cost conditional on that Second User makes a request at this instant, it suffices to keep in mind Second User's request probabilities ${}^t q_i, i = 1, \ldots, n$, at the time-point of the current request.

Let T_i be the (backward) waiting time to the first occurrence in Poisson process i, cf. Figure 4.1.1. Then

$$P(T_i > u^*) = \exp\left\{ - \int_0^{u^*} {}^t p_i(t^*) dt^* \right\}$$

and T_i's density

$$f_{T_i}(u^*) = {}^t p_i(u^*) \exp\left\{ - \int_0^{u^*} {}^t p_i(t^*) dt^* \right\}.$$

All the results on the search cost above may be generalized to the case of time-dependent request probabilities. As an illustration we carry this through for Second User's expected long-term search cost, that is, the cost when enough time has passed to ensure that the tree has 'forgotten' its initial shape.

Theorem 6.3.1 *Second User's expected long-term search cost is*

$$\begin{aligned} E\left(S^{long\text{-}term}(t)\right) & \\ = \ 1 + \sum_{1 \le i < j \le n} & \int_0^\infty [{}^t q_i \ {}^t p_j(u^*) + {}^t q_j \ {}^t p_i(u^*)] \\ & \times \exp\left\{ - \int_0^{u^*} ({}^t p_i(t^*) + \cdots + {}^t p_j(t^*)) dt^* \right\} du^*. \end{aligned}$$

Proof. Following the line of reasoning in the proof of Theorem 6.1.1.1, (i), we consider some $j \in \{1,\dots,n\}$ and condition on $T_j = u^*$. Recall that $\mathbf{1}_i^j(t)$ indicates whether j is an ancestor of i at time t.

The event that j is an ancestor of i splits into two cases:
If $j < i$, then analogous to (6.1),

$$\begin{aligned} P_{u^*}(\mathbf{1}_i^j(t) = 1) &= P(T_{j+1} > u^*)P(T_{j+2} > u^*)\cdots P(T_i > u^*) \\ &= \exp\left\{ -\int_0^{u^*} ({}^tp_{j+1}(t^*) + \cdots + {}^tp_i(t^*))dt^* \right\} \end{aligned} \tag{6.19}$$

and if $j > i$, analogous to (6.2),

$$\begin{aligned} P_{u^*}(\mathbf{1}_i^j(t) = 1) &= P(T_i > u^*)P(T_{i+1} > u^*)\cdots P(T_{j-1} > u^*) \\ &= \exp\left\{ -\int_0^{u^*} ({}^tp_i(t^*) + \cdots + {}^tp_{j-1}(t^*))dt^* \right\}. \end{aligned} \tag{6.20}$$

The expected long-term cost of searching record i at time t is

$$E\left(S_i^{long\text{-}term}(t)\right) = E\left(1 + \sum_{j\neq i} \mathbf{1}_i^j(t)\right) = 1 + \sum_{j \neq i} P(\mathbf{1}_i^j(t) = 1).$$

Hence by (6.19) and (6.20)

$$\begin{aligned} & E\left(S^{long\text{-}term}(t)\right) \\ = \;& \sum_i {}^tq_i E\left(S_i^{long\text{-}term}(t)\right) \\ = \;& 1 + \sum_i {}^tq_i \sum_{j \neq i} \int_0^\infty P_{u^*}(\mathbf{1}_i^j(t) = 1)\, {}^tp_j(u^*) \exp\left\{ -\int_0^{u^*} {}^tp_j(t^*)dt^* \right\} du^* \\ = \;& 1 + \sum_i {}^tq_i \\ & \times \Bigg[\sum_{j:j<i} \int_0^\infty \exp\left\{ -\int_0^{u^*} ({}^tp_j(t^*) + {}^tp_{j+1}(t^*) + \cdots + {}^tp_i(t^*))dt^* \right\} {}^tp_j(u^*)du^* \\ & + \sum_{j:j>i} \int_0^\infty \exp\left\{ -\int_0^{u^*} ({}^tp_i(t^*) + \cdots + {}^tp_{j-1}(t^*) + {}^tp_j(t^*))dt^* \right\} {}^tp_j(u^*)du^* \Bigg] \end{aligned}$$

$$\begin{aligned} &= 1 + \sum_{1 \le i < j \le n} \int_0^\infty [{}^tq_i \ {}^tp_j(u^*) + {}^tq_j \ {}^tp_i(u^*)] \\ &\qquad \times \exp\left\{ - \int_0^{u^*} ({}^tp_i(t^*) + \cdots + {}^tp_j(t^*)) dt^* \right\} du^*. \end{aligned}$$

□

Appendix A

With notation as in Section 6.1.1 we derive:

Theorem A.1 *For a tree, initially in shape τ, the variance of Second User's search cost at time t is*

$$
\begin{aligned}
&\mathrm{Var}\left(S^{\tau}(t)\right) = \left(E\left(S^{\tau}(t)\right) - 1\right)\left(2 - E\left(S^{\tau}(t)\right)\right) \\
&\quad + \quad 2 \sum_{1 \le i < j < k \le n} \Bigg\{ \frac{(p_i q_j + q_i p_j) p_k}{(p_i + \cdots + p_k)(p_i + \cdots + p_j)} \\
&\qquad + \frac{p_i (p_j q_k + q_j p_k)}{(p_i + \cdots + p_k)(p_j + \cdots + p_k)} \\
&\qquad + \Bigg[\frac{p_j}{p_i + \cdots + p_k} \Bigg(\frac{q_i p_k}{p_{j+1} + \cdots + p_k} + \frac{p_i q_k}{p_i + \cdots + p_{j-1}} \\
&\qquad - \frac{p_i q_j p_k}{(p_i + \cdots + p_{j-1})(p_{j+1} + \cdots + p_k)} \Bigg) \\
&\qquad + \mathbf{1}_i^j(0) \left(\mathbf{1}_j^k(0) - \frac{p_k}{p_{j+1} + \cdots + p_k} \right) q_i \\
&\qquad + \left(\mathbf{1}_j^i(0) \mathbf{1}_j^k(0) - \frac{\mathbf{1}_j^k(0) p_i}{p_i + \cdots + p_{j-1}} - \frac{\mathbf{1}_j^i(0) p_k}{p_{j+1} + \cdots + p_k} \right) q_j \\
&\qquad + \mathbf{1}_k^j(0) \left(\mathbf{1}_j^i(0) - \frac{p_i}{p_i + \cdots + p_{j-1}} \right) q_k \Bigg] \exp\{-(p_i + \cdots + p_k) t\} \\
&\qquad + \Bigg[\frac{p_j p_k}{(p_i + \cdots + p_j)(p_{j+1} + \cdots + p_k)} \left(\frac{p_i q_j}{p_i + \cdots + p_{j-1}} - q_i \right) \\
&\qquad + \frac{p_k}{p_{j+1} + \cdots + p_k} (q_i \mathbf{1}_i^j(0) + q_j \mathbf{1}_j^i(0)) \Bigg] \exp\{-(p_i + \cdots + p_j) t\} \\
&\qquad + \Bigg[\frac{p_i p_j}{(p_i + \cdots + p_{j-1})(p_j + \cdots + p_k)} \left(\frac{q_j p_k}{p_{j+1} + \cdots + p_k} - q_k \right)
\end{aligned}
$$

