The Dependent Doors Problem: An Investigation into Sequential Decisions without Feedback[*][†]

Amos Korman[1] **and Yoav Rodeh**[2]

1 **CNRS and University Paris Diderot, Paris, France**
amos.korman@irif.fr

2 **Weizmann Institute of Science, Rehovot, Israel**
yoav.rodeh@gmail.com

Abstract

We introduce the *dependent doors problem* as an abstraction for situations in which one must perform a sequence of possibly dependent decisions, without receiving feedback information on the effectiveness of previously made actions. Informally, the problem considers a set of d doors that are initially closed, and the aim is to open all of them as fast as possible. To open a door, the algorithm knocks on it and it might open or not according to some probability distribution. This distribution may depend on which other doors are currently open, as well as on which other doors were open during each of the previous knocks on that door. The algorithm aims to minimize the expected time until all doors open. Crucially, it must act at any time without knowing whether or which other doors have already opened. In this work, we focus on scenarios where dependencies between doors are both positively correlated and acyclic.

The fundamental distribution of a door describes the probability it opens in the best of conditions (with respect to other doors being open or closed). We show that if in two configurations of d doors corresponding doors share the same fundamental distribution, then these configurations have the same optimal running time up to a universal constant, no matter what are the dependencies between doors and what are the distributions. We also identify algorithms that are optimal up to a universal constant factor. For the case in which all doors share the same fundamental distribution we additionally provide a simpler algorithm, and a formula to calculate its running time. We furthermore analyse the price of lacking feedback for several configurations governed by standard fundamental distributions. In particular, we show that the price is logarithmic in d for memoryless doors, but can potentially grow to be linear in d for other distributions.

We then turn our attention to investigate precise bounds. Even for the case of two doors, identifying the optimal sequence is an intriguing combinatorial question. Here, we study the case of two cascading memoryless doors. That is, the first door opens on each knock independently with probability p_1. The second door can only open if the first door is open, in which case it will open on each knock independently with probability p_2. We solve this problem almost completely by identifying algorithms that are optimal up to an additive term of 1.

1998 ACM Subject Classification F.1.2 Modes of Computation, F.2.2 Sequencing and Scheduling

Keywords and phrases No Feedback, Sequential Decisions, Probabilistic Environment, Exploration and Exploitation, Golden Ratio

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.81

[*] The full version of this paper appears in https://arxiv.org/abs/1704.06096.
[†] This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 648032).

1 Introduction

Often it is the case that one must accomplish multiple tasks whose success probabilities are dependent on each other. In many cases, failure to achieve one task will tend to have a more negative affect on the success probabilities of other tasks. In general, such dependencies may be quite complex, and balancing the work load between different tasks becomes a computational challenge. The situation is further complicated if the ability to detect whether a task has been accomplished is limited. For example, if task B highly depends on task A then until A is accomplished, all efforts invested in B may be completely wasted. How should one divide the effort between these tasks if feedback on the success of A is not available?

In this preliminary work we propose a setting that captures some of the fundamental challenges that are inherent to the process of decision making without feedback. We introduce the *dependent doors* problem, informally described as follows. There are $d \geq 2$ doors (representing tasks) which are initially closed, and the aim is to open all of them as fast as possible. To open a door, the algorithm can "knock" on it and it might open or not according to some governing probability distribution, that may depend on other doors being open or closed[1]. We focus on settings in which doors are positively correlated, which informally means that the probability of opening a door is never decreased if another door is open. The governing distributions and their dependencies are known to the algorithm in advance. Crucially, however, during the execution, it gets no direct feedback on whether or not a door has opened unless all d doors have opened, in which case the task is completed.

This research has actually originated from our research on heuristic search on trees [4]. Consider a tree of depth d with a treasure placed at one of its leaves. At each step the algorithm can "check" a vertex, which is child of an already checked vertex. Moreover, for each level of the tree, the algorithm has a way to compare the previously checked vertices on that level. This comparison has the property that if the ancestor of the treasure on that level was already checked, then it will necessarily be considered as the "best" on that level. Note, however, that unless we checked all the vertices on a given level, we can never be sure that the vertex considered as the best among checked vertices in the level is indeed the correct one. With such a guarantee, and assuming that the algorithm gets no other feedback from checked vertices, any reasonable algorithm that is about to check a vertex on a given level, will always choose to check a child of the current best vertex on the level above it. Therefore, the algorithm can be described as a sequence of levels to inspect. Moreover, if we know the different distributions involved, then we are exactly at the situation of the dependent doors problem. See the full version for more details on this example.

Another manifestation of d dependent doors can arise in the context of cryptography. Think about a sequence of d cascading encryptions, and separate decryption protocols to attack each of the encryptions. Investing more efforts in decrypting the i'th encryption would increase the chances of breaking it, but only if previous encryptions where already broken. On the other hand, we get no feedback on an encryption being broken unless all of them are.

The case of two doors can serve as an abstraction for exploration vs. exploitation problems, where it is typically the case that deficient performances on the exploration part may result in much waste on the exploitation part [10, 17]. It can also be seen as the question of balance between searching and verifying in algorithms that can be partitioned thus [1, 15]. In both

[1] Actually, the distribution associated with some door i may depend on the state of other doors (being open or closed) not only at the current knock, but also at the time of each of the previous knocks on door i.

examples, there may be partial or even no feedback in the sense that we don't know that the first procedure succeeded unless the second one also succeeds.

For simplicity, we concentrate on scenarios in which the dependencies are *acyclic*. That is, if we draw the directed dependency graph between doors, then this graph does not contain any directed cycles. The examples of searching and verifying and the heuristic search on trees can both be viewed as acyclic. Moreover, despite the fact that many configurations are not purely acyclic, one can sometimes obtain a useful approximation that is.

