Parallel Bayesian Search with no Coordination

Pierre Fraigniaud, Amos Korman, Yoav Rodeh

Parallel Bayesian Search with no Coordination [*]

Pierre Fraigniaud[†] Amos Korman [‡] Yoav Rodeh [§]

Abstract

Coordinating the actions of agents (e.g., volunteers analyzing radio signals in SETI@home) yields efficient *search* algorithms. However, such an efficiency is often at the cost of implementing complex coordination mechanisms which may be expensive in term of communication and/or computation overheads. Instead, *non-coordinating* algorithms, in which each agent operates independently from the others, are typically very simple, and easy to implement. They are also inherently robust to slight misbehaviors, or even crashes of agents. In this paper, we investigate the "price of non-coordinating", in term of search performance, and we show that this price is actually quite small. Specifically, we consider a parallel version of a classical *Bayesian* search problem, where set of $k \geq 1$ searchers are looking for a treasure placed in one of the boxes indexed by positive integers, according to some distribution p. Each searcher can open a random box at each step, and the objective is to find the treasure in a minimum number of steps. We show that there is a very simple non-coordinating algorithm which has expected running time at most $4(1 - \frac{1}{k+1})^2\ \mathsf{OPT} + 10$, where OPT is the expected running time of the best fully coordinated algorithm. Our algorithm does not even use the precise description of the distribution p, but only the relative likelihood of the boxes. We prove that, under this restriction, our algorithm has the best possible competitive ratio with respect to OPT. For the case where a complete description of the distribution p is given to the search algorithm, we describe an optimal non-coordinating algorithm for Bayesian search. This latter algorithm can be twice as fast as our former algorithm in practical scenarios such as uniform distributions. All these results provide a complete characterization of non-coordinating Bayesian search. The take-away message is that, for their simplicity and robustness, non-coordinating algorithms are viable alternatives to complex coordinating mechanisms subject to significant overheads. Most of these results apply as well to *linear* search, in which the indices of the boxes reflect their relative importance, and where important boxes must be visited first.

[*]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).

[†]IRIF, CNRS and University Paris Diderot, Paris, France. E-mail: `Pierre.Fraigniaud@irif.fr`.

[‡]IRIF, CNRS and University Paris Diderot, Paris, France. E-mail: `Amos.Korman@irif.fr`.

[§]Weizmann Institute of Science. E-mail: `yoav.rodeh@gmail.com`.

1 Introduction

BOINC [22] (Berkeley Open Infrastructure for Network Computing) is a platform for volunteer computing supporting dozens of projects including the famous SETI@home analyzing radio signals for identifying signs of extra terrestrial intelligence. Most projects maintained at BOINC use parallel search mechanisms where a central server controls, and distributes the work to volunteers. The framework in this paper is an abstraction for projects operated at platforms similar to BOINC with hundreds of thousands distributed searchers. We address the following question: how to distribute the work among the searchers with respect to the amount of coordination between them provided by the central server? This paper drives to the conclusion that no coordination might actually be a quite viable strategy, both efficient and robust.

Specifically, we consider a parallel variant of the classical Bayesian search problem, typically attributed to Blackwell [6]. A *treasure* is placed according to some distribution p in one of infinitely many boxes, indexed by the positive integers. The search for the treasure is performed in parallel by $k \geq 1$ agents, also called *searchers*, which means that at each time step each searcher can "peek" into a box to check whether the treasure is present there. The goal is to minimize the expected time until the first searcher finds the treasure. We assume that the number k of searchers is known to the algorithm. We will consider two cases, whether p is given to the algorithm, or not. However, in the latter case, we assume that the algorithm is aware of the relative likelihood of the boxes. In both cases, we can assume, w.l.o.g., that the boxes B_x, $x \geq 1$, are ordered so that p is non-increasing, i.e., $p(x+1) \leq p(x)$ for every $x \geq 1$.

Let $s_1, \ldots, s_k$ be the k searchers at hand. If coordination is allowed, let $\mathtt{A}_{coord}$ be the algorithm that lets searcher s_i peek into box $B_{(t-1)k+i}$ at time t. A simple application of the rearrangement inequality shows that $\mathtt{A}_{coord}$ is an optimal algorithm, that is, it minimizes the expected time until one searcher finds the treasure. This time is $\sum_{x\geq 1} p(x)\lceil x/k \rceil$ since the box B_x is opened at time $\lceil x/k \rceil$ in $\mathtt{A}_{coord}$, and this box has probability $p(x)$ to contain the treasure. In particular, the optimal expected time to find the treasure with a single searcher is $\sum_{x\geq 1} x\, p(x)$. Therefore, if coordination is allowed, k searchers essentially allow to find the treasure k times faster than one searcher alone, in expectation. (Specifically, the speedup resulting from using k searchers approaches k when the expectation of the distribution p grows to infinity). However, as simple as this algorithm is, $\mathtt{A}_{coord}$ is very sensitive to faults of all sorts. For example, if one searcher crashes at some point during the execution then the searchers may completely miss the treasure, unless the protocol employs some mechanism for detecting such faults. Indeed, in $\mathtt{A}_{coord}$, each box is eventually opened by just one searcher. Namely, box $B_{(t-1)k+i}$ is opened only by searcher s_i, for every $t \geq 1$ and $1 \leq i \leq k$.

In this paper, we highlight the usefulness of a class of search algorithms, called *non-coordinating*, which is inherently robust. In such algorithms, all searchers operate independently, executing the same protocol, differing only in the outcome of the flips of their private random coins. A canonical example is the case of multiple random walkers that search a graph [1]. Although many search problems cannot be efficiently parallelized without coordination, when such parallelization can be achieved, the benefit can potentially be high, not only in terms of saving in communication and overhead in computation, but also in terms of robustness. To get some intuition, observe that when executing a non-coordinating algorithm, the correct operation as well as the running time can only improve if more searchers than planned are actually being used. Suppose for instance that an oblivious adversary is allowed to crash at most k' out of the k searchers at arbitrary times during the execution. To overcome the presence of k' faults, one can simply run the non-coordinating algorithm that is designed for the case of $k - k'$ searchers. If the running time of the non-coordinating algorithm for x searchers without crashes is $T(x)$, then the running time of the new robust (non-coordinating) algorithm would be at most $T(k - k')$. Note that even when coordination is allowed, one cannot expect to obtain robustness at a cost less than $\widehat{T}(k - k')$ in the worst case, where $\widehat{T}(x)$ denotes the cost of an optimal *coordinating* algorithm for x searchers without crashes, since the number of searchers that remain alive is in the worst case $k - k'$. Hence, if $T(\cdot)$ and $\widehat{T}(\cdot)$ are close, we get robustness almost for free by using a non-coordinating algorithm.

In this paper, we are interested in computing how much we lose in term of performance when using non-coordinating algorithms. Specifically, let $k \geq 1$, and let us denote by $\mathbb{T}_k(A, x)$ the expected time for an algorithm A to find the treasure with k searchers running in parallel if this treasure is placed at box x. Further, given a distribution p over the placement of the treasure in the boxes, let $\mathbb{T}_{p,k}(A)$ denote the expected time for A to find the treasure when it is placed in one of the boxes according to p. We have

$$\mathbb{T}_{p,k}(A) = \sum_{x \geq 1} p(x) \mathbb{T}_k(A, x). \tag{1}$$

With this notation, the expected running time of the optimal coordinating algorithm is $\mathbb{T}_{p,k}(\mathtt{A}_{coord}) = \sum_{x \geq 1} p(x) \lceil x/k \rceil$. We are interested in comparing these two terms, i.e., $\mathbb{T}_{p,k}(A)$ for a non-coordinating algorithm A versus $\mathbb{T}_{p,k}(\mathtt{A}_{coord})$, the complexity of the best search algorithm with full coordination. For this purpose, we use competitive analysis, and say that an algorithm A is c-competitive for k searchers looking for a treasure placed according to p if there is a constant b such that

$$\mathbb{T}_{p,k}(A) \leq c\, \mathbb{T}_{p,k}(\mathtt{A}_{coord}) + b.$$

We show that there is a non-coordinating algorithm with small competitive ratio, hence establishing that indeed one does not lose much in using non-coordinating algorithms.