$$+\frac{p_i}{p_i+\cdots+p_{j-1}}(q_j\mathbf{1}_j^k(0)+q_k\mathbf{1}_k^j(0))\Big]\exp\{-(p_j+\cdots+p_k)t\}$$
$$+\frac{p_iq_jp_k}{(p_i+\cdots+p_{j-1})(p_{j+1}+\cdots+p_k)}\exp\{-p_jt\}\Bigg\},$$

with $(E(S^\tau(t)))$ *as in Theorem 6.1.1.1.*

The corresponding single-user result follows immediately when $q_i \equiv p_i$, $i=1,\dots,n$.

Corollary A.2 *For a tree, initially in shape* τ*, the variance of a single user's search cost at time* t *is*

$$\begin{aligned}
\operatorname{Var}(S^\tau(t)) &= (E(S^\tau(t))-1)\,(2-E(S^\tau(t)))\\
+\;\; 2 &\sum_{1\le i<j<k\le n}\Bigg\{2\frac{p_ip_jp_k}{p_i+\cdots+p_k}\left(\frac{1}{p_i+\cdots+p_j}+\frac{1}{p_j+\cdots+p_k}\right)\\
&+\frac{p_ip_jp_k}{(p_i+\cdots+p_{j-1})(p_{j+1}+\cdots+p_k)}\\
&\times\Bigg[\left(1-\frac{2p_j}{p_i+\cdots+p_k}\right)\exp\{-(p_i+\cdots+p_k)t\}\\
&+\left(\frac{2p_j}{p_i+\cdots+p_j}-1\right)\exp\{-(p_i+\cdots+p_j)t\}\\
&+\left(\frac{2p_j}{p_j+\cdots+p_k}-1\right)\exp\{-(p_j+\cdots+p_k)t\}+\exp\{-p_jt\}\Bigg]\\
&+\Bigg[\mathbf{1}_i^j(0)\mathbf{1}_j^k(0)p_i+\mathbf{1}_j^i(0)\mathbf{1}_j^k(0)p_j+\mathbf{1}_j^i(0)\mathbf{1}_k^j(0)p_k\\
&-\frac{p_i(\mathbf{1}_j^k(0)p_j+\mathbf{1}_k^j(0)p_k)}{p_i+\cdots+p_{j-1}}-\frac{p_i(\mathbf{1}_i^j(0)p_i+\mathbf{1}_j^i(0)p_j)p_k}{p_{j+1}+\cdots+p_k}\Bigg]\\
&\times\exp\{-(p_i+\cdots+p_k)t\}\\
&+\frac{(\mathbf{1}_i^j(0)p_i+\mathbf{1}_j^i(0)p_j)p_k}{p_{j+1}+\cdots+p_k}\exp\{-(p_i+\cdots+p_j)t\}\\
&+\frac{p_i(\mathbf{1}_j^k(0)p_j+\mathbf{1}_k^j(0)p_k)}{p_i+\cdots+p_{j-1}}\exp\{-(p_j+\cdots+p_k)t\}\Bigg\},
\end{aligned}$$

with $(E(S^\tau(t)))$ *as in Theorem 6.1.1.2.*

Proof of Theorem A.1. Using that $(\mathbf{1}_i^j(t))^2 = \mathbf{1}_i^j(t)$, we have

$$
\begin{aligned}
&E(S^\tau(t)^2)\\
&= \sum_i q_i E\Big[(1+\sum_{j:j\neq i}\mathbf{1}_i^j(t))^2\Big]\\
&= 1+\sum_i q_i E\Big[2\sum_{j:j\neq i}\mathbf{1}_i^j(t)+\sum_{j:j\neq i}(\mathbf{1}_i^j(t))^2+\sum_{j:j\neq i}\sum_{\substack{k:k\neq j\\ k\neq i}}\mathbf{1}_i^j(t)\mathbf{1}_i^k(t)\Big]\\
&= 1+3(E(S^\tau(t))-1)+2\sum_i q_i\sum_{j:j\neq i}\sum_{\substack{k:k>j\\ k\neq i}}P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t)=1).
\end{aligned}
$$

Rewriting

$$[E(S^\tau(t))]^2 = 1+2(E(S^\tau(t))-1)+(E(S^\tau(t))-1)^2$$

we get

$$
\begin{aligned}
\mathrm{Var}(S^\tau(t)) &= E(S^\tau(t)^2)-[E(S^\tau(t))]^2\\
&= (E(S^\tau(t))-1)(2-E(S^\tau(t)))\\
&\quad +2\sum_i q_i\sum_{j:j\neq i}\sum_{\substack{k:k>j\\ k\neq i}}P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t)=1),
\end{aligned}
$$

and by Lemma A.3, we have the result.

$\square$

Lemma A.3

$$
\begin{aligned}
&\sum_i q_i\sum_{j:j\neq i}\sum_{\substack{k:k>j\\ k\neq i}}P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t)=1)\\
&= \sum_{1\le i<j<k\le n}\Bigg\{\frac{(p_iq_j+q_ip_j)p_k}{(p_i+\cdots+p_k)(p_i+\cdots+p_j)}\\
&\quad +\frac{p_i(p_jq_k+q_jp_k)}{(p_i+\cdots+p_k)(p_j+\cdots+p_k)}\\
&\quad +\Bigg[\frac{p_j}{p_i+\cdots+p_k}\Bigg(\frac{q_ip_k}{p_{j+1}+\cdots+p_k}+\frac{p_iq_k}{p_i+\cdots+p_{j-1}}
\end{aligned}
$$