To illustrate the problem, consider the following presumably simple case of two dependent memoryless doors. The first door opens on each knock independently with probability $1/2$. The second door can only open if the first door is open, in which case it opens on each knock independently, with probability $1/2$. What is the sequence of knocks that minimizes the expected time to open both doors, remembering that we don't know when door 1 opens? It is easy to see that the alternating sequence $1, 2, 1, 2, 1, 2, \ldots$ results in 6 knocks in expectation. Computer simulations indicate that the best sequence gives a little more than 5.8 and starts with $1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2$. Applied to this particular scenario, our theoretical lower bound gives 5.747, and our upper bound gives a sequence with expected time 5.832.

1.1 Context and Related Work

This paper falls under the framework of decision making under uncertainty, a large research subject that has received significant amount of attention from researchers in various disciplines, including computer science, operational research, biology, sociology, economy, and even psychology and cognition, see, *e.g.,* [2, 3, 5, 6, 7, 8, 9, 16].

Performing despite limited feedback would fit the framework of reinforced learning [17] and is inherent to the study of exploration vs. exploitation type of problems, including Multi-Armed Bandit problems [10]. In this paper we study the impact of having no feedback whatsoever. Understanding this extreme scenario may serve as an approximation for cases where feedback is highly restricted, or limited in its impact. For example, if it turns out that the price of lacking feedback is small, then it may well be worth to avoid investing efforts in complex methods for utilizing the partial feedback.

Of particular interest is the case of two doors. As mentioned, difficulties resulting from the lack of feedback can arise when one aims to find a solution by alternating between two subroutines: Producing promising candidate solutions and verifying these candidates. Numerous strategies are based on this interplay, including heuristics based on brute force or trail and error approaches [1, 15], sample and predict approaches [11, 14, 17], iterative local algorithms [12, 13], and many others. Finding strategies for efficiently balancing these two tasks can be therefore applicable.

1.2 Setting

There are $d \geq 2$ doors and each door can be either open or closed. Doors start closed, and once a door opens it never closes. To open a door, an algorithm can knock on it and it might open or not according to some probability distribution. The goal is to minimize the expected number of knocks until all doors open. Crucially, the algorithm has no feedback on whether or not a door has opened, unless all doors have opened, in which case the task is completed.

The probability that a door opens may depend on the state of other doors (being open or closed) at the time of the current knock as well as on their state during each of the previous knocks on the door. For example, the probability that a certain knock at door i succeeds may depend on the number of previous knocks on door i, but counting only those that were made while some other specific door j was open. The idea behind this definition is that the more time we invest in opening a door the more likely it is to open, and the quality of each knock depends on what is the state of the doors it depends on at the time of the knock.

Below we provide a semi-formal description of the setting. The level of detail is sufficient to understand the content of the main text, which is mainly concerned with independent and cascading configurations. The reader interested in a more formal description of the model is referred to the full version.

A specific setting of doors is called a *configuration* (normally denoted $\mathcal{C}$). This includes a description of all dependencies between doors and the resulting probability distributions. In this paper we assume that the dependency graph of the doors is acyclic, and so we may assume that a configuration describes an ordering of the doors, such that each door depends only on lower index doors. Furthermore, we assume that the correlation between doors is positive, *i.e.*, a door being open can only improve the chances of other doors to open.

Perhaps the simplest configuration is when all doors are *independent* of each other. In this case, door i can be associated with a function $p_i : \mathbb{N} \to [0,1]$, where $p_i(n)$ is the probability that door i is not open after knocking on it n times. Another family of acyclic configurations are *cascading* configurations. Here, door i cannot open unless all doors of lower index are already open. In this case, the configuration can again be described by a set of functions $\{p_i\}_{i=1}^{d}$, where $p_i(n)$ describes the probability that door i is not open after knocking on it n times, where the count starts only after door $i-1$ is already open.

In general, given a configuration, each door i defines a non-decreasing function $p_i : \mathbb{N} \to [0,1]$, called the *fundamental distribution* of the door, where $p_i(n)$ is the probability that the door is not open after knocking on it n times in the best of conditions, *i.e.*, assuming all doors of lower index are open. In the case of independent and cascading configurations, the fundamental distribution p_i coincides with the functions mentioned above. Two doors are *similar* if they have the same fundamental distribution. Two configurations are *similar* if for every i, door i of the first configuration is similar to door i of the second.

When designing an algorithm, we will assume that the configuration it is going to run in is known. As there is no feedback, a deterministic algorithm can be thought of as a possibly infinite sequence of door knocks. A randomized algorithm is therefore a distribution over sequences, and as all of them will have expected running time at least as large as that of an optimal sequence (if one exists), the expected running time of a randomized algorithm cannot be any better. Denote by $\mathbb{T}_{\mathcal{C}}(\pi)$, the expected time until all doors open when running sequence π in configuration $\mathcal{C}$. We define $\mathbb{T}_{\mathcal{C}} = \min_\pi \mathbb{T}_{\mathcal{C}}(\pi)$. As proved in the full version of the paper, there exists a sequence achieving this minimum. Therefore, by the aforementioned arguments, we can restrict our discussion to deterministic algorithms only.

If we had feedback we would knock on each door until it opens, and then continue to the next. Denoting by $E_i = \sum_{n=0}^{\infty} p_i(n)$ the expected time to open door i on its own, the expected running time then does not depend on the specific dependencies between doors at all, and is $\sum_i E_i$. Also, this value is clearly optimal. To evaluate the impact of lacking feedback for a configuration $\mathcal{C}$, we therefore define:

$$\texttt{Price}(\mathcal{C}) = \frac{\mathbb{T}_{\mathcal{C}}}{\sum_i E_i} .$$

Obviously $\texttt{Price}(\mathcal{C}) \geq 1$, and for example, if all doors start closed and open after just 1 knock, it is in fact equal to 1. In the full version of this paper we also show that $\texttt{Price}(\mathcal{C}) \leq d$.