Before going into the details of our results, let us observe that although the random placement of the treasure is the common setting in Bayesian search problems, yielding Eq. (1) for defining the complexity of a search algorithm, there is another abstract search setting which deserves to be investigated, that we call *linear* search. Indeed, searching for a proper divisor of a given number n, the typical approach to solve the problem consists of enumerating the candidate divisors in increasing order, from 2 to $\sqrt{n}$, and checking them one by one. This is because the probability that a random number is divisible by a given prime is inversely proportional to this prime. Similarly, in cryptography, an attack is better proceeded by systematically checking smaller keys than longer ones, because the time to check a key is typically exponential in its size. There are thus several contexts in which the search space can be *ordered* in a way such that, given that the previous trials were not successful, the next candidate according to the order is either the most preferable, or most likely to be valid, or the easiest to check. This led us to consider another measure of complexity, comparing the search time of an algorithm A to the search time of the algorithm with one searcher opening the boxes sequentially in order of their indices, namely

$$\mathbb{T}_k(A) = \max_{x \geq 1} \mathbb{T}_k(A, x)/x. \tag{2}$$

Again, we say that a search algorithm A is c-competitive for k searchers looking for a treasure arbitrarily placed in one box if there is a constant b such that

$$\mathbb{T}_k(A) \leq c\, \mathbb{T}_k(\mathtt{A}_{coord}) + b.$$

In the linear search setting, the aforementioned algorithm $\mathtt{A}_{coord}$ is also optimal. We show that, as for Bayesian search, one does not lose much in using non-coordinating algorithms in linear search.

1.1 Our Results

First, we design and analyze an *optimal* non-coordinating algorithm for Bayesian search, where p is given. Our algorithm, called $\mathtt{A}^{\star}$, has optimal expected running time among all non-coordinating algorithms. Specifically, for every distribution p, every $k \geq 1$, and every non-coordinating algorithm A,

$$\mathbb{T}_{p,k}(\mathtt{A}^{\star}) \leq \mathbb{T}_{p,k}(A).$$

A remarkable property satisfied by our non-coordinating algorithm $\mathtt{A}^\star$ is that, at any time $t > 1$ during its execution, all boxes that received a positive probability to be checked at some time $t' < t$, are now going to be checked at time t with equal probability. The design of $\mathtt{A}^\star$ is complex. However, when p is the uniform distribution over a finite domain, $\mathtt{A}^\star$ becomes simple to describe: at each step, each searcher running $\mathtt{A}^\star$ chooses a box uniformly among those it did not check at previous step. This natural algorithm for the uniform setting is optimal among all non-coordinating algorithms, and is shown to be at most 2 times slower than $\mathtt{A}_{coord}$.

Next, we focus on the notion of *order-invariant* algorithms, that is, algorithms assuming *only the knowledge of the relative likelihood* of the boxes (and not knowing the exact probability of finding the treasure in each box). Such algorithms are appealing because they are "universal", in the sense that, once the boxes have been reordered such that B_x is not less likely to contain the treasure than B_{x+1}, any order-invariant algorithm acts the same for all distributions. We present a very simple yet highly efficient non-coordinating order-invariant algorithm, called $\mathtt{A}_{order}$. In this algorithm, at step t, each searcher checks a box uniformly chosen among those it did not check yet in $\{1, \dots, \lceil \frac{t}{2} \rceil (k+1)\}$. The performance of $\mathtt{A}_{order}$ is essentially at most 4 times the expected running time of the best fully coordinated algorithm $\mathtt{A}_{coord}$. Precisely, for every distribution p, and every $k \geq 1$,

$$\mathbb{T}_{p,k}(\mathtt{A}_{order}) \leq 4\left(1 - \frac{1}{k+1}\right)^2 \mathbb{T}_{p,k}(\mathtt{A}_{coord}) + 10. \tag{3}$$

Ignoring the constant additive term, the aforementioned upper bound implies that the cost paid for not coordinating is just at most $16/9$ for two searchers, $9/4$ for three searchers, and approaches 4 as the number of searchers goes to infinity. In fact we show that these costs are tight in a very strong sense, as, for any given number of searchers, there is no non-coordinating order-invariant algorithm that achieves a better competitive ratio. Specifically, for every distribution p, every $k \geq 1$, and every order-invariant non-coordinating algorithm A, if there exist b and c such that $\mathbb{T}_{p,k}(A) \leq c\, \mathbb{T}_{p,k}(\mathtt{A}_{coord}) + b$, then

$$c \geq 4\left(1 - \frac{1}{k+1}\right)^2. \tag{4}$$

Algorithm $\mathtt{A}_{order}$ remembers all the boxes it checked, and so each searcher needs memory linear in the running time of the algorithm. We also consider $\mathtt{A}_{obliv}$ which at step t chooses one box uniformly at random in $\{1, \dots, k\lceil \frac{t}{2} \rceil\}$, hence potentially choosing many times the same box at different time steps. This algorithm uses memory that is just logarithmic in its running time, but performs almost as well as $\mathtt{A}_{order}$ for large number of searchers. Precisely, for every distribution p, and every $k \geq 1$,

$$\mathbb{T}_{p,k}(\mathtt{A}_{obliv}) \leq 4\, \mathbb{T}_{p,k}(\mathtt{A}_{coord}) + 2. \tag{5}$$

All the aforementioned upper bound results on order-invariant algorithms are actually established by considering the linear search setting, where the treasure is placed at an arbitrary box, and boxes are ordered by importance, that is, when focussing on the complexity $\mathbb{T}_k(A) = \max_{x \geq 1} \mathbb{T}_k(A, x)/x$ of any algorithm A (cf. Eq. (2)). Indeed, it turns out that the two settings (Bayesian search and linear search) are highly related, as far as order-invariant algorithms are concerned: an order-invariant algorithm that works well against a treasure placed arbitrarily would also work well in any probabilistic setting (under the assumption that the indices of the boxes are ordered according to their relative likelihood). In fact, in the linear search setting, $\mathtt{A}_{order}$ and $\mathtt{A}_{obliv}$ have the same competitive ratio as those mentioned in Eq. (3) and (5), respectively. Moreover, the lower bound of Eq. (4) also holds in the linear search setting, i.e., $\mathtt{A}_{order}$ has also optimal competitive ratio in this latter setting.

As mentioned earlier, since we are dealing with non-coordinating search algorithms, we immediately get robustness with respect to *crashes* of searchers. We point out that, in addition, our order-invariant algorithms are robust also for the case where each searcher has a possibly *different, distorted, list of boxes.* Such a distorted list may contain some extra boxes, and/or the order of the true boxes (those possibly containing the treasure) may be permuted. More precisely, for every box x, let $\sigma_{s_i}(x)$ be the box index of B_x in the "eyes" of searcher s_i, and let us denote by $\overline{\sigma}(x) = \frac{1}{k}\sum_{i=1}^{k}\sigma_{s_i}(x)$. We show that if $\overline{\sigma}(x) = x + o(x)$ for all x, then $\mathtt{A}_{order}$ and $\mathtt{A}_{obliv}$ still have essentially the same respective multiplicative ratios compared to $\mathtt{A}_{coord}$ running on the non-distorted list. In other words, the algorithm works practically just as well even if the average placement of x is not very far from its correct placement.

All the characteristics and performances of our algorithms are summarized in Table 1.