$$
\begin{aligned}
&-\frac{p_i q_j p_k}{(p_i+\cdots+p_{j-1})(p_{j+1}+\cdots+p_k)}\Bigg)\\
&+\mathbf{1}_i^j(0)\left(\mathbf{1}_j^k(0)-\frac{p_k}{p_{j+1}+\cdots+p_k}\right)q_i\\
&+\left(\mathbf{1}_j^i(0)\mathbf{1}_j^k(0)-\frac{\mathbf{1}_j^k(0)p_i}{p_i+\cdots+p_{j-1}}-\frac{\mathbf{1}_j^i(0)p_k}{p_{j+1}+\cdots+p_k}\right)q_j\\
&+\mathbf{1}_k^j(0)\left(\mathbf{1}_j^i(0)-\frac{p_i}{p_i+\cdots+p_{j-1}}\right)q_k\Bigg]\exp\{-(p_i+\cdots+p_k)t\}\\
&+\Bigg[\frac{p_j p_k}{(p_i+\cdots+p_j)(p_{j+1}+\cdots+p_k)}\left(\frac{p_i q_j}{p_i+\cdots+p_{j-1}}-q_i\right)\\
&+\frac{p_k}{p_{j+1}+\cdots+p_k}(q_i\mathbf{1}_i^j(0)+q_j\mathbf{1}_j^i(0))\Bigg]\exp\{-(p_i+\cdots+p_j)t\}\\
&+\Bigg[\frac{p_i p_j}{(p_i+\cdots+p_{j-1})(p_j+\cdots+p_k)}\left(\frac{q_j p_k}{p_{j+1}+\cdots+p_k}-q_k\right)\\
&+\frac{p_i}{p_i+\cdots+p_{j-1}}(q_j\mathbf{1}_j^k(0)+q_k\mathbf{1}_k^j(0))\Bigg]\exp\{-(p_j+\cdots+p_k)t\}\\
&+\frac{p_i q_j p_k}{(p_i+\cdots+p_{j-1})(p_{j+1}+\cdots+p_k)}\exp\{-p_j t\}\Bigg\}.
\end{aligned}
$$

Proof of Lemma A.3. Again, let L_i be the time elapsed since Controller's last request for i, $i = 1, \ldots, n$. Basically, there are three cases:

(i) If $j < k < i$, $\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1$ if and only if $\mathbf{1}_i^j(t) = \mathbf{1}_i^k(t) = 1$, that is, if and only if both j and k are ancestors of i at time t. Since $j < k$, the only way this can happen is if j is an ancestor of k which in turn is an ancestor of i. By Observation 5.2.1 we demand that Controller's last request for k has occurred after his resp. last requests for $k+1, \ldots, i$ and further that his last request for j has occurred after his last request for any of $j+1, \ldots, k$. (In other words, j must have been requested later than all the records $j+1, \ldots, i$). By the embedding this is equivalent to the following condition:

$$L_j < L_{j+1}, \cdots, L_j < L_k, L_k < L_{k+1}, \cdots, L_k < L_i.$$

Now condition with respect to $L_j = u_j^*$ and $L_k = u_k^*$.

(a) If $0 \leq u_j^* < t$ and $0 \leq u_k^* < t$, then Controller has requested j and k at least once each since the start. Using the independence implied by the Poisson embedding we have

$$\begin{aligned}
&P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1, 0 \leq L_j < t, 0 \leq L_k < t) \\
&= \int_0^t \int_{u_j^*}^t p_j e^{-p_j u_j^*} \exp\{-(p_{j+1} + \cdots + p_{k-1})u_j^*\} \\
&\qquad \times p_k e^{-p_k u_k^*} \exp\{-(p_{k+1} + \cdots + p_i)u_k^*\} du_k^* du_j^* \\
&= \frac{p_j p_k}{(p_k + \cdots + p_i)(p_j + \cdots + p_i)}(1 - \exp\{-(p_j + \cdots + p_i)t\}) \\
&\quad - \frac{p_j p_k \exp\{-(p_k + \cdots + p_i)t\}}{(p_k + \cdots + p_i)(p_j + \cdots + p_{k-1})}(1 - \exp\{-(p_j + \cdots + p_{k-1})t\}).
\end{aligned}$$

(b) If $u_j^* = t$ and $0 \leq u_k^* < t$, then Controller has requested k at least once each since the start but never j. In this case, j cannot possibly be an ancestor of k.

$$P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1, L_j = t, 0 \leq L_k < t) = 0.$$

(c) If $0 \leq u_j^* < t$ and $u_k^* = t$, then Controller has requested j at least once each since the start but never k. Hence, k must initially have been an ancestor of i and this relation must not have been destroyed, that is, Controller must not have requested any of $k+1, \ldots, i$. Furthermore, he must have requested j after his last request for any of $j+1, \ldots, k-1$. (Note that j is known to have been requested after k by the condition on u_j^* and u_k^*.)

$$\begin{aligned}
&P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1, 0 \leq L_j < t, L_k = t) \\
&= \mathbf{1}_i^k(0) e^{-p_k t} \exp\{-(p_{k+1} + \cdots + p_i)t\} \\
&\qquad \times \int_0^t p_j e^{-p_j u_j^*} \exp\{-(p_{j+1} + \cdots + p_{k-1})u_j^*\} du_j^* \\
&= \mathbf{1}_i^k(0) \exp\{-(p_k + \cdots + p_i)t\} \frac{p_j}{p_j + \cdots + p_{k-1}} \\
&\quad \times (1 - \exp\{-(p_j + \cdots + p_{k-1})t\}).
\end{aligned}$$

(d) If $u_j^* = u_k^* = t$, then Controller has requested neither j nor k. Hence, the ancestry must have been the desired one since the start, that is, j is an ancestor of k and k is an ancestor of i in the initial tree and these relations have not been ruined during the elapsed time t.

$$P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1, L_j = L_k = t) = \mathbf{1}_k^j(0)\mathbf{1}_i^k(0)\exp\{-(p_j + \cdots + p_i)t\}.$$

(ii) If $j < i < k$, then j and k are both ancestors of i if Controller has neither requested any of $j+1, \ldots, i$ after his last request for i nor any of $i, \ldots, k-1$ after his last request for k, that is, if and only if

$$L_j < L_{j+1}, \cdots, L_j < L_i, L_k < L_i, \cdots, L_k < L_{k-1}.$$

Conditioning on $L_i = u_i^*$, $L_j = u_j^*$ and $L_k = u_k^*$ we have:
(a) If $0 \le u_j^* < t$ and $0 \le u_k^* < t$,

$$\begin{aligned}
&P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1, 0 \le L_j < t, 0 \le L_k < t) \\
&= \int_0^t \int_0^{u_i^*} \int_0^{u_i^*} p_i e^{-p_i u_i^*} p_j e^{-p_j u_j^*} \exp\{-(p_{j+1} + \cdots + p_{i-1})u_j^*\} \\
&\qquad \times p_k e^{-p_k u_k^*} \exp\{-(p_{i+1} + \cdots + p_{k-1})u_k^*\} du_k^* du_j^* du_i^* \\
&= \cdots \\
&= \frac{p_j p_k}{p_j + \cdots + p_k}\left(\frac{1}{p_j + \cdots + p_i} + \frac{1}{p_i + \cdots + p_k}\right) \\
&\quad - \frac{p_i p_j p_k}{(p_j + \cdots + p_{i-1})(p_{i+1} + \cdots + p_k)} \\
&\quad \times \left[\frac{\exp\{-p_i t\}}{p_i} - \frac{\exp\{-(p_j + \cdots + p_i)t\}}{p_j + \cdots + p_i} - \frac{\exp\{-(p_i + \cdots + p_k)t\}}{p_i + \cdots + p_k}\right. \\
&\quad \left. + \frac{\exp\{-(p_j + \cdots + p_k)t\}}{p_j + \cdots + p_k}\right].
\end{aligned}$$