1.3 Our Results

We have two main results. The first one, presented in Section 2, states that any two similar configurations have the same optimal running time up to a constant factor. We stress that this constant factor is universal in the sense that it does not depend on the specific distributions or on the number of doors d.

Furthermore, given a configuration, we identify an algorithm that is optimal for it up to a constant factor. We then show that for configurations where all doors are similar, there is a much simpler algorithm which is optimal up to a constant factor, and describe a formula that computes its approximate running time. We conclude Section 2 by analysing the price of lacking feedback for several configurations governed by standard fundamental distributions. In particular, we show that the price is logarithmic in d for memoryless doors, but can potentially grow to be linear in d for other distributions.

We then turn our attention to identify exact optimal sequences. Perhaps the simplest case is the case of two cascading memoryless doors. That is, the first door opens on each knock independently with probability p_1. The second door can only open if the first door is open, in which case it opens on each knock independently, with probability p_2. In Section 3 we present our second main result: Algorithms for these configurations that achieve the precise optimal running time up to an additive term of 1.

On the technical side, to establish such an extremely competitive algorithm, we first consider a semi-fractional variant of the problem and find a sequence that achieves the precise optimal bound. We then approximate this semi-fractional sequence to obtain an integer solution losing only an additive term of 1 in the running time. A nice anecdote is that in the case where $p_1 = p_2$ and are very small, the ratio of 2-knocks over 1-knocks in the sequence we get approaches the golden ratio. Also, in this case, the optimal running time approaches $3.58/p_1$ as p_1 goes to zero. It follows that in this case, the price of lacking feedback tends to $3.58/2$ and the price of dependencies, *i.e.*, the multiplicative gap between the cascading and independent settings, tends to $3.58/3$.

2 Near Optimal Algorithms

The following important lemma is proved in the full version using a coupling argument:

▶ **Lemma 1.** *Consider similar configurations $\mathcal{C}, \mathcal{X}$ and $\mathcal{I}$, where $\mathcal{X}$ is cascading and $\mathcal{I}$ is independent. For every sequence π, $\mathbb{T}_{\mathcal{I}}(\pi) \leq \mathbb{T}_{\mathcal{C}}(\pi) \leq \mathbb{T}_{\mathcal{X}}(\pi)$. This also implies that $\mathbb{T}_{\mathcal{I}} \leq \mathbb{T}_{\mathcal{C}} \leq \mathbb{T}_{\mathcal{X}}$.*

The next theorem presents a near optimal sequence of knocks for a given configuration. In fact, by Lemma 1, this sequence is near optimal for any similar configuration, and so we get that the optimal running time for any two similar configurations is the same up to a universal multiplicative factor.

▶ **Theorem 2.** *There is a polynomial algorithm*[2]*, that given a configuration $\mathcal{C}$ generates a sequence π such that $\mathbb{T}_{\mathcal{C}}(\pi) = \Theta(\mathbb{T}_{\mathcal{I}})$. In fact, $\mathbb{T}_{\mathcal{C}}(\pi) \leq 2 + 4\mathbb{T}_{\mathcal{I}} \leq 2 + 4\mathbb{T}_{\mathcal{C}}$.*

Proof. Denote by $p_1, \ldots, p_d$ the fundamental distributions of the doors of $\mathcal{C}$. For a finite sequence of knocks α, denote by $\mathtt{SC}_{\mathcal{C}}(\alpha)$ the probability that after running α in configuration

[2] A polynomial algorithm in our setting generates the next knock in the sequence in polynomial time in the index of the knock and in d, assuming that reading any specific value of any of the fundamental distributions of a door takes constant time.

$\mathcal{C}$, some of the doors are still closed. Note that if α is *sorted*, that is, if all knocks on door 1 are done first, followed by the knocks on doors 2, etc., then $\texttt{SC}_{\mathcal{X}}(\alpha) = \texttt{SC}_{\mathcal{I}}(\alpha)$.

We start by showing that for any T, we can construct in polynomial time a finite sequence α_T of length T that maximizes the probability that all doors will open, *i.e.*, minimizes $\texttt{SC}_{\mathcal{I}}(\alpha_T)$. As noted above, if we sort the sequence, this is equal to $\texttt{SC}_{\mathcal{X}}(\alpha_T)$.

The algorithm follows a dynamic programming approach, and calculates a matrix A, where $A[i,t]$ holds the maximal probability that a sequence of length t has of opening all of the doors $1, 2, \ldots, i$. All the entries $A[0,\cdot]$ are just 1, and the key point is that for each i and t, knowing all of the entries in $A[i,\cdot]$, it is easy to calculate $A[i+1,t]$:

$$A[i+1,t] = \max_{k=0}^{t} A[i,t-k] \cdot (1 - p_{i+1}(k)) \ .$$

Calculating the whole table takes $O(dT^2)$ time, and $A[d,T]$ will give us the highest probability a sequence of length T can have of opening all doors. Keeping tabs on the choices the max in the formula makes, we can get an optimal sequence α_T, and can take it to be sorted.

Consider the sequence $\pi = \alpha_2 \cdot \alpha_4 \cdots \alpha_{2^n} \cdots$. The complexity of generating this sequence up to place T is $O(dT^2)$, and so this algorithm is polynomial. Our goal will be to compare $\mathbb{T}_{\mathcal{X}}(\pi)$ with $\mathbb{T}_{\mathcal{I}}(\pi^\star)$, where $\pi^\star$ is the optimal sequence for $\mathcal{I}$.