Algorithms	Performances	Comments
$\mathtt{A}_{coord}$	optimal expected and worst-case running time	Bayesian and linear search coordination is allowed
$\mathtt{A}^\star$	optimal expected running time 2 times slower than $\mathtt{A}_{coord}$ for $p \sim$ Unif	Bayesian search distribution p is given
$\mathtt{A}_{order}$	$4\left(1-\frac{1}{k+1}\right)^2$ times slower than $\mathtt{A}_{coord}$ optimal competitiveness w.r.t. $\mathtt{A}_{coord}$	Bayesian search relative likelihood is given
$\mathtt{A}_{obliv}$	4 times slower than $\mathtt{A}_{coord}$ uses logarithmic size memory	Bayesian search relative likelihood is given
$\mathtt{A}_{order}$ and $\mathtt{A}_{obliv}$	same competitiveness w.r.t. $\mathtt{A}_{coord}$ as for Bayesian $\mathtt{A}_{order}$ has optimal competitiveness w.r.t. $\mathtt{A}_{coord}$	Linear search
any $\mathtt{A}_{non-coord}$	robust to inconsistencies in the box indexing	Bayesian and linear search

Table 1: Characteristics and performances of the algorithms presented in the paper. In all cases, the number k of searchers is given as input to each searcher. All algorithms but $\mathtt{A}_{coord}$ run in absence of coordination.

In terms of techniques, our approach uses a crucial aspect of non-coordinating algorithms: they can be represented by infinite matrices. The performance analyses of our algorithms rely on carefully analyzing the corresponding matrices, and on using known properties of the Gamma function. Not surprisingly, demonstrating the optimality of $\mathtt{A}^\star$, and establishing the optimality of the competitive ratio of $\mathtt{A}_{order}$ are the hardest tasks. To establish these results, we first approximate matrices by continuous functions, and use compactness arguments. In fact, we establish a general technical lemma in the context of Measure theory, that we apply to different settings in our paper.

1.2 Related work

The case of a single searcher that searches for a randomly placed treasure received significant amount of attention from the communities of statistics, operational research and computer science (see e.g., [6, 11, 21]), and has been studied under various settings, including the cases where different costs are associated with the queries to the locations where the treasure is suspected to be (in our case, a query is just opening a box, and takes one step), where queries can be noisy (e.g., responses might be unreliable), and where the target may be mobile (see the book [24]).

When it comes to parallel search, most of the literature deals with mobile agents that search graphs of different topologies, and it is typically assumed that agents employ some form of communication. The literature on this subject is vast, and some good references can be found in, e.g., [2, 3, 10, 18, 23]. The major difference between our setting and the mobile agent setting, is that we allow "random access" to the different boxes. That is, our searcher can jump between different boxes at no cost. In other words, our focus is on the *query complexity* rather than the *move complexity.* Note, however, that the settings are related as the number of queries is always a lower bound on the number of moves.

Multiple random walkers are a special case of non-coordinating searchers. In a series of papers [1, 8, 13, 12] several results regarding hitting time, cover time, and mixing times are established, such as a linear speedup for several graph families including expanders and random graphs.

The linear search settings, in which the boxes are linearly ordered and the objective is to find a treasure placed in a box in time that is compared to its index, are related to the *cow-path problem* [4, 5, 19], which traditionally focused on the move complexity of a single mobile agent. In this context, it is worth pointing out [9] which considers the case of faulty agents.

Motivated by applications to central search foraging by desert ants, a parallel variant of the cow-path problem, called the ANTS problem, was introduced in [15, 16]. These latter works focus on non-coordinating searchers that start the search at a single location in a grid topology, and aim to find nearby treasures as fast as possible. In particular, it was shown therein that a speedup of $O(k)$ can be achieved with k non-coordinating searchers, and that a linear speedup cannot be achieved unless the agents have some knowledge of k. Following [15, 16] several other aspects of the ANTS problem were studied in, e.g., [20, 7, 14].

1.3 Outline

The paper is organized as follows. The next section provides the formal statement of the different search problems we are interested in. This section also includes results about the robustness of non-coordinating search algorithms, hence demonstrating their interest in fault-prone environments. In Section 3, we focus on the *order-invariant* algorithms (i.e., algorithms which have only access to the relative order of the boxes, either in term of likelihood, or importance), and we analyse the performances of $\mathtt{A}_{order}$ and $\mathtt{A}_{obliv}$. Then Section 4 focuses on the case of algorithms which are given a complete description of the distribution p as input. Algorithm $\mathtt{A}^\star$ is described and analyzed in that section. The lower bound for order-invariant algorithms is established in Section 5. Finally, Section 6 includes concluding remarks, and research perspectives. The paper is complemented by an Appendix containing the proofs of technical lemmas used throughout the paper.

2 Model and Formalization of the Problem

2.1 Bayesian and linear search

Let us consider an infinite set of *boxes* B_x indexed by the integers $x \geq 1$, and let us assume that a *treasure* is hidden in one of these boxes. A *searcher* is an algorithm that is given the ability to query boxes sequentially, in an order that is specified by the algorithm. The trivial search algorithm consists of querying box B_t at step $t \geq 1$. The result of a query to box B_x is a boolean b_x returning whether this box contains the treasure or not (hence, there is a unique x such that $b_x = \text{true}$).

Given $k \geq 1$, a k-searcher algorithm is a parallel algorithm performed concurrently by k searchers. The algorithm $\mathtt{A}_{coord}$ lets searcher s_i, $i \in \{1, \ldots, k\}$, query box $B_{k(t-1)+i}$ at step $t \geq 1$. This algorithm assumes the capacity to fully coordinate the searchers. Instead, this paper focuses on *non-coordinating* algorithms, that is, parallel algorithms in which each searcher performs individually the same algorithm, obliviously to the behaviors of the other searchers. During the execution of this type of algorithm, a searcher may reopen a box that has been opened before by another searcher. This is not noticeable by the former as querying a box lets no trace visible from the other searchers. In particular, two or more searchers may query the same box at the same time, but, to the perspective of each searcher, this is as if it is querying the box alone. Given a parallel algorithm A, we denote by $\mathbb{T}_k(A,x)$ the expected time until at least one of the k searchers executing A visits B_x for the first time, that is, $\mathbb{T}_k(A,x)$ is the expected number of steps until at least one searcher queries box B_x. (The expectation is taken over all the random choices made by the searchers during their execution). From this point on, for the sake of simplifying the notations, we shall not explicitly mention the number of searchers k in the notations. So, formally,

$$\mathbb{T}(A,x)=\sum_{t=0}^{\infty}\Pr[B_x \text{ was not queried by time } t]. \tag{6}$$

In this paper, we consider two scenarios regarding the way the treasure is placed in one of the boxes.

Bayesian search. The treasure is placed according to some probability distribution p over the positive integers, i.e., $p(x)\geq 0$ for every $x\geq 1$, and $\sum_{x\geq 1}p(x)=1$. We consider the cases in which the algorithm may or may not be given the distribution p as input. However, we always assume that the algorithm is given the relative likelihood of the boxes as input. Hence, we systematically assume that the boxes $B_x, x\geq 1$, are indexed such that $p(x+1)\leq p(x)$ for every $x\geq 1$. In this scenario, we focus on the expected time for finding the treasure, that is, we focus on the *average* complexity

$$\mathbb{T}_{\texttt{avg}}(A)=\sum_{x\geq 1}p(x)\,\mathbb{T}(A,x).$$

In the context of Bayesian search, we systematically assume that the distribution p satisfies $\sum_{x\geq 1}x\,p(x)<\infty$.

Linear search. The treasure is placed at an arbitrary position. In this scenario, the index of a box reflects its "importance" with respect to finding the treasure in it. The performances of the non-coordinating algorithms are thus compared to the position of the box where the treasure stands because this is the time a single searcher opening boxes in increasing order of their importance would take until it queries box B_x. That is, we focus on the *worst-case* complexity

$$\mathbb{T}_{\texttt{worst}}(A)=\max_{x\geq 1}\mathbb{T}(A,x)/x.$$

The terminology *linear* search comes from the fact that the boxes are linearly ordered, and must ideally be checked in that order.