(b) If $u_j^* = t$ and $0 \le u_k^* < t$, then j must initially be an ancestor of i and Controller must not have requested any of $j+1, \ldots, i$ in order for the ancestry still to be intact. Furthermore, Controller's last request for k must have occurred later than those for $i+1, \ldots, k-1$. Hence, conditioning on $L_k = u_k^*$,

$$\begin{aligned}
&P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1, L_j = t, 0 \le L_k < t) \\
&\quad = \mathbf{1}_i^j(0)e^{-p_j t}\exp\{-(p_{j+1} + \cdots + p_i)t\} \\
&\qquad \times \int_0^t p_k e^{-p_k u_k^*}\exp\{-(p_{i+1} + \cdots + p_{k-1})u_k^*\}du_k^* \\
&\quad = \mathbf{1}_i^j(0)\exp\{-(p_j + \cdots + p_i)t\}\frac{p_k}{p_{i+1} + \cdots + p_k} \\
&\qquad \times(1 - \exp\{-(p_{i+1} + \cdots + p_k)t\}).
\end{aligned}$$

(c) If $0 \le u_j^* < t$ and $u_k^* = t$, then k must initially be an ancestor of i, this relationship must not have been ruined (no Controller request for $i, \ldots, k-1$ has occurred) and Controller's last request for j must have occurred later than his last request for any of $j+1, \ldots, i-1$.

$$\begin{aligned}
&P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1, 0 \le L_j < t, L_k = t) \\
&\quad = \mathbf{1}_i^k(0)e^{-p_k t}\exp\{-(p_i + \cdots + p_{k-1})t\} \\
&\qquad \times \int_0^t p_j e^{-p_j u_j^*}\exp\{-(p_{j+1} + \cdots + p_{i-1})u_j^*\}du_j^* \\
&\quad = \mathbf{1}_i^k(0)\exp\{-(p_i + \cdots + p_k)t\}\frac{p_j}{p_j + \cdots + p_{i-1}} \\
&\qquad \times(1 - \exp\{-(p_j + \cdots + p_{i-1})t\}).
\end{aligned}$$

(d) If $u_j^* = u_k^* = t$, then Controller has requested neither j nor k.

$$P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1, L_j = L_k = t) = \mathbf{1}_i^j(0)\mathbf{1}_i^k(0)\exp\{-(p_j + \cdots + p_k)t\}.$$

(iii) If $i < j < k$, we have a case similar to (i). This time we demand:

$$L_k < L_j, \cdots, L_k < L_{k-1}, L_j < L_i, \cdots, L_j < L_{j-1}.$$

(a) If $0 \le u_j^* < t$ and $0 \le u_k^* < t$, we have

$$\begin{aligned}
&P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t) = 1, 0 \le L_j < t, 0 \le L_k < t) \\
&\quad = \int_0^t \int_{u_k^*}^t p_k e^{-p_k u_k^*}\exp\{-(p_{j+1} + \cdots + p_{k-1})u_k^*\} \\
&\qquad \times p_j e^{-p_j u_j^*}\exp\{-(p_i + \cdots + p_{j-1})u_j^*\}du_j^* du_k^*
\end{aligned}$$

$$\begin{aligned}
&= \frac{p_j p_k}{(p_i+\cdots+p_j)(p_i+\cdots+p_k)}\Big(1-\exp\{-(p_i+\cdots+p_k)t\}\Big)\\
&\quad -\frac{p_j p_k \exp\{-(p_i+\cdots+p_j)t\}}{(p_i+\cdots+p_j)(p_{j+1}+\cdots+p_k)}\Big(1-\exp\{-(p_{j+1}+\cdots+p_k)t\}\Big).
\end{aligned}$$

(b) If $u_j^* = t$ and $0 \le u_k^* < t$, then

$$\begin{aligned}
&P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t)=1, L_j=t, 0\le L_k<t)\\
&= \mathbf{1}_i^j(0)\exp\{-(p_i+\cdots+p_j)t\}\frac{p_k}{p_{j+1}+\cdots+p_k}\\
&\quad \times(1-\exp\{-(p_{j+1}+\cdots+p_k)t\}).
\end{aligned}$$

(c) If $0 \le u_j^* < t$ and $u_k^* = t$, then k cannot possibly be an ancestor of j.

$$P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t)=1, 0\le L_j<t, L_k=t)=0.$$

(d) If $u_j^* = u_k^* = t$, then

$$P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t)=1, L_j=L_k=t) = \mathbf{1}_j^k(0)\mathbf{1}_i^j(0)\exp\{-(p_i+\cdots+p_k)t\}.$$

Using (i)-(iii) we get the quantity in question.

$$\begin{aligned}
&\sum_i q_i \sum_{j:j\ne i}\sum_{\substack{k:k>j\\ k\ne i}} P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t)=1)\\
&= \sum_i q_i \sum_{j:j<i}\sum_{\substack{k:k>j\\ k<i}}\left\{\frac{p_j p_k}{(p_k+\cdots+p_i)(p_j+\cdots+p_i)}\right.\\
&\qquad \times(1-\exp\{-(p_j+\cdots+p_i)t\})\\
&\qquad -\frac{p_j p_k\exp\{-(p_k+\cdots+p_i)t\}}{(p_k+\cdots+p_i)(p_j+\cdots+p_{k-1})}\\
&\qquad \times(1-\exp\{-(p_j+\cdots+p_{k-1})t\})\\
&\qquad +\mathbf{1}_i^k(0)\exp\{-(p_k+\cdots+p_i)t\}\frac{p_j}{p_j+\cdots+p_{k-1}}\\
&\qquad \times(1-\exp\{-(p_j+\cdots+p_{k-1})t\})\\
&\qquad \left.+\mathbf{1}_k^j(0)\mathbf{1}_i^k(0)\exp\{-(p_j+\cdots+p_i)t\}\right\}
\end{aligned}$$