The following observation stems from the fact that for any natural valued random variable X, $\texttt{E}[X] = \sum_{n=0}^{\infty} \texttt{Pr}[X > n]$ and $\texttt{Pr}[X > n]$ is a non-increasing function of n.

▶ **Observation 3.** *Let $\{a_n\}_{n=1}^{\infty}$ be a strictly increasing sequence of natural numbers, and X be some natural valued random variable. Then:*

$$\sum_{n=1}^{\infty}(a_{n+1} - a_n)\texttt{Pr}[X > a_{n+1}] \le \texttt{E}[X] \le a_1 + \sum_{n=1}^{\infty}(a_{n+1} - a_n)\texttt{Pr}[X > a_n] \ .$$

For a sequence π, denote by $\pi[n]$ the prefix of π of length n. In this terminology, $\mathbb{T}_{\mathcal{C}}(\pi) = \sum_{n=0}^{\infty} \texttt{SC}_{\mathcal{C}}(\pi[n])$. Setting $a_n = 2 + 4 + \ldots + 2^n$ in the right side of Observation 3, and letting X be the number of rounds until all doors open when using π, we get:

$$\begin{aligned}\mathbb{T}_{\mathcal{X}}(\pi) &\le 2 + \sum_{n=1}^{\infty} 2^{n+1} \cdot \texttt{SC}_{\mathcal{X}}(\pi[2 + \ldots + 2^n]) \le 2 + \sum_{n=1}^{\infty} 2^{n+1} \cdot \texttt{SC}_{\mathcal{X}}(\alpha_{2^n}) \\ &= 2 + \sum_{n=1}^{\infty} 2^{n+1} \cdot \texttt{SC}_{\mathcal{I}}(\alpha_{2^n}) \le 2 + \sum_{n=1}^{\infty} 2^{n+1} \cdot \texttt{SC}_{\mathcal{I}}(\pi^\star[2^n]) \le 2 + 4\mathbb{T}_{\mathcal{I}}(\pi^\star)\end{aligned}$$

The last step is using Observation 3 with $a_n = 2^{n-1}$. Theorem 2 concludes. ◀

2.1 Configurations where all Doors are Similar

In this section we focus on configurations where all doors have the same fundamental distribution $p(n)$. We provide simple algorithms that are optimal up to a universal constant, and establish the price of lacking feedback with respect to a few natural distributions. Corresponding proofs appear in the full version of the paper.

2.1.1 Simple Algorithms

Let us consider the following very simple algorithm $A_{\texttt{simp}}$. It runs in phases, where in each phase it knocks on each door once, in order. As a sequence, we can write $A_{\texttt{simp}} = (1, 2, \ldots, d)^{\infty}$. Let $X_1, \ldots, X_d$ be i.i.d. random variables taking positive integer values, satisfying $\texttt{Pr}[X_i > n] = p(n)$. The following is straightforward:

▶ **Claim 4.** $\mathbb{T}_{\mathcal{I}}(A_{\texttt{simp}}) = \Theta\left(d \cdot \texttt{E}\left[\max\left\{X_1, \dots, X_d\right\}\right]\right)$

This one is less trivial:

▶ **Claim 5.** *If all doors are similar then* $\mathbb{T}_{\mathcal{I}}(A_{\texttt{simp}}) = \Theta(\mathbb{T}_{\mathcal{I}})$

The claim above states that $A_{\texttt{simp}}$ is optimal up to a multiplicative constant factor in the independent case, where all doors are similar. As a result, we can also show:

▶ **Claim 6.** *Denote by* α_n *the sequence* $1^{2^n}, \dots, d^{2^n}$*. If all doors are similar then for any configuration* $\mathcal{C}$*,* $\mathbb{T}_{\mathcal{C}}\left(\alpha_0 \cdot \alpha_1 \cdot \alpha_2 \cdots\right) = \Theta(\mathbb{T}_{\mathcal{C}})$*.*

In plain words, the above claim states that the following algorithm is optimal up to a universal constant factor for any configuration where all doors are similar: Run in phases where phase n consists of knocking 2^n consecutive times on each door, in order.

2.1.2 On the Price of Lacking Feedback

By Claims 4 and 5, investigating the price of lacking feedback when all doors are similar boils down to understanding the expected maximum of i.i.d. random variables.

$$\texttt{Price} = \Theta\left(\frac{\texttt{E}\left[\max\left\{X_1, \dots, X_d\right\}\right]}{\texttt{E}\left[X_1\right]}\right) \tag{1}$$

Note that we omitted dependency on the configuration, as by Theorem 2, up to constant factors, it is the same price as in the case where the doors are independent. Let us see a few examples of this value. First:

▶ **Lemma 7.** *If* $X_1, \dots, X_d$ *are i.i.d. random variables taking natural number values, then:*

$$\texttt{E}\left[\max(X_1, \dots, X_d)\right] = \Theta\left(\kappa + d\sum_{n=\kappa}^{\infty} \texttt{Pr}\left[X_i > n\right]\right)$$

Where $\kappa = \min\left\{n \in \mathbb{N} \,|\, \texttt{Pr}\left[X_1 > n\right] < 1/d\right\}$

▶ **Example 8.** After the first knock on it, each door opens with probability $1 - 1/d$ and if it doesn't, it will open at its $d+1$'st knock. The expected time to open each door on its own is 2. By Lemma 7, as $\kappa = d+1$, we get that $\texttt{Price} = \Omega(\kappa) = \Omega(d)$. Since always $\texttt{Price} \leq d$, $\texttt{Price} = \Theta(d)$.