2.2 The Best Coordinating Algorithm

The performances of our algorithm will be compared to the performances of the best algorithm with full coordination, i.e., the one minimizing $\mathbb{T}_{\texttt{avg}}$ or $\mathbb{T}_{\texttt{worst}}$ depending on the context. Recall that Algorithm $\texttt{A}_{coord}$ lets searcher s_i, $i\in\{1,\dots,k\}$, query box $B_{k(t-1)+i}$ at step $t\geq 1$. Note that $\mathbb{T}(\texttt{A}_{coord},x)=\lceil\frac{x}{k}\rceil$, and therefore, $\frac{1}{k}\,\mathbb{E}X\leq\mathbb{T}_{\texttt{avg}}(\texttt{A}_{coord})\leq\frac{1}{k}\,\mathbb{E}X+1$ where X is distributed according to p, and $\mathbb{T}(\texttt{A}_{coord},x)/x\to\frac{1}{k}$ when $x\to+\infty$. These facts express that, for both settings, $\texttt{A}_{coord}$ has essentially a speed up of k compared to a single searcher opening boxes in increasing order of their indices, up to additive constants.

Lemma 1. *$\texttt{A}_{coord}$ has optimal search time among coordinating algorithms, for both Bayesian search, and linear search.*

Proof. Let $k \geq 1$. Because we allow coordination, any randomized search algorithm is centralized, and thus can be seen as a distribution over deterministic search algorithms. It follows that it is enough to consider deterministic search algorithms. W.l.o.g, as this cannot harm the running time, one can restrict our analysis to algorithms that check each box only once, and for which there are exactly k boxes that are checked at each step. Let $\mathtt{OPT}$ be an optimal algorithm. The infinite sequence $\mathbb{T}(\mathtt{OPT}, 1), \mathbb{T}(\mathtt{OPT}, 2), \ldots$ contains exactly k copies of each positive integer. In the random placement scenario, let us consider a distribution p. Since $p(x)$ is a non-increasing sequence, it follows from the Rearrangement inequality that $\mathbb{T}_{\mathtt{avg}}(\mathtt{OPT}) = \sum_x p(x)\mathbb{T}(\mathtt{OPT}, x)$ is minimized when the $\mathbb{T}(\mathtt{OPT}, x)$'s are arranged in a non-decreasing order, which is exactly what algorithm $\mathtt{A}_{coord}$ does. Similarly, in the importance-driven scenario, $\mathbb{T}_{\mathtt{worst}}(\mathtt{OPT}) = \max_{x \geq 1} \mathbb{T}(\mathtt{OPT}, x)/x$ is also minimized when the $\mathbb{T}(\mathtt{OPT}, x)$'s are arranged in a non-decreasing order. □

2.3 Order-Invariance and Functional View

2.3.1 Order-Invariance

Among all types of search algorithms, one class of algorithms is particularly appealing in the context of non-coordinating algorithms: the class of *order-invariant* algorithms. We say that two distributions p and p' over the positive integers have *similar shapes* if, for every two integers x and y, we have: $p(x) \leq p(y) \iff p'(x) \leq p'(y)$.

Definition 2. *A search algorithm A is* order-invariant *if, for any two distributions p and p' over the positive integers with similar shapes, the queries performed by A for p and p' are identically distributed.*

By definition, order-invariant algorithms act the same for all distributions. We will show that some simple order-invariant algorithms are extremely efficient, and therefore practical.

2.3.2 Functional View

The notion of *functional view* of a search algorithm, defined below, is a central notion in our analysis.

Definition 3. *Given a non-coordinating search algorithm A, the* functional view *of A is the function $N : \mathbb{N}^+ \times \mathbb{N} \to [0, 1]$ defined as $N(x, t) = \Pr[B_x$ was not yet checked by time t by searcher $s_i]$ where s_i is an arbitrary searcher performing A.*

Hence, the probability that none of the k searchers checked x by time t is $N(x, t)^k$, and thus, by Eq. (6),

$$\mathbb{T}(A, x) = \sum_{t=0}^{\infty} N(x, t)^k.$$

As we shall throughout the paper see, the information encoded in this functional view of A is all that is needed to assess its running time (both in term of lower and upper bounds).

2.4 Competitive analysis

Lemma 1 yields the following definition for analyzing the performances of non-coordinating algorithms, independently from which of the two scenarios we consider, i.e., Bayesian search and linear search.

Definition 4. *Let $c \geq 1$. A Bayesian search algorithm A is* c-competitive *if there exists a constant $b \geq 0$ such that $\mathbb{T}_{\mathtt{avg}}(A) \leq c\, \mathbb{T}_{\mathtt{avg}}(\mathtt{A}_{coord}) + b$. Similarly, a linear search algorithm A is* c-competitive *if there* *exists a constant $b \geq 0$ such that $\mathbb{T}_{\texttt{worst}}(A) \leq c\, \mathbb{T}_{\texttt{worst}}(\mathtt{A}_{coord}) + b$. Lastly, an algorithm A is* pointwise *c-competitive if there exists a constant $b \geq 0$ such that, for every $x \geq 1$, $\mathbb{T}(A, x) \leq c\, \mathbb{T}(\mathtt{A}_{coord}, x) + b$.*

The different notions of competitiveness described in Definition 4 are related as follows:

Lemma 5. *For both Bayesian search and linear search, if a search algorithm is pointwise c-competitive, then it is c-competitive.*

Proof. Let $k \geq 1$, and let us consider any distribution p. For all x, $\mathbb{T}(A, x) \leq c\,\mathbb{T}(\mathtt{A}_{coord}, x) + b$. Therefore, in the Bayesian search, we have

$$\mathbb{T}_{\texttt{avg}}(A) = \sum_{x \geq 1} p(x)\mathbb{T}(A, x) \leq \sum_{x \geq 1} p(x)\left(c\,\mathbb{T}(\mathtt{A}_{coord}, x) + b\right) = c\,\mathbb{T}_{\texttt{avg}}(\mathtt{A}_{coord}) + b.$$

Similarly, in the linear search,

$$\mathbb{T}_{\texttt{worst}}(A) = \max_{x \geq 1} \mathbb{T}(A, x)/x \leq \max_{x \geq 1}\left(c\,\mathbb{T}(\mathtt{A}_{coord}, x)/x + b/x\right) \leq c\,\mathbb{T}_{\texttt{worst}}(\mathtt{A}_{coord}) + b,$$

which completes the proof. $\square$

The following section about robustness also serves as a warm up for the proof techniques in the paper.

2.5 Robustness

We conclude the section by some remarks on the robustness of non-coordinating algorithms. We have already observed that non-coordinating search algorithms are inherently robust with respect to crashes of searchers. We now point out that, for coordinating search, even a small difference in the boxes' order may be devastating to some algorithms (such as $\mathtt{A}_{coord}$). We show that, in the case of non-coordinating algorithms, this actually has little effect, as long as the biases are not too large. For every $x \geq 1$, let us denote by $\sigma_s(x)$ the index of box B_x in the "eyes" of searcher s_i, $i \in \{1, \ldots, k\}$. We call σ the resulting *disordering*, and we denote by $\mathbb{T}^{\sigma}(A, x)$ the expected time to check B_x when running an algorithm A under this disordering. We denote by

$$\overline{\sigma}(x) = \left\lceil \frac{1}{k} \sum_{i=1}^{k} \sigma_{s_i}(x) \right\rceil$$

the average placing of x according to σ. In essence, we show that if a non-coordinating algorithm is c-competitive, then it is also c-competitive under disordering of the boxes. More specifically:

Theorem 6. *Assume that a non-coordinating search algorithm A is pointwise c-competitive, and let σ be some disordering of the boxes. Then, there exists a constant b such that, for every $x \geq 1$, $\mathbb{T}^{\sigma}(A, x) \leq c\,\mathbb{T}\left(\mathtt{A}_{coord}, \overline{\sigma}(x)\right) + b$.*

Proof. Since A is c-competitive, there exists a constant a such that, for every $x \geq 1$, $\mathbb{T}(A, x) \leq c\,\mathbb{T}(\mathtt{A}_{coord}, x) + a$. Recall that N denotes the functional view of A, i.e., $N(x, t)$ denotes the probability that a fixed searcher s_i running A did not check box B_x by time t (when the boxes are ordered correctly). The probability that none of the searchers checked box x by time t in the disordered setting is $N(\sigma_1(x), t) \cdots N(\sigma_k(x), t)$. On the other hand, we have

$$\mathbb{T}^{\sigma}(A, x) = \sum_{t=0}^{\infty} \mathtt{Pr}\left[x \text{ was not checked by time } t\right].$$