$$
\begin{aligned}
+ \quad & \sum_i q_i \sum_{j:j<i} \sum_{k:k>i} \Biggl\{ \frac{p_j p_k}{p_j + \cdots + p_k} \left(\frac{1}{p_j + \cdots + p_i} + \frac{1}{p_i + \cdots + p_k} \right) \\
& -\frac{p_i p_j p_k}{(p_j + \cdots + p_{i-1})(p_{i+1} + \cdots + p_k)} \\
& \times \biggl[\frac{\exp\{-p_i t\}}{p_i} - \frac{\exp\{-(p_j + \cdots + p_i)t\}}{p_j + \cdots + p_i} \\
& - \frac{\exp\{-(p_i + \cdots + p_k)t\}}{p_i + \cdots + p_k} \\
& + \frac{\exp\{-(p_j + \cdots + p_k)t\}}{p_j + \cdots + p_k} \biggr] \\
& + \mathbf{1}_i^j(0) \exp\{-(p_j + \cdots + p_i)t\} \frac{p_k}{p_{i+1} + \cdots + p_k} \\
& \times (1 - \exp\{-(p_{i+1} + \cdots + p_k)t\}) \\
& + \mathbf{1}_i^k(0) \exp\{-(p_i + \cdots + p_k)t\} \frac{p_j}{p_j + \cdots + p_{i-1}} \\
& \times (1 - \exp\{-(p_j + \cdots + p_{i-1})t\}) \\
& + \mathbf{1}_i^j(0) \mathbf{1}_i^k(0) \exp\{-(p_j + \cdots + p_k)t\} \Biggr\} \\
+ \quad & \sum_i q_i \sum_{j:j>i} \sum_{k:k>j} \Biggl\{ \frac{p_j p_k}{(p_i + \cdots + p_j)(p_i + \cdots + p_k)} \\
& \times \Bigl(1 - \exp\{-(p_i + \cdots + p_k)t\} \Bigr) \\
& - \frac{p_j p_k \exp\{-(p_i + \cdots + p_j)t\}}{(p_i + \cdots + p_j)(p_{j+1} + \cdots + p_k)} \\
& \times \Bigl(1 - \exp\{-(p_{j+1} + \cdots + p_k)t\} \Bigr) \\
& + \mathbf{1}_i^j(0) \exp\{-(p_i + \cdots + p_j)t\} \frac{p_k}{p_{j+1} + \cdots + p_k} \\
& \times (1 - \exp\{-(p_{j+1} + \cdots + p_k)t\}) \\
& + \mathbf{1}_j^k(0) \mathbf{1}_i^j(0) \exp\{-(p_i + \cdots + p_k)t\} \Biggr\}.
\end{aligned}
$$

Changing indices in the first two sums, we have

$$
\begin{aligned}
&\sum_i q_i \sum_{j:j\neq i} \sum_{\substack{k:k>j\\ k\neq i}} P(\mathbf{1}_i^j(t)\mathbf{1}_i^k(t)=1)\\
&= \sum_{1\le i<j<k\le n} \Bigg\{ \frac{(p_iq_j+q_ip_j)p_k}{(p_i+\cdots+p_k)(p_i+\cdots+p_j)}\\
&\quad + \frac{p_i(p_jq_k+q_jp_k)}{(p_i+\cdots+p_k)(p_j+\cdots+p_k)}\\
&\quad + \Bigg[\frac{p_j}{p_i+\cdots+p_k}\Bigg(\frac{q_ip_k}{p_{j+1}+\cdots+p_k}+\frac{p_iq_k}{p_i+\cdots+p_{j-1}}\\
&\quad - \frac{p_iq_jp_k}{(p_i+\cdots+p_{j-1})(p_{j+1}+\cdots+p_k)}\Bigg)\\
&\quad + \mathbf{1}_i^j(0)\left(\mathbf{1}_j^k(0)-\frac{p_k}{p_{j+1}+\cdots+p_k}\right)q_i\\
&\quad + \left(\mathbf{1}_j^i(0)\mathbf{1}_j^k(0)-\frac{\mathbf{1}_j^k(0)p_i}{p_i+\cdots+p_{j-1}}-\frac{\mathbf{1}_j^i(0)p_k}{p_{j+1}+\cdots+p_k}\right)q_j\\
&\quad + \mathbf{1}_k^j(0)\left(\mathbf{1}_j^i(0)-\frac{p_i}{p_i+\cdots+p_{j-1}}\right)q_k\Bigg]\exp\{-(p_i+\cdots+p_k)t\}\\
&\quad + \Bigg[\frac{p_jp_k}{(p_i+\cdots+p_j)(p_{j+1}+\cdots+p_k)}\left(\frac{p_iq_j}{p_i+\cdots+p_{j-1}}-q_i\right)\\
&\quad + \frac{p_k}{p_{j+1}+\cdots+p_k}(q_i\mathbf{1}_i^j(0)+q_j\mathbf{1}_j^i(0))\Bigg]\exp\{-(p_i+\cdots+p_j)t\}\\
&\quad + \Bigg[\frac{p_ip_j}{(p_i+\cdots+p_{j-1})(p_j+\cdots+p_k)}\left(\frac{q_jp_k}{p_{j+1}+\cdots+p_k}-q_k\right)\\
&\quad + \frac{p_i}{p_i+\cdots+p_{j-1}}(q_j\mathbf{1}_j^k(0)+q_k\mathbf{1}_k^j(0))\Bigg]\exp\{-(p_j+\cdots+p_k)t\}\\
&\quad + \frac{p_iq_jp_k}{(p_i+\cdots+p_{j-1})(p_{j+1}+\cdots+p_k)}\exp\{-p_jt\}\Bigg\}.
\end{aligned}
$$

□

Appendix B

Differentiating under the integral in the proof of Theorem 4.5.1 is allowed by the following Theorem due to Lebesgue.

Let μ be a measure defined on the measurable space $(\Omega, \mathcal{F})$ and let Λ be an open subset of $\mathbb{R}$. Consider the function $f : \Lambda \times \Omega \mapsto \mathbb{R}$ and suppose that for every $\lambda \in \Lambda$, the function $\omega \mapsto f(\lambda, \omega)$ is μ-integrable over Ω. Let

$$F(\lambda) := \int_\Omega f(\lambda, \omega) d\mu(\omega).$$

Theorem B.1 (Lebesgue) *Suppose that*

(i) For μ-almost every $\omega \in \Omega$, the function $\lambda \mapsto f(\lambda, \omega)$ is differentiable over Λ and for every $\lambda \in \Lambda$, the function $\omega \mapsto \frac{\partial f}{\partial \lambda}(\lambda, \omega)$ is μ-integrable over Ω.

(ii) There exists a μ-integrable real-valued function $h(\cdot)$ such that

$$\sup_{\lambda \in \Lambda} \left| \frac{\partial f}{\partial \lambda}(\lambda, \omega) \right| \leq |h(\omega)|$$

for μ-almost every $\omega \in \Omega$.