▶ **Example 9.** If $p(n) = q^n$ for some $1/2 < q < 1$, then $\texttt{Price} = \Theta(\log(d))$.

▶ **Example 10.** If for some $c > 0$ and $a > 1$, $p(n) = \min(1, c/n^a)$, then $\texttt{Price} = \Theta(d^{\frac{1}{a}})$.

Sometimes we know a bound on some moment of the distribution of opening a door. If $\texttt{E}\left[X_1\right] < M$, since $\texttt{Price} \leq d$, then $\mathbb{T} = O(d^2 M)$. Also,

▶ **Example 11.** If $\texttt{E}\left[X_1^a\right] < M$ for some $a > 1$, then $\mathbb{T} = O\left(d^{1+\frac{1}{a}} M^{1/a}(1 + \frac{1}{a-1})\right)$.

For example, if the second moment of the time to open a door on its own is bounded, we get an $O(d^{3/2})$ algorithm.

3 Two Memoryless Cascading Doors

One can say that by Theorem 2 we solved much of the dependent doors problem. There is an equivalence of the independent and cascading models, and we give an up to constant factor optimal algorithm for any situation. However, we still find the question of finding the true optimal sequences for cascading doors to be an interesting one. What is the precise cost of having no feedback, in numbers? Even the simple case of two doors, each opening with probability 1/2 on each knock, turns out to be quite challenging and has a not so intuitive optimal sequence.

In this section, we focus on a very simple yet interesting case of the cascading door problem, and solve it almost exactly. We have two doors. Door 1 opens with probability p_1 each time we knock on it, and door 2 opens with probability p_2. We further extend the setting to consider different durations. Specifically, we assume that a knock on door 1 takes one time unit, and a knock on door 2 takes c time units. Denote $q_1 = 1 - p_1$ and $q_2 = 1 - p_2$. For brevity, we will call a knock on door 1 a *1-knock*, and a knock on door 2 a *2-knock*.

The Semi-Fractional Model. As finding the optimal sequence directly proved to be difficult, we introduce a relaxation of our original model, termed the *semi-fractional model.* In this model, we allow 1-knocks to be of any length. A knock of length t, where t is a non-negative real number, will have probability of $1 - q_1^t$ of opening the door. In this case, a sequence consists of the alternating elements 1^t and 2, where 1^t describes a knock of length t on door 1. We call sequences in the semi-fractional model *semi-fractional sequences*, and to differentiate, we call sequences in the original model *integer sequences.*

As our configuration $\mathcal{C}$ will be clear from context, for a sequence π, we define $\mathtt{E}[\pi] = \mathbb{T}_{\mathcal{C}}(\pi)$ to be the expected running time of the sequence. Clearly, every integer sequence has a similar semi-fractional sequence with the same expected running time. As we will see, the reverse is not far from being true. That being so, finding the optimal semi-fractional sequence will give an almost optimal integer sequence.

3.1 Equivalence of Models

▶ **Theorem 12.** *Every semi-fractional sequence π has an integer sequence π', s.t.,* $\mathtt{E}[\pi'] \le \mathtt{E}[\pi] + 1$.

For this purpose, in this subsection only, we describe a semi-fractional sequence π as a sequence of non-decreasing non-negative real numbers: $\pi_0, \pi_1, \pi_2, \ldots$, where $\pi_0 = 0$. This sequence describes the following semi-fractional sequence (in our original terms):

$$1^{\pi_1 - \pi_0} \cdot 2 \cdot 1^{\pi_2 - \pi_1} \cdot 2 \cdots$$

This representation simplifies our proofs considerably. Here are some observations:

- 1-knocks can be of length 0, yet we still consider them in our indexing.
- The sequence is an integer sequence iff for all i, $\pi_i \in \mathbb{N}$.
- The i-th 2-knock starts at time $\pi_i + c(i-1)$ and ends at $\pi_i + ci$.
- The probability of door 1 being closed after the completion of the i-th 1-knock is $q_1^{\pi_i}$, and so the probability it opens at 1-knock i is $q_1^{\pi_{i-1}} - q_1^{\pi_i}$

▶ **Lemma 13.** *For two sequences $\pi = (\pi_0, \pi_1, \ldots)$ and $\pi' = (\pi'_1, \pi'_2, \ldots)$, if for all i, $\pi_i \le \pi'_i \le \pi_i + 1$ then* $\mathtt{E}[\pi'] \le \mathtt{E}[\pi] + 1$.

Lemma 13 is the heart of our theorem. Indeed, once proven, Theorem 12 follows in a straightforward manner. Given a semi-fractional sequence π, define $\pi'_i = \lceil \pi_i \rceil$. Then, π' is an integer sequence, and it satisfies the conditions of the lemma, so we are done. The lemma makes sense, as the sequence π' in which for all $i > 0$, $\pi'_i = \pi_i + 1$, can be thought of as adding a 1-knock of length one in the beginning of the sequence. Even if this added 1-knock did nothing, the running time would increase by at most 1. However, the proof is more involved, since in the lemma, while some of the 2-knocks may have an increased chance of succeeding, some may actually have a lesser chance.