Therefore,

$$\begin{aligned}\mathbb{T}^{\sigma}(A,x) &= \sum_{t=0}^{\infty} N(\sigma_1(x),t)\cdots N(\sigma_k(x),t)\\ &\leq \left(\sum_{t=0}^{\infty} N(\sigma_1(x),t)^k\right)^{\frac{1}{k}} \cdots \left(\sum_{t=0}^{\infty} N(\sigma_k(x),t)^k\right)^{\frac{1}{k}}\end{aligned}$$

where the inequality follows from a generalized form of Hölder's inequality [17]. Since

$$\mathbb{T}(A,x) = \sum_{t=0}^{\infty} N(x,t)^k,$$

we get from the above that

$$\mathbb{T}^{\sigma}(A,x) \leq \prod_{s=1}^{k} \mathbb{T}(A,\sigma_s(x))^{\frac{1}{k}}.$$

Using the AM-GM inequality for the second step below, it follows that

$$\begin{aligned}\mathbb{T}^{\sigma}(A,x) &\leq \prod_{s=1}^{k}\left(a + c\left\lceil\frac{\sigma_s(x)}{k}\right\rceil\right)^{\frac{1}{k}} \leq \frac{1}{k}\sum_{s=1}^{k}\left(a + c\left\lceil\frac{\sigma_s(x)}{k}\right\rceil\right) = a + \frac{c}{k}\sum_{s=1}^{k}\left\lceil\frac{\sigma_s(x)}{k}\right\rceil\\ &\leq a + c + \frac{c}{k}\sum_{s=1}^{k}\frac{\sigma_s(x)}{k} \leq a + c + \frac{c}{k}\overline{\sigma}(x) \leq a + c + c\left\lceil\frac{\overline{\sigma}(x)}{k}\right\rceil = a + c + c\mathbb{T}\left(A,\overline{\sigma}(x)\right).\end{aligned}$$

Hence, $\mathbb{T}^{\sigma}(A,x) \leq c\,\mathbb{T}\left(A,\overline{\sigma}(x)\right) + (a+c)$ as claimed (with $b = a+c$). □

Example 7. Assume that $\overline{\sigma}(x) = x + o(x)$, i.e., on average, the mistakes in ordering are not too large. In this case, Theorem 6 gives:

$$\begin{aligned}\mathbb{T}^{\sigma}(A,x) &\leq c\,\mathbb{T}\left(\mathtt{A}_{coord},\overline{\sigma}(x)\right) + b = c\left\lceil\frac{\overline{\sigma}(x)}{k}\right\rceil + b\\ &= (c+o(1))\left\lceil\frac{x}{k}\right\rceil + b = (c+o(1))\mathbb{T}\left(\mathtt{A}_{coord},x\right) + b\end{aligned}$$

where the $o(1)$ term goes to 0 when x goes to infinity. So, by Lemma 5, even under reasonable disordered conditions, where the average conceived ordering of a box is not too far from its correct place, we can achieve a competitiveness that is close to the competitiveness obtained assuming the boxes are perfectly ordered.

One cannot expect this same slight deterioration in competitiveness if boxes are disordered to be at a linear distance from where they truly are, and not at a sub-linear one. This is clear since our definition of disordering allows searchers to have "fake" boxes, i.e., boxes that have no corresponding true box. For example, let A have a true pointwise competitiveness of c, i.e., there is an infinite number of x's such that $\mathbb{T}(A,x) \geq c\mathbb{T}(\mathtt{A}_{coord},x)$. Take σ such that for all i and all x, $\sigma_i(x) = ax$ for some constant $a > 1$. Then, for all these x's:

$$\mathbb{T}^{\sigma}(A,x) = \mathbb{T}(A,ax) \geq c\mathbb{T}(\mathtt{A}_{coord},ax) \approx ac\mathbb{T}(\mathtt{A}_{coord},x).$$

As this is true for an infinite number of x's, the linear competitiveness of A is ac.

3 Non-Coordinating Search in Ordered Boxes

In this section we consider the setting in which only the ordering of boxes is known to the algorithm designer. This ordering may come from an a priori ordering of the boxes according to their "importance" (like in linear search), or be induced by the relative likelihood of the boxes coming from some unknown distribution p (like in Bayesian search). We design two algorithms, $\mathtt{A}_{order}$ and $\mathtt{A}_{obliv}$, and analyse their performances. Both will be shown to be highly competitive. While $\mathtt{A}_{order}$ achieve better competitive ratio (in fact, this competitive ratio will be shown to be optimal in Section 5), $\mathtt{A}_{obliv}$ is essentially memoryless, and its simplicity and efficiency make it an outstanding candidate for real life purposes.

3.1 Order-Invariant Algorithms

In this subsection, we study the competitiveness of the order-invariant non-coordinating algorithm $\mathtt{A}_{order}$, defined hereafter.

```
Algorithm $\mathtt{A}_{order}$ (as performed by each searcher):
    $\mathcal{I} \leftarrow \emptyset$
    for $t = 1$ to $\infty$ do
        pick a index $x$ uniformly at random in $\{1, \ldots, \lceil \frac{t}{2} \rceil (k+1)\} \setminus \mathcal{I}$.
        $\mathcal{I} \leftarrow \mathcal{I} \cup \{x\}$
        open box $B_x$
```

Before analysing the performances of the algorithm, we first establish the following technical lemma. Asymptotically, it is a known property of the Gamma function, but we will need this inequality for small a and b as well. The technical proof is deferred to Appendix A.

Lemma 8. *For any integers $b \geq a \geq 1$, and every real ϕ with $0 < \phi \leq 1$, $\prod_{i=a}^{b} \frac{i}{i+\phi} \leq \left(\frac{a}{b}\right)^{\phi}$.*

Theorem 9. *For every distribution p, and every $k \geq 1$,*

$$\mathbb{T}_{\mathtt{avg}}(\mathtt{A}_{order}) \leq 4\left(1 - \frac{1}{k+1}\right)^2 \mathbb{T}_{\mathtt{avg}}(\mathtt{A}_{coord}) + 10, \text{ and } \mathbb{T}_{\mathtt{worst}}(\mathtt{A}_{order}) \leq 4\left(1 - \frac{1}{k+1}\right)^2 \mathbb{T}_{\mathtt{worst}}(\mathtt{A}_{coord}) + 10.$$

Proof. Using Lemma 5, it is sufficient to show that, for every x,

$$\mathbb{T}(\mathtt{A}_{order}, x) \leq 4\left(1 - \frac{1}{k+1}\right)^2 \mathbb{T}(\mathtt{A}_{coord}, x) + 10.$$

For avoiding distinguishing odd and even time steps, count the time by grouping two consecutive steps, so at each step t, $\mathtt{A}_{order}$ chooses two distinct new boxes in $\{1, \ldots, t(k+1)\}$. We shall double the time later in the analysis, for balancing this acceleration of $\mathtt{A}_{order}$. (Note that the speeded up $\mathtt{A}_{order}$ might actually end mid-step, but this just yields an over estimation of its performances). The number of elements the algorithm chooses from at step t is $(k+1)t - 2(t-1) = (k-1)t + 2$. Box x starts to have some probability of being checked at time $s = \lceil x/(k+1) \rceil$, and for $t \geq s$ the probability of x not being checked by time t is:

$$\prod_{i=s}^{t}\left(1 - \frac{2}{(k-1)i+2}\right)^k = \prod_{i=s}^{t}\left(\frac{(k-1)i}{(k-1)i+2}\right)^k = \left(\prod_{i=s}^{t} \frac{i}{i + \frac{2}{k-1}}\right)^k.$$

We claim that the product inside the parenthesis is at most $\left(\frac{s+1}{t}\right)^{\frac{2}{k-1}}$. Two cases:

1. $k \geq 3$. In this case, $2/(k-1) \leq 1$, and so by Lemma 8 the product is at most:

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

2. $k = 2$. In this case, $2/(k-1) = 2$. If $s = t$ then the claim is clearly true, as the product is at most 1, and $((s+1)/t)^2 \geq 1$. Otherwise, the product is telescopic,

$$\prod_{i=s}^{t} \frac{i}{i+2} = \frac{s(s+1)}{(t+1)(t+2)} \leq \frac{(s+1)^2}{t^2}.$$

Denoting $a = 2k/(k-1)$, the expected running time $\mathbb{T}(\mathtt{A}_{order}, x)$ is then at most (times 2):

$$\sum_{t=0}^{\infty} \mathtt{Pr}\left[x \text{ not checked by time } t\right] \leq s + 2 + \sum_{t=s+2}^{\infty} \left(\frac{s+1}{t}\right)^a.$$

As $((s+1)/t)^a$ is decreasing with t, we can bound the sum from above by taking the integral but starting it at $s+1$ and not $s+2$. This gives the upper bound of:

$$\begin{aligned} s + 2 + \int_{s+1}^{\infty} \left(\frac{s+1}{t}\right)^a \mathrm{d}t &= s + 2 + (s+1) \int_{1}^{\infty} t^{-a}\, \mathrm{d}t = s + 2 + \frac{s+1}{a-1} \\ &= 1 + (s+1)\left(1 + \frac{1}{\frac{2k}{k-1} - 1}\right) = 1 + \left(\left\lceil \frac{x}{k+1} \right\rceil + 1\right)\left(1 + \frac{k-1}{k+1}\right) \\ &\leq 1 + \left(\frac{x}{k+1} + 2\right)\left(\frac{2k}{k+1}\right) \leq 5 + \frac{2k}{(k+1)^2} x \leq 5 + \frac{2k^2}{(k+1)^2}\left\lceil \frac{x}{k} \right\rceil. \end{aligned}$$

Multiplying by bound by 2 to rebalance the acceleration of $\mathtt{A}_{order}$ yields the result. □

3.2 Memory Efficient Version

In Algorithm $\mathtt{A}_{obliv}$, every searcher performs obliviously to what it has done at previous steps (it just has to count steps):

```
Algorithm $\mathtt{A}_{obliv}$ (as performed by each searcher):
    for $t = 1$ to $\infty$ do
        pick one index $x$ uniformly at random in $\{1, \ldots, \lceil \frac{t}{2} \rceil k\}$
        open box $B_x$
```

Theorem 10. *For every distribution p, every $k \geq 1$, and every non-coordinating algorithm A,*

$$\mathbb{T}_{\mathtt{avg}}(\mathtt{A}_{obliv}) \leq 4\,\mathbb{T}_{\mathtt{avg}}(\mathtt{A}_{coord}) + 2, \text{ and } \mathbb{T}_{\mathtt{worst}}(\mathtt{A}_{obliv}) \leq 4\,\mathbb{T}_{\mathtt{worst}}(\mathtt{A}_{coord}) + 2.$$

Proof. The proof proceeds in a very similar manner to that of Theorem 9, yet is in fact a little simpler. Again, by Lemma 5, it is sufficient to show that, for every x,

$$\mathbb{T}(\mathtt{A}_{obliv}, x) \leq 4\,\mathbb{T}(\mathtt{A}_{coord}, x) + 2.$$

As in the proof of Theorem 9, we count time by grouping two consecutive steps, resulting in $\mathtt{A}_{obliv}$ independently picking two indices i and j uniformly at random in $\{1, \ldots, kt\}$ at every step t. Box x starts to have some probability of being checked at time $s = \lceil x/k \rceil$, and for $t \geq s$ the probability of B_x not being checked by time t is:

$$\prod_{i=s}^{t} \left(\left(1 - \frac{1}{ki} \right)^2 \right)^k \leq \prod_{i=s}^{t} \left(1 - \frac{1}{ki+1} \right)^{2k} = \prod_{i=s}^{t} \left(\frac{i}{i + \frac{1}{k}} \right)^{2k} \leq \left(\frac{s}{t} \right)^2,$$

where the last inequality is by Lemma 8. The expected running time for x is then at most (times 2):

$$s + 1 + \sum_{t=s+1}^{\infty} \left(\frac{s}{t} \right)^2.$$

As $(s/t)^2$ is decreasing with t, we can bound the sum from above by taking the integral, but starting it at s and not $s+1$. This gives the upper bound of:

$$s + 1 + \int_{s}^{\infty} \left(\frac{s}{t} \right)^2 \mathrm{d}t = s + 1 + s \int_{1}^{\infty} \frac{1}{t^2} \, \mathrm{d}t = 2s + 1.$$

Multiplying by 2 gives the result. □

4 Bayesian search with complete knowledge

Ignoring the small additive term in Theorem 9, as k grows larger, $\mathtt{A}_{order}$ is about 4 times worse than the best coordinating algorithm. In this section we show that it is possible, for some distributions, to improve on this if the exact distribution is given as input to the algorithm, by designing an *optimal* algorithm for this setting. We denote this algorithm $\mathtt{A}^{\star}$.

4.1 The Case of Uniform Distributions

The first example that comes to mind is when the treasure is uniformly placed in one of the boxes $\{1, \ldots, M\}$. The most natural algorithm in this case is that each searcher, at each step, chooses uniformly between all boxes it did not check already. As we will show, this is exactly what algorithm $\mathtt{A}^{\star}$ does, and since we will show it is optimal, then the optimality of this natural algorithm follows. The analysis is simple, and we approximate it here for the case where $M \gg k$.

$$\begin{aligned}
\mathbb{T}_{\mathtt{avg}}(\mathtt{A}^{\star}) &= \sum_{t=0}^{M} \mathtt{Pr}\left[\text{treasure not found by time } t\right] \\
&= \sum_{t=0}^{M} \prod_{i=0}^{t-1} \left(1 - \frac{1}{M-i} \right)^k = \sum_{t=0}^{M} \prod_{i=0}^{t-1} \left(\frac{M-i-1}{M-i} \right)^k \\
&= \sum_{t=0}^{M} \left(\frac{M-t}{M} \right)^k = \frac{1}{M^k} \sum_{i=0}^{M} i^k \approx \frac{1}{M^k} \frac{M^{k+1}}{k+1} = \frac{M}{k+1}.
\end{aligned}$$

Note that with coordination, the expected running time would be approximately $M/2k$, so we lose about a factor of 2 by non-coordination as opposed to 4 in the case of Algorithm $\mathtt{A}_{order}$. This algorithm is memory intensive, yet if we choose to simplify and just choose uniformly at random a box from all boxes at each step, we get that the running time is practically the same for large M:

$$\sum_{t=0}^{\infty}\left(1-\frac{1}{M}\right)^{kt}=\frac{1}{1-\left(1-\frac{1}{M}\right)^{k}}\approx\frac{M}{k}.$$

4.2 Preliminaries

Let us fix some distribution p over the positive integers, and let us consider a non-coordinating algorithm A that is running on k searchers. Recall (cf. Definition 3) that the fucntional view N of A is defined by $N(x,t)=\Pr[B_x$ was not already checked by time t by $s]$ where s is an arbitrary searcher performing A.