Then, the function $F(\cdot)$ is differentiable over Λ for every $\lambda \in \Lambda$;

$$\frac{\partial}{\partial \lambda} \int_\Omega f(\lambda, \omega) d\mu(\omega) = \int_\Omega \frac{\partial f}{\partial \lambda}(\lambda, \omega) d\mu(\omega).$$

Proof. Let $(\lambda_n, n \in \mathbb{N})$ be a sequence in Λ such that $\lambda_n \to \lambda$.
For $\lambda_n \neq \lambda$ we have

$$\frac{F(\lambda_n) - F(\lambda)}{\lambda_n - \lambda} = \int_\Omega \frac{f(\lambda_n, \omega) - f(\lambda, \omega)}{\lambda_n - \lambda} d\mu(\omega).$$

We know by (i) that for μ-almost every $\omega \in \Omega$,

$$\tilde{f}_n(\lambda_n, \lambda, \omega) := \frac{f(\lambda_n, \omega) - f(\lambda, \omega)}{\lambda_n - \lambda} \longrightarrow \frac{\partial f}{\partial \lambda}(\lambda, \omega) =: \tilde{f}(\lambda, \omega).$$

On the other hand, by Lagrange's mean value theorem, there is a $\xi \in (\lambda_n, \lambda)$ such that

$$|\tilde{f}_n(\lambda_n, \lambda, \omega)| = \left| \frac{f(\lambda_n, \omega) - f(\lambda, \omega)}{\lambda_n - \lambda} \right| = \left| \frac{\partial f}{\partial \lambda}(\xi, \omega) \right|$$

and by (ii),

$$\left| \frac{\partial f}{\partial \lambda}(\xi, \omega) \right| \le \sup_{\eta \in \Lambda} \left| \frac{\partial f}{\partial \lambda}(\eta, \omega) \right| \le |h(\omega)|.$$

Hence by dominated convergence,

$$\int_{\Omega} \tilde{f}_n(\lambda_n, \lambda, \omega) d\mu(\omega) \longrightarrow \int_{\Omega} \tilde{f}(\lambda, \omega) d\mu(\omega),$$

i.e.,

$$\frac{\partial F}{\partial \lambda}(\lambda) = \int_{\Omega} \frac{\partial f}{\partial \lambda}(\lambda, \omega) d\mu(\omega).$$

□

Bibliography

[1] Allen, B. and Munro, I. (1978). Self-organizing search trees. *Journal of the Association for Computing Machinery* **25**, 526–535.

[2] d'Amore, F., Marchetti-Spaccamela, A. and Nanni, U. (1993). The weighted list update problem and the lazy adversary. *Theoretical Computer Science* **108**, 371–384.

[3] Anderson, E. J., Nash, P. and Weber, R. R. (1982). A counterexample to a conjecture on optimal list ordering. *Journal of Applied Probability* **19**, 730–732.

[4] Bentley, J. L. and McGeoch, C. (1985). Amortized analysis of self-organizing sequential search heuristics. *Communications of the Association for Computing Machinery* **28**, 404–411.

[5] Bitner, J. R. (1979). Heuristics that dynamically organize data structures. *SIAM Journal on Computing* **8**, 82–110.

[6] Blom, G. and Holst, L. (1991). Embedding procedures for discrete problems in probability. *The Mathematical Scientist* **16**, 29–40.

[7] Bodell, J. (1995) *Aspects of the search time in binary search trees modified according to the move-to-root heuristic.* Licentiate thesis, TRITA/MAT - 94/0048, Department of Mathematics, Royal Institute of Technology, Stockholm.

[8] Bodell, J. (1996) Populärt om populariteter. *Nämnaren* **3**, 34–38.

[9] Booth A. D. and Colin A. J. (1960). On the efficiency of a new method of dictionary construction. *Information and Control* **3**, 327–334.

[10] Brown, G. G. and Shubert, B. O. (1984). On random binary trees. *Mathematics of Operations Research* **9**, 43–65.

[11] Burville, P. J. and Kingman, J. F. C. (1973). On a model for storage and search. *Journal of Applied Probability* **10**, 697–701.

[12] Chassaing, P. (1993). Optimality of move-to-front for self-organizing data structures with locality of references. *Annals of Applied Probability* **3**, 1219–1240.

[13] Chung, F. R. K., Hajela, D. J. and Seymour, P. D. (1988). Self-organizing sequential search and Hilbert's inequalities. *Journal of Computer and System Sciences* **36**, 148–157.

[14] Diaconis, P., Fill, J. A. and Pitman, J. (1992). Analysis of top to random shuffles. *Combinatorics, Probability and Computing* **1**, 135–155.

[15] Dobrow, R. P. and Fill, J. A. (1995). The move-to-front rule for self-organizing lists with Markov dependent requests. *Discrete Probability and Algorithms*, 57–80, *The IMA Volumes in Mathematics and its Applications* **72**, Springer, New York.

[16] Dobrow, R. P. and Fill, J. A. (1995). On the Markov chain for the move-to-root rule for binary search trees. *Annals of Applied Probability* **5**, 1–19.

[17] Dobrow, R. P. and Fill, J. A. (1995). Rates of convergence for the move-to-root Markov chain for binary search trees. *Annals of Applied Probability* **5**, 20–36.

[18] Donnelly, P. (1991). The heaps process, libraries, and size-biased permutations. *Journal of Applied Probability* **28**, 321–335.

[19] Ewens, W. J. (1990). Population genetics theory — the past and the future. In *Mathematical and Statistical Developments of Evolutionary Theory*, ed. S. Lessard, Kluwer, Dordrecht.

[20] Fill, J. A. (1996). On the distribution of binary search trees under the random permutation model. *Random Structures and Algorithms* **8**, 1–25.

[21] Fill, J. A. (1996). An exact formula for the move-to-front rule for self-organizing lists. *Journal of Theoretical Probability* **9**, 113–160.

[22] Fill, J. A. (1997). Limits and rates of convergence for the distribution of search cost under the move-to-front rule. *Theoretical Computer Science A*, **171**. To appear.

[23] Fill, J. A. and Holst, L. (1996). On the distribution of search cost for the move-to-front rule. *Random Structures and Algorithms* **8**, 179–186.

[24] Flajolet, P., Gardy, D. and Thimonier, L. (1992). Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics* **39**, 207–229.

[25] Flajolet, P., Gardy, D. and Thimonier, L. (1988). Random allocations and probabilistic languages. *Automata, languages and programming* (Tampere, 1988), 239–253, Lecture Notes in Computer Science, **317**, Springer, Berlin-New York, 1988. Analysis of demand paging algorithms, in Proceedings IFIP 1971 Congress, Ljubljana, **1**, 485–490, North-Holland, Amsterdam.

(*Note*: This is the same paper as [24].)

[26] Fletcher, R. (1987). *Practical Methods of Optimization*, Wiley, New York.

[27] Gill, P.E., Murray, W., Saunders M.A. and Wright, M. (1986). User's guide for *NPSOL (Version 4.0)*: a *Fortran* package for nonlinear programming. Report number SOL 86–2, Department of Operations Research, Stanford University.

[28] Gill, P. E., Murray, W. and Wright M. (1981). *Practical Optimization*, Academic Press, New York.