Proof. Given a sequence π and an event X, we denote by $\mathtt{E}\left[\pi \,|\, X\right]$ the expected running time of π given the event X. Let X_i denote the event that door 1 opens at its i-th 1-knock. As already said:

$$\mathtt{Pr}\left[X_i\right] = q_1^{\pi_{i-1}} - q_1^{\pi_i} = \int_{\pi_{i-1}}^{\pi_i} q_1^x \ln(q_1)\, \mathrm{d}x$$

Where the last equality comes as no surprise, as it can be seen as modelling door 1 in a continuous fashion, having an exponential distribution fitting its geometrical one. Now:

$$\mathtt{E}\left[\pi\right] = \sum_{i=1}^{\infty} \mathtt{Pr}\left[X_i\right] \mathtt{E}\left[\pi \,|\, X_i\right] = \sum_{i=1}^{\infty} \int_{\pi_{i-1}}^{\pi_i} q_1^x \ln(q_1)\, \mathrm{d}x \cdot \mathtt{E}\left[\pi \,|\, X_i\right] = \int_0^{\infty} q_1^x \ln(q_1) \cdot \mathtt{E}\left[\pi \,\middle|\, X_{i(x)}\right] \mathrm{d}x$$

Where $i(x) = \max_i \{x \geq \pi_{i-1}\}$, that is, the index of the 1-knock that x belongs to when considering only time spent knocking on door 1. Defining X'_i and $i'(x)$ in an analogous way for π', we want to show that for all x,

$$\mathtt{E}\left[\pi' \,\middle|\, X'_{i'(x)}\right] \leq 1 + \mathtt{E}\left[\pi \,\middle|\, X_{i(x)}\right]$$

as using it with the last equality will prove the lemma. We need the following three claims:

1. If $j \leq i$, then $\mathtt{E}\left[\pi \,|\, X_j\right] \leq \mathtt{E}\left[\pi \,|\, X_i\right]$
2. For all x, $i'(x) \leq i(x)$
3. For all i, $\mathtt{E}\left[\pi' \,|\, X'_i\right] \leq 1 + \mathtt{E}\left[\pi \,|\, X_i\right]$

Together they give what we need:

$$\mathtt{E}\left[\pi' \,\middle|\, X'_{i'(x)}\right] \leq 1 + \mathtt{E}\left[\pi \,\middle|\, X_{i'(x)}\right] \leq 1 + \mathtt{E}\left[\pi \,\middle|\, X_{i(x)}\right]$$

The first is actually true trivially for all sequences, as the sooner the first door opens, the better the expected time to finish. For the second, since for all i, $\pi'_i \geq \pi_i$, then $x \geq \pi'_i$ implies that $x \geq \pi_i$, and so:

$$i'(x) = \max_i \left\{x \geq \pi'_{i-1}\right\} \leq \max_i \left\{x \geq \pi_{i-1}\right\} = i(x)$$

For the third, denote by Y_j the event that door 2 opens at the j'th 2-knock. Then:

$$\mathtt{E}\left[\pi \,|\, X_i\right] = \sum_{j=i}^{\infty} (\pi_j + cj) \mathtt{Pr}\left[Y_j \,|\, X_i\right]$$

Let us consider this same expression as it occurs in π'. First note that $\mathtt{Pr}\left[Y_j \,|\, X_i\right] = \mathtt{Pr}\left[Y'_j \,\middle|\, X'_i\right]$, as all that matters for its evaluation is $j - i$. Therefore:

$$\mathtt{E}\left[\pi' \,|\, X'_i\right] = \sum_{j=i}^{\infty} (\pi'_j + cj) \mathtt{Pr}\left[Y'_j \,\middle|\, X'_i\right] \leq \sum_{j=i}^{\infty} (\pi_j + 1 + cj) \mathtt{Pr}\left[Y_j \,|\, X_i\right]$$

$$= \mathtt{E}\left[\pi \,|\, X_i\right] + \sum_{j=i}^{\infty} \mathtt{Pr}\left[Y_j \,|\, X_i\right] \leq \mathtt{E}\left[\pi \,|\, X_i\right] + 1\,. \qquad ◀$$

3.2 The Optimal Semi-Fractional Sequence

A big advantage of the semi-fractional model is that we can find an optimal sequence for it. For that we need some preparation:

▶ **Definition 14.** For a semi-fractional sequence π, and some $0 \le x \le 1$, denote by $\mathtt{E}_x[\pi]$ the expected running time of π when started with door 1 being closed with probability x. In this notation, $\mathtt{E}[\pi] = \mathtt{E}_1[\pi]$.

▶ **Lemma 15.** *Let $y = x/(q_2 + p_2 x)$. Then:*

$$\mathtt{E}_x\left[1^t \cdot \pi\right] = t + \mathtt{E}_{q_1^t x}[\pi] \qquad\qquad \mathtt{E}_x[2 \cdot \pi] = c + \frac{x}{y}\mathtt{E}_y[\pi]$$

Proof. The first equation is clear, since starting with door 1 being closed with probability x, and then knocking on it for t rounds, the probability that this door is closed is $q_1^t x$.

As for the second equation, if door 1 is closed with probability x, then knocking on door 2, we have a probability of $p_2(1-x)$ of terminating, and so the probability we did not finish is:

$$1 - p_2(1-x) = 1 - p_2 + p_2 x = q_2 + p_2 x = \frac{x}{y}$$

It remains to show that conditioning on the fact that we indeed continue, the probability that door 1 is closed is y. It is the following expression, evaluated after a 2-knock:

$$\frac{\mathtt{Pr}[\text{door 1 is closed}]}{\mathtt{Pr}[\text{door 1 is closed}] + \mathtt{Pr}[\text{door 1 is open but not door 2}]} = \frac{x}{x + (1-x)q_2} = y\,. \qquad ◀$$

Applying Lemma 15 iteratively on a finite sequence w, we get:

$$\mathtt{E}_x[w\pi] = a(x,w) + b(x,w)\mathtt{E}_{\delta(x,w)}[\pi] \tag{2}$$

Of specific interest is $\delta(x,w)$. It can be thought of as the *state*[3] of our algorithm after running the sequence w, when we started at state x. Lemma 15 and Equation (2) give us the behaviour of $\delta(x,w)$:

$$\delta(x,1^t) = q_1^t x\,, \qquad \delta(x,2) = \frac{x}{q_2 + p_2 x}\,, \qquad \delta(x,aw) = \delta(\delta(x,a),w)\,.$$

We start with the state being 1, since we want to calculate $\mathtt{E}_1[\pi]$. Except for this first moment, as we can safely assume any reasonable algorithm will start with a 1-knock, the state will always be in the interval $(0,1)$. A 1-knock will always decrease the state and a 2-knock will increase it.