Observation 11. *The functional view N of A satisfies $N(x,0)=1$ for all $x\geq 1$, and, for all $x\geq 1$ and $t\geq 1$, $N(x,t)=N(x,t-1)\cdot\Pr[x$ is not checked at time $t \mid x$ was not checked by time $t-1]$.*

Let us now consider such functions N on their own, possibly without a corresponding algorithm. So, let $N:\mathbb{N}^+\times\mathbb{N}\to[0,1]$, and, for time $t\geq 0$, let us consider

$$C_N(t)=\sum_{x\geq 1}(1-N(x,t)).$$

Note that all our sums over x are only over boxes with $p(x)>0$. In the case of an algorithm A, $C_N(t)$ is the expected number of elements that were checked by time t by just one of the searchers running A, and is therefore at most t. We say that N satisfies the *column requirement* at time t if

$$C_N(t)\leq t.$$

We define the set $\mathcal{V}$ of *valid* functions as:

$$\mathcal{V}=\left\{N:\mathbb{N}^+\times\mathbb{N}\to[0,1]\,\middle|\,\forall t, C_N(t)\leq t\right\}.$$

So functions N corresponding to algorithms are always valid. Finally, the *running time* of a valid function N is

$$\mathbb{T}(N)=\sum_{x\geq 1}p(x)\sum_{t\geq 0}N(x,t)^k=\sum_{t\geq 0}\sum_{x\geq 1}p(x)N(x,t)^k.$$

The running time of N is thus defined so that $\mathbb{T}(N)$ is indeed the expected running time of algorithm A when N is the functional view of A. Indeed,

$$\mathbb{T}(A,x)=\sum_{t\geq 0}\mathtt{Pr}\left[x\text{ was not found by time }t\right]=\sum_{t\geq 0}N(x,t)^k.$$

Example 12. Consider $\mathtt{A}_{order}$ when set to run on two searchers. At $t=1$ it picks uniformly from one of the boxes $\{1,2,3\}$, and at $t=2$ chooses another one out of the two not chosen. It then picks uniformly from one of the four remaining boxes of $\{1,\dots 6\}$, and then again. The following is a functional representation corresponding to to this part of the algorithm. Note that the column requirement is satisfied with equality, owing to the fact that $\mathtt{A}_{order}$ never rechecks a box.

$$\mathtt{A}_{order}=\begin{array}{c|ccccc}
{}^{t\rightarrow}_{x\downarrow} & 0 & 1 & 2 & 3 & 4\\
\hline
1 & 1 & 2/3 & 1/3 & 1/4 & 1/6\\
2 & 1 & 2/3 & 1/3 & 1/4 & 1/6\\
3 & 1 & 2/3 & 1/3 & 1/4 & 1/6\\
4 & 1 & 1 & 1 & 3/4 & 1/2\\
5 & 1 & 1 & 1 & 3/4 & 1/2\\
6 & 1 & 1 & 1 & 3/4 & 1/2
\end{array}$$

To lower bound the running time of algorithms, we find the optimal $N \in \mathcal{V}$, in the sense that it minimizes $\mathbb{T}(N)$. For that, we introduce an important lemma, that we call the *Presentation Lemma* for it provides a presentation of the optimal function we are looking for. At this point only a simple version of the Presentation Lemma is needed, yet we present it in its full generality, as it will be needed later in the paper. The notations adopted for the Presentation Lemma are taken from Measure theory, for two reasons at least. First, we need to apply the lemma to different settings, corresponding to different measure spaces. Second, and more importantly, our optimization methodology requires compactness arguments, which are perfectly expressed in measure theory.

Fix some $k \geq 2$, and let $(X, \mathcal{X}, \mu)$ be a measure space. For $T \geq 0$, let

$$V(T) = \{f : X \to [0,1] \text{ measurable, such that } \int (1-f)\,\mathrm{d}\mu \leq T\}.$$

For any measurable function $c : X \to [0, \infty)$, and for any real $\alpha \geq 0$, let us define the function

$$f_{c,\alpha} : X \to [0,1]$$

as:

$$f_{c,\alpha}(x) = \begin{cases} 1 & \text{if } c(x) = 0, \\ \min\left\{1, \alpha\, c(x)^{-\frac{1}{k-1}}\right\} & \text{otherwise.} \end{cases}$$

The proof of the Presentation Lemma can be found in Appendix B.

Lemma 13. (Presentation Lemma) *Let $c : X \to [0,\infty)$ be a measurable function, and let $T \geq 0$. If there is some $h \in V(T)$ such that $\int c\, h^k \,\mathrm{d}\mu < \infty$, then $\alpha = \min\{\beta \geq 0 \mid f_{c,\beta} \in V(T)\}$ exists, and, for every $g \in V(T)$, $\int c\, f_{c,\alpha}^k \,\mathrm{d}\mu \leq \int c\, g^k \,\mathrm{d}\mu$. Furthermore, if $\alpha \neq 0$ then $\int (1 - f_{c,\alpha})\,\mathrm{d}\mu = T$.*

This lemma is quite technical, but its purpose is given the weights $c(\cdot)$ of the elements of X, to find the $f : X \to [0,1]$, that minimizes $\sum_x c(x) f(x)^k$. All this under a simple linear constraint on f. This is exactly what we need.

As intuition for the proof of the lemma, assume X is a finite set, and consider some x and y where f is not zero. Let us change f slightly, by setting $f'(x) = f(x) + \epsilon$ and $f'(y) = f(y) - \epsilon$, for some small ϵ. This maintains the linear constraint, and changes the sum we wish to minimize by:

$$c(x)\left((f(x)+\epsilon)^k - f(x)^k\right) + c(y)\left((f(y)-\epsilon)^k - f(y)^k\right).$$

Taking the derivative w.r.t. ϵ, this is ϵk times:

$$c(x)(f(x)+\epsilon)^{k-1} - c(y)(f(y)-\epsilon)^{k-1}.$$

If f indeed minimizes the final sum, then at $\epsilon = 0$ this is supposed to be 0. Therefore, $c(x)f(x)^{k-1} = c(y)f(y)^{k-1}$. Put differently, fixing one y, and taking any x, $f(x) = \alpha c(x)^{-1/(k-1)}$, where $\alpha = f(y)c(y)^{1/(k-1)}$. Therefore, we see the form of the function as stated in the Lemma.

Towards finding the optimal $N \in \mathcal{V}$, fix some $t \geq 0$. Our aim is to minimize $\sum_{x \geq 1} p(x) N(x,t)^k$. This can be done for each t completely separately, where the Presentation Lemma comes into play. Let $\{1, \ldots, M\}$ be the support of the distribution p, where

$$M = \max\{x \geq 1 \mid p(x) > 0\}$$

if this maximum exists, and $M = +\infty$ otherwise. Recall that a function L satisfies the column requirement at time t if $C_L(t) \leq t$.

Lemma 14. *The following function L is in $\mathcal{V}$, and achieves minimal $\mathbb{T}(\cdot)$ over all valid functions.*

$$L(x,t) = \begin{cases} 0 & \text{if } t \geq M, \\ 1 & \text{if } t < M \text{ and } p(x) = 0, \\ \min\{1, \alpha(t)q(x)\} & \text{otherwise,} \end{cases}$$

where $q(x) = p(x)^{-\frac{1}{k-1}}$, and, for all $t < M$, $\alpha(t)$ is such that $C_L(t) = t$.

Proof. For $t \geq M$, L satisfies the column requirement and is clearly optimal. Fix $t < M$. Setting $X = \mathbb{N}^+$ with the trivial measure $\mu(x) = 1$ for all x, $T = t$, and $c = p$, Lemma 13 gives the form of the optimal function for this specific t, which is as it appears in L. In this case $\alpha(t) \neq 0$ because otherwise the column requirement is not satisfied, and so, by the last part of Lemma 13, $C_L(t) = t$. To check the condition of Lemma 13, take the constant function $h(x) = 1$. Clearly $h \in V(t)$, and $\int ch^k \, d\mu = \sum_x p(x) = 1 < \infty$. $\square$

Example 15. Let us consider a simple example: $k = 2$, $p(1) = 1/2$, $p(2) = 1/3$, and $p(3) = 1/6$, with $M = 3$. In this case, $q(1) = 2$, $q(2) = 3$ and $q(3) = 6$, and some quick calculations show that $\alpha(1) = 1/5$, $\alpha(2) = 1/11$, and $\alpha(3) = 0$. This gives:

$L =$

$t\to$ / $x\downarrow$	0	1	2	3
1	1	0.4	$2/11$	0
2	1	0.6	$3/11$	0
3	1	1	$6/11$	0

4.3 The Optimal Algorithm $\mathtt{A}^\star$

Although it may seem that every valid function N has a corresponding algorithm, it is not at all clear, because the conditional probabilities arising from Observation 11 quickly become complicated for general N when t increases. However, it turns out that because of the specific structure of the function L in Lemma 14, there is in fact an algorithm that has L as its functional view.

Example 16. A corresponding algorithm for the function L in Example 15 is

1. choose box 1 w.p. 0.6, and otherwise choose box 2.
2. choose box 3 w.p. 5/11, and otherwise the unchosen box of 1 and 2.
3. choose the last remaining box.

Note especially step 2, where the remaining probability of 6/11 is used to check the unchosen box B from 1 and 2, and indeed, by Observation 11, $(2/11)/0.4 = (3/11)/0.6 = 5/11$, which is the probability of not checking B given that it was not checked up to this point.

We next present Algorithm $\mathtt{A}^\star$, as depicted in Algorithm 1, which, given p, calculates the function L, and randomly chooses boxes so as to get L as its functional view.

Let us start with an informal explanation of Algorithm $\mathtt{A}^\star$. At step $t \geq 1$, the first thing $\mathtt{A}^\star$ does is to calculate the values of $L(x,t)$ for all x, so that it can recreate them with its random choices. For that $\mathtt{A}^\star$ needs to calculate $\alpha(t)$, which, by Lemma 14, means solving the equation:

$$t = C_L(t) = \sum_{x\geq 1}(1 - L(x,t)) = \sum_{x\geq 1}(1 - \min\{1, \alpha(t)q(x)\}). \tag{7}$$

Algorithm 1 Algorithm $\mathtt{A}^\star$ for $k \geq 2$ searchers, as executed by one of these searchers.

```
$\mathcal{I} \leftarrow \emptyset$                                    ▷ set of already opened boxes
$\mathtt{ac}(0) \leftarrow 0$                  ▷ $\mathtt{ac}(t)$ denotes the set of boxes susceptible to be opened at step $t$
for $t = 1$ to $M$ do                              ▷ $M = \max\{x \geq 1 \mid p(x) > 0\}$ can be $\infty$
    $y \leftarrow \mathtt{ac}(t-1)$                 ▷ Calculate the set $\mathtt{ac}(t)$ of "active" boxes at step $t$
    while $\left(y < M \text{ and } \sum_{x=1}^{y+1}(1 - q(x)/q(y)) \leq t\ \right)$ do $y \leftarrow y+1$      ▷ $q(x) = p(x)^{-\frac{1}{k-1}}$
    $\mathtt{ac}(t) \leftarrow y$                     ▷ The search space is $\{1, \ldots, \mathtt{ac}(t)\}$ at step $t$
    $\alpha(t) \leftarrow (\mathtt{ac}(t) - t)/\sum_{x=1}^{\mathtt{ac}(t)} q(x)$                    ▷ Calculate $\alpha(t)$
    for $x = 1$ to $\mathtt{ac}(t)$ do                         ▷ Set the probability of choosing $x$
        if $x \notin \mathcal{I}$ then             ▷ for boxes which have not been opened at previous steps
            if $x \leq \mathtt{ac}(t-1)$ then
                $\varphi(x) \leftarrow 1 - \alpha(t)/\alpha(t-1)$        ▷ all these boxes get same chance being opened
            else
                $\varphi(x) \leftarrow 1 - \alpha(t)q(x)$       ▷ new active boxes get decreasing chances of being opened
    pick $x \in \{1, \ldots, \mathtt{ac}(t)\} \setminus \mathcal{I}$ according to the probability distribution $\varphi$      ▷ Choose one index
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{x\}$                                 ▷ remember this index
    open box $B_x$                                      ▷ look for treasure
```

To calculate $\alpha(t)$, the first thing to do is to figure out which indices x actually contribute to the sum above. Say box x is *active* at time t if $L(x,t) < 1$. As L is non-decreasing in x, there is some $\mathtt{ac}(t)$, s.t. the set of active boxes at time t is $\{1, \ldots, \mathtt{ac}(t)\}$. To calculate $\mathtt{ac}(t)$, $\mathtt{A}^\star$ gradually decreases $\alpha(t)$, while keeping the column requirement satisfied. A box x is active when $\alpha(t) < 1/q(x)$, and so to see which box is active, $\mathtt{A}^\star$ needs only to check $\alpha(t) = 1/q(1), 1/q(2), \ldots$. Once $\mathtt{ac}(t)$ is found, solving Eq. (7) and finding $\alpha(t)$ is straightforward.

Once $L(x,t)$ is calculated, $\mathtt{A}^\star$ randomly chooses a box to check according to the functional view L, using the fact that, up to step t, the probability that box x was not checked is $L(x, t-1)$. If a box was not active, and now is, then clearly it should be checked with probability $1 - q(x)\alpha(t)$. If it was already active, then it should change from $q(x)\alpha(t-1)$ to $q(x)\alpha(t)$, which, by Observation 11, means it should be checked with probability $1 - \alpha(t)/\alpha(t-1)$.

Remark. As an interesting side note, observe that at each step, all previously active yet unchecked boxes get the same probability of being checked in $\mathtt{A}^\star$. Moreover, this probability does not depend at all at the previous choices made by the algorithm. This point sounds counter-intuitive from a Bayesian point of view, as we would expect a rescaling of the probabilities that differs according to the history we have already seen. Another important point is that $\mathtt{A}^\star$ has at each step a finite set of boxes to choose from. As p goes to 0, q goes to infinity, and so if there are an infinite number of active boxes, then α must be 0, but that means that all boxes were surely checked. Lastly, note that in the case of the uniform distribution of Section 4.1, as $q(x)$ is equal for all boxes, a searcher running $\mathtt{A}^\star$ will at the first step choose among them uniformly, and continue to do so at each step, choosing from those that it did not check yet. Thus, $\mathtt{A}^\star$ is indeed the natural algorithm for this distribution as analysed there.

We now show that $\mathtt{A}^\star$ is well defined, and is optimal.

Theorem 17. *$\mathtt{A}^\star$ is well defined, and, for every distribution p, every number of searchers $k \geq 2$, and every non-coordinating algorithm A, we have $\mathbb{T}_{\mathtt{avg}}(\mathtt{A}^\star) \leq \mathbb{T}_{\mathtt{avg}}(A)$.*

Proof. Note that $y \leq \mathtt{ac}(t)$ if and only if $\alpha(t) < 1/q(y)$, and so, to calculate $\mathtt{ac}(t)$, it is enough to check values for $\alpha(t)$ that are equal to $1/q(y)$ for $y > \mathtt{ac}(t-1)$. Once we know $\mathtt{ac}(t)$, by Lemma 14:

$$t = \sum_{1\le x\le \mathtt{ac}(t)} (1-\alpha(t)q(x)).$$

Solving this for $\alpha(t)$ is what the algorithm does. To show that the next part of $\mathtt{A}^\star$ is at all valid, we show that the probabilities of each step add up to at most 1. The number of boxes that were already active at step $t-1$, and were not checked yet at step t is $\mathtt{ac}(t-1)-(t-1)$. So, summing all the probabilities of the different boxes, we get

$$(\mathtt{ac}(t-1)-t+1)\left(1-\frac{\alpha(t)}{\alpha(t-1)}\right) + \sum_{\mathtt{ac}(t-1)<x\le \mathtt{ac}(t)} (1-\alpha(t)q(x)). \tag{8}$$

Again, by Lemma 14, we get

$$\sum_{1\le x\le \mathtt{ac}(t-1)} \alpha(t-1)q(x) = t-1 \quad\Longrightarrow\quad \sum_{1\le x\le \mathtt{ac}(t-1)} \alpha(t-1)q(x) = \mathtt{ac}(t-1)-t+1.$$

Plugging this is Eq. (8), the sum of the probabilities is

$$\sum_{x=1}^{\mathtt{ac}(t-1)} (\alpha(t-1)-\alpha(t))q(x) + \sum_{x=\mathtt{ac}(t-1)}^{\mathtt{ac}(t)} 1-\alpha