[29] Golin, M. J. (1994). A provably fast linear-expected-time maxima-finding algorithm. *Algorithmica*, **11** 501–524.

[30] Gonnet, G. H., Munro, J. I. and Suwanda, H. (1981). Exegesis of self-organizing linear search. *SIAM Journal on Computing* **10**, 613–637.

[31] Hendricks, W. J. (1972). The stationary distribution of an interesting Markov chain. *Journal of Applied Probability* **9**, 231–233.

[32] Hendricks, W. J. (1976). An account of self-organizing systems. *SIAM Journal on Computing* **5**, 715–723.

[33] Hester, J. H. and Hirschberg, D. S. (1985). Self-organizing linear search. *Association for Computing Machinery, Computer Surveys* **19**, 261–296.

[34] Hibbard T. N. (1962). Some combinatorial properties of certain trees with applications to searching and sorting. *Journal of the Association for Computing Machinery* **9**, 13-28.

[35] Hofri, M. and Shachnai, H. (1991). On the optimality of the counter scheme for dynamic linear lists. *Information Processing Letters* **37**, 175–179.

[36] Hofri, M. and Shachnai, H. (1991). Self-organizing lists and independent references: A statistical synergy. *Journal of Algorithms* **12**, 533–555.

[37] Holst, L. (1981). On sequential occupancy problems. *Journal of Applied Probability* **18**, 435–442.

[38] Irani, S. (1991). Two results on the list update problem. *Information Processing Letters* **38**, 301–306.

[39] Kan, Y. C. and Ross, S. M. (1980). Optimal list order under partial memory constraints. *Journal of Applied Probability* **17**, 1004–1015.

[40] Kapoor, S. and Reingold, E. M. (1991). Stochastic rearrangement rules for self-organizing data structures. *Algorithmica* **6**, 278–291.

[41] King, W. F. (1971). Analysis of demand paging algorithms, in Proceedings IFIP 1971 Congress, Ljubljana, **1**, 485–490, North-Holland, Amsterdam.

[42] Kingman, J. F. C. (1975). Random discrete distributions. *Journal of the Royal Statistical Society*, **B37**, 1–22.

[43] Knuth, D. E. (1973). *The Art of Computer Programming, vol. 2 Sorting and searching.* Addison-Wesley, Reading, Mass.

[44] Knuth, D. E. (1992). Two notes on notation. *The American Mathematical Monthly* **99**, 403–422.

[45] Konneker, L. K. and Varol, Y. L. (1981). A note on heuristics for dynamic organization of data structures. *Information Processing Letters* **12**, 213–216.

[46] Lam, K. (1984). Comparison of self-organizing linear search rules. *Journal of Applied Probability* **21**, 763–776.

[47] Lam, K., Leung, M. and Siu, M. (1984). Self-organizing files with dependent access. *Journal of Applied Probability* **21**, 343–359.

[48] Letac, G. (1974). Transience and recurrence of an interesting Markov chain. *Journal of Applied Probability* **11**, 818–824.

[49] Li, H. and Shaked, M. (1993). Stochastic majorization of stochastically monotone families. *Advances in Applied Probability* **25**, 895–913.

[50] Lindvall, T. (1992). *Lectures on the Coupling Method.* Wiley, New York.

[51] Louchard, G. (1987). Exact and asymptotic distributions in digital and binary search trees. *Theoretical Informatics and Applications* **21**, 479–496.

[52] Mahmoud, H. M. (1992). *Evolution of Random Search Trees.* Wiley, New York.

[53] Mahmoud, H. M. and Pittel, B. (1988). Analysis of the space of search trees under the random insertion algorithm. *Journal of Algorithms* **9**, 1–24.

[54] Marshall, A. W. and Olkin, I. (1979). *Inequalities: Theory of Majorization and its Applications.* Academic Press, New York.

[55] Matthews, D., Rotem, D. and Bretholz, E. (1980). Self-organizing doubly linked lists. *International Journal of Computer Mathematics* **8**, 99–106.

[56] Mauchly, J. (1946). *Theory and Techniques for the Design of Electronic Digital Computers* ed. by G. W. Patterson, **1** 9.7-9.8; **3**, 22.1-22.20.

[57] McCabe, J. (1965). On serial files with relocatable records. *Operations Research* **13**, 609–618.

[58] Mäkinen, E. (1987). On top-down splaying. *BIT* **27**, 330–339.

[59] Nelson, P. R. (1977). Single-shelf library-type Markov chains with infinitely many books. *Journal of Applied Probability* **14**, 298–308.

[60] Oommen, B. J. and Hansen, E. R. (1987). List organizing strategies using stochastic move-to-front and stochastic move-to-rear operations. *SIAM Journal on Computing* **16**, 705–716.

[61] Oommen, B. J. and Hansen, E. R. (1986). Expedient stochastic move-to-front and optimal stochastic move-to-rear list organizing strategies. ICDT '86 Proceedings of the international conference on database theory held in Rome, September 8–10, 1986, edited by G. Ausiello and P. Atzeni, 249–364. Lecture Notes in Computer Science, Springer Berlin.

[62] Phatarfod, R. M. (1991). On the matrix occurring in a linear search problem. *Journal of Applied Probability* **28**, 336–346.

[63] Phatarfod, R. M. (1994). On the transition probabilities of the move-to-front scheme. *Journal of Applied Probability* **31**, 570–574.

[64] Phatarfod, R. M. and Dyte, D. (1993). The linear search problem with Markov dependent requests. Preprint.

[65] Rivest, R. (1976). On self-organizing sequential search heuristics. *Communications of the Association for Computing Machinery* **19**, 63–67.

[66] Rodrigues, E. R. (1995). The performance of the move-to-front scheme under some particular forms of Markov requests. *Journal of Applied Probability* **32**, 1089–1102.

[67] Rodrigues, E. R. (1995). Convergence to stationary state for a Markov move-to-front scheme. *Journal of Applied Probability* **32**, 768–776.

[68] Sedgewick, R. (1988). *Algorithms*, 2nd ed. Addison-Wesley, Reading, MA.

[69] Schoenmakers, B. (1993). A systematic analysis of splaying. *Information Processing Letters* **45**, 41–50.

[70] Sleator, D. D. and Tarjan, R. E. (1985). Amortized efficiency of list update and paging rules. *Communications of the Association for Computing Machinery* **28**, 202-208.

[71] Sleator, D. D. and Tarjan, R. E. (1985). Self-adjusting binary search trees. *Journal of the Association for Computing Machinery* **32**, 652–686.

[72] Stephenson, C. J. (1980). A method for constructing binary search trees by making insertions at the root. *International Journal of Computer and Information Sciences* **9**, 15–29.

[73] Tenenbaum, A. (1978). Simulations of dynamic sequential search algorithms. *Communications of the Association for Computing Machinery* **21**, 790–791.

[74] Tenenbaum, A. and Nemes, R. M. (1982). Two spectra of self-organizing sequential search algorithms. *SIAM Journal on Computing* **11**, 557–566.

[75] Tsetlin, M. L. (1963). Finite automata and models of simple forms of behaviour. *Russian Mathematical Surveys* **18** (**4**), 1–27.

[76] Valiveti, R. S. and Oommen, B. J. (1993). Self-organizing doubly linked lists. *Journal of Algorithms* **14**, 88–114.

[77] Valiveti, R. S., Oommen, B. J. and Zgierski, J. R. (1995). Adaptive linear list reorganization under a generalized query system. *Journal of Applied Probability* **32**, 793–804.

[78] Windley, P. F. (1960 - 61). Trees, forests and rearranging. *The Computer Journal* **3**, 84-88.

Notation

A_x^π	set of real numbers $y \in (0,1]$ such that record $\lceil ny \rceil$ is after record $\lceil nx \rceil$ in the initial list arrangement
$Bern(p)$	Bernoulli distribution with success probability p
C_n	$\frac{1}{n+1}\binom{2n}{n}$, the Catalan numbers
$d_\pi(t)$	Kolmogorov-Smirnov distance between the cost distribution at time t for a list started in permutation π and one in stationarity
$d_{id}(t)$	dito with $\pi = id$
$d_{rev}(t)$	dito with $\pi = rev$
$\tilde{d}(t)$	Kolmogorov-Smirnov distance for the search cost distributions for move-to-front lists started in permutations rev resp. id
$\hat{d}(t)$	$\max_\pi d_\pi(t)$
δ_0	one-point distribution in 0
$\Delta_{id}(t)$	$E(S^{stat}(t)) - E(S^{id}(t))$
$\Delta_{rev}(t)$	$E(S^{rev}(t)) - E(S^{stat}(t))$
$G_n(\beta)$	$\sum_{i=1}^n i^\beta$
$H(c)$	cf. p. 71
$h(c)$	cf. p. 71
H_n	$\sum_{i=1}^n i^{-1}$; the n:th harmonic number
$H_n(\alpha)$	$\sum_{i=1}^n i^{-\alpha}$
	In particular:
$H_n(2)$	$\sum_{i=1}^n i^{-2}$; the n:th harmonic number of order 2
id	$(1, 2, \ldots, n)$; the identity permutation
$\mathbf{I}(A)$	indicator indicating whether A is true
$\mathbf{1}_{ij}(t)$	indicator indicating whether record i precedes record j in a move-to-front list at time t

$\mathbf{1}_i^j(t)$	indicator indicating whether record j is an ancestor of record i in a move-to-root binary search tree at time t
$\mathbf{1}_i^j$	indicator indicating whether record j is an ancestor of record i in a move-to-root binary search tree in stationarity
$\mathcal{L}$	distribution/'law'
L_k	$\min\{T_k, t\}$; time elapsed since Controller last requested record k, taken into account that the clock was started time t ago
${}^tL_k$	$\min\{{}^tT_k, t\}$; time elapsed since Controller last requested record k, taken into account that the clock was started time t ago
${}^{nt}L_k$	$\min\{{}^{nt}T_k, t\}$; rescaled time elapsed since Controller last requested record k, taken into account that the clock was started time nt (equivalent to rescaled time t) ago
n	number of records
$N(c)$	cf. p. 71
$\mathbb{N}$	natural numbers
$o(\cdot)$	little o; $f(x) = o(g(x))$ as $x \to a$ means $f(x)/g(x) \to 0$ as $x \to a$
p_k	probability that on a request, Controller wants record k
${}^tp_k(t^*)$	intensity at backward-time t^* of a Poisson process according to which Controller requests record k
${}^{nt}p_k(t^*, n)$	intensity at backward-time nt^* of a Poisson process according to which Controller requests record k in a list with n records
${}^tp(\cdot, \cdot)$	intensity of a two-dimensional Poisson process governing Controller's requests
π	initial permutation of the move-to-front list
$\pi(k)$	the label of the kth record (and its corresponding key) in a list in permutation π
q_k	probability that on a request, Second User wants record k
${}^{nt}q_k(n)$	probability that on a request at time nt, Second User requests record k in a list with n records
${}^tq(\cdot)$	Second User's request density at time t
${}^tQ(\cdot)$	distribution function corresponding to density ${}^tq(\cdot)$
rev	$(n, n-1, \ldots, 1)$; the reversed permutation
$\mathbb{R}$	real numbers
$S(t)$	Second User's search cost at time t
$S_{k,m}$	Second User's search cost in a tree consisting of the records $k, \ldots, m$

S_k	search cost in a tree consisting of k records, each having equal probability of being requested
$S^{\pi}(t)$	Second User's search cost at time t when the move-to-front list was initially in permutation π
$S^{\pi}_k(t)$	Second User's cost of searching record k at time t when the move-to-front list was initially in permutation π
$S^{\tau}(t)$	Second User's search cost at time t when the move-to-root tree was initially in shape τ
$S^{\tau}_k(t)$	Second User's cost of searching record k at time t when the move-to-root tree was initially in shape τ
S^{stat}	Second User's stationary search cost
$S^{stat}(t)$	Second User's stationary search cost
$S^{long\text{-}term}(t)$	Second User's long-term search cost at time t when the request probabilities change in time
$S^{long\text{-}term}_k(t)$	Second User's long-term cost of searching record k at time t when the request probabilities change in time
t	instant in ordinary /forward/ time
t^*	instant in (possibly rescaled) backward time
$t(n,c)$	cf. p. 71
T_k	backward waiting time until first occurrence in the Poisson process corresponding to Controller's requests for record k
${}^tT_k$	backward waiting time until first occurrence in the Poisson process corresponding to requests by Controller for record k when the records' popularities change in time
${}^{nt}T_k$	rescaled backward time until first occurrence in the Poisson process, 'slice', corresponding to requests by Controller for record k when the records' popularities change in time
T'	set of future time-points of Second User requests
$TV^{\pi}(t)$	$\sup_{t'\geq t}\max_A \lvert P(S^{\pi}(t')\in A)-P(S^{stat}(t')\in A)\rvert$
$TV(t)$	$\max_{\pi} TV^{\pi}(t)$
$\lfloor\cdot\rfloor$	integer part of $\cdot$, the largest integer $m\leq\cdot$
$\lceil\cdot\rceil$	rounding up of $\cdot$, the smallest integer $m\geq\cdot$
$\oplus$	distribution of sum, cf. p. 50
$*$	superindex indicating backward time

Index

Memory is a deadly enemy to mathematics, and though it has certain advantages, arithmetically speaking, memory is a bad thing!

The Lesson by Eugene Ionesco