Our point in all this, is that we wish to exploit the fact that our doors are memoryless, and if we encounter a state we've already been at during the running of the sequence, then we should probably make the same choice now as we did then. The following definition and lemma capture this point.

▶ **Definition 16.** We say a non-empty finite sequence w is x-invariant, if $\delta(x,w) = x$.

The following Lemma is proved in the full version of this paper, and formalizes our intuition about how an optimal algorithm should behave.

▶ **Lemma 17.** *If w is x-invariant, and $\mathtt{E}_x[w\pi] \le \mathtt{E}_x[\pi]$ then $\mathtt{E}_x[w^\infty] \le \mathtt{E}_x[w\pi]$.*

[3] There is an intuitive meaning behind this. Going through Lemma 15, we can see that $\delta(1,w)$ is actually the probability that after running w, door 1 is closed conditioned on door 2 being closed. Indeed, After running some finite sequence, the only feedback we have is that the algorithm did not finish yet. We can therefore calculate from our previous moves what is the probability that door 1 is closed, and that is the only information we need for our next steps.

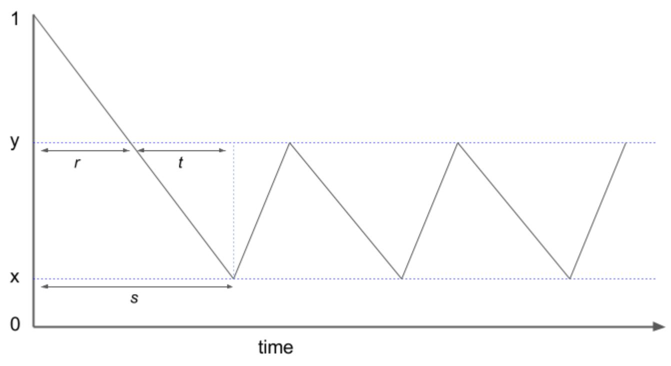

Figure 1 How the state evolves as a function of time. 1-knocks decrease the state, and 2-knocks increase it. Note that $r = \log_{q_1}(y)$ and $s = \log_{q_1}(x)$.

3.2.1 The Actual Semi-Fractional Sequence

▶ **Theorem 18.** *There is an optimal semi-fractional sequence $\pi^\star$ of the form $1^s(21^t)^\infty$, for some positive real values s and t, and its running time is:*

$$\mathtt{E}\left[\pi^\star\right] = \min_{z\in[0,1]} \left(\log_{q_1}(1-z) + \frac{c + (1-p_2 z)\log_{q_1}(1-p_2 z)}{p_2 z} \right) .$$

Proof. In the full version of this paper, we prove that there is an optimal semi-fractional sequence π. It clearly starts with a non-zero 1-knock, and so we can write $\pi = 1^s 2\pi'$. Intuitively, in terms of its state, this sequence starts at 1, goes down for some time with a 1-knock, and then jumps back up with a 2-knock. The state it reaches now was already passed through on the first 1-knock, and so as this is an optimal sequence we can assume it will choose the same as it did before, and keep zig-zaging up and down.

We next prove that indeed there is an optimal sequence following the zig-zaging form above. Again, take some optimal π, and write $\pi = 1^s 2\pi'$. Denote $x = \delta(1, 1^s)$ and $y = \delta(1, 1^s 2) = \delta(x, 2) > x$ (see Figure 1). Taking $r = \log_{q_1}(y) < s$, we get $\delta(1, 1^r) = y$. Denoting $t = s - r$, this means that $1^t 2$ is y-invariant. Since π is optimal, then:

$$\mathtt{E}\left[\pi\right] = \mathtt{E}\left[1^r(1^t 2)\pi'\right] \le \mathtt{E}\left[1^r \pi'\right] \quad \text{which implies:} \quad \mathtt{E}_y\left[1^t 2\pi'\right] \le \mathtt{E}_y\left[\pi'\right] .$$

So by Lemma 17:

$$\mathtt{E}_y\left[(1^t 2)^\infty\right] \le \mathtt{E}_y\left[1^t 2\pi'\right] \quad \text{which implies:} \quad \mathtt{E}\left[1^r(1^t 2)^\infty\right] \le \mathtt{E}\left[1^r 1^t 2\pi'\right] = \mathtt{E}\left[\pi\right] .$$

Therefore, $1^r(1^t 2)^\infty = 1^s(21^t)^\infty$ is optimal. We denote this sequence $\pi^\star$.

Now for the analysis of the running time of this optimal sequence. We will use Lemma 15 many times in what follows.

$$\mathtt{E}_1\left[1^s(21^t)^\infty\right] = s + \mathtt{E}_x\left[(21^t)^\infty\right] .$$

Denote $\alpha = (21^t)^\infty$.

$$\mathtt{E}_x\left[\alpha\right] = \mathtt{E}_x\left[21^t\alpha\right] = c + \frac{x}{y}\mathtt{E}_y\left[1^t\alpha\right] = c + \frac{x}{y}(t + \mathtt{E}_x\left[\alpha\right]) .$$

Since $t = s - r = \log_{q_1}(x/y)$:

$$\mathtt{E}_x\left[\alpha\right] = \frac{c}{1-\frac{x}{y}} + \frac{\frac{x}{y}}{1-\frac{x}{y}}\log_{q_1}(x/y) .$$

By Lemma 15, as our y is the state resulting from a 2-knock starting at state x, it follows that $y = x/(q_2 + p_2x)$. Since $x/y = q_2 + p_2x$, then $1 - x/y = p_2(1 - x)$ and then we get:

$$\frac{c}{p_2(1-x)} + \frac{q_2 + p_2x}{p_2(1-x)} \log_{q_1}(q_2 + p_2x) .$$

And in total:

$$\mathtt{E}_1\left[1^s(21^t)^\infty\right] = \log_{q_1}(x) + \frac{c + (q_2 + p_2x)\log_{q_1}(q_2 + p_2x)}{p_2(1-x)} .$$

Changing variable to $z = 1 - x$, results in $q_2 + p_2x = 1 - p_2z$, and we get the expression in the statement of the theorem. ◀

3.3 Actual Numbers

Theorem 18 gives the optimal semi-fractional sequence and a formula to calculate its expected running time. This formula can be approximated as accurately as we wish for any specific values of p_1, p_2 and c, but it is difficult to obtain a closed form formula from it. In the full version, we show an approximation with an additive error term $p2/\log(1/q_1)$. This is pretty close to p_2/p_1, and so when $p_1 \geq p_2$ it is just an additive error of 1.

In general, when p_1 is small, then the running time is shown to depend on $\theta \approx cp_1/p_2$, which is the expected time to open door 2 on its own, divided by the time to open door 1 on its own - a natural measure of the system. Then, ignoring the additive mistake, we show there that the lower bound is approximately $\mathcal{F}(\theta)/p_1$, where $\mathcal{F}$ is some function not depending on the parameters of the system. For example $\mathcal{F}(1) = 3.58$. So opening two similar doors without feedback when p is small takes about 3.58 times more time than opening one door as opposed to the case with feedback, where the factor is only 2.

We also note, that when the two doors are independent and similar, it is quite easy to see that the optimal expected running time is at most $3/p$. As a last interesting point, if $c = 1$ and $p = p_1 = p_2$ approaches zero, then the ratio between the number of 2-knocks and the number of 1-knocks approaches $\frac{1}{2}(1 + \sqrt{5})$, which is the golden ratio. These last two points are also shown in the full version of the paper.

3.4 Examples

For $p_1 = p_2 = 1/2$ and $c = 1$, the lower bound is 5.747. Simulations show that the best algorithm for this case is slightly more than 5.8, so the lower bound is quite tight, but our upper bound is 6.747 which is pretty far. However, the sequence we get from the upper bound proof starts with:

$$1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, \ldots$$

The value it gives is about 5.832, which is very close to optimal. For $p_1 = p_2 = 1/100$ and $c = 1$, the sequence we get is:

$$1^{97}, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 2, 2, 1, 2, 1, 2, \ldots$$

And the value it gives is about 356.756, while the lower bound can be calculated to be approximately 356.754. As we see this is much tighter than the +1 that our upper bound promises.

References

1 Xiaohui Bei, Ning Chen, and Shengyu Zhang. On the complexity of trial and error. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 31–40, 2013. `doi:10.1145/2488608.2488613`.

2 David E. Bell. Regret in decision making under uncertainty. *Operations Research*, 30(5):961–981, 1982. `doi:10.1287/opre.30.5.961`.

3 Michael Ben-Or and Avinatan Hassidim. The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS, 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 221–230, 2008. `doi:10.1109/FOCS.2008.58`.

4 Lucas Boczkowski, Amos Korman, and Yoav Rodeh. Searching on trees with noisy memory. *CoRR*, abs/1611.01403, 2016. URL: `http://arxiv.org/abs/1611.01403`.

5 Matthias Brand, Christian Laier, Mirko Pawlikowski, and Hans J. Markowitsch. Decision making with and without feedback: The role of intelligence, strategies, executive functions, and cognitive styles. *Journal of Clinical and Experimental Neuropsychology*, 31(8):984–998, 2009. PMID: 19358007. `doi:10.1080/13803390902776860`.

6 Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. Deterministic and probabilistic binary search in graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 519–532, 2016. `doi:10.1145/2897518.2897656`.

7 Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, October 1994. `doi:10.1137/S0097539791195877`.

8 L. A. Giraldeau and T. Caraco. *Social Foraging Theory*. Monographs in behavior and ecology. Princeton University Press, 2000.

9 Richard M. Karp and Robert Kleinberg. Noisy binary search and its applications. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'07, pages 881–890, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=1283383.1283478`.

10 Michael N. Katehakis and Arthur F. Veinott, Jr. The multi-armed bandit problem: Decomposition and computation. *Math. Oper. Res.*, 12(2):262–268, May 1987. `doi:10.1287/moor.12.2.262`.

11 Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, USA, 1994.

12 Amos Korman, Jean-Sébastien Sereni, and Laurent Viennot. Toward more localized local algorithms: removing assumptions concerning global knowledge. *Distributed Computing*, 26(5-6):289–308, 2013. `doi:10.1007/s00446-012-0174-8`.

13 Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1053, 1986. `doi:10.1137/0215074`.

14 Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

15 Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, 2006.

16 Andrzej Pelc. Searching games with errors – fifty years of coping with liars. *Theor. Comput. Sci.*, 270(1-2):71–109, 2002. `doi:10.1016/S0304-3975(01)00303-6`.

17 Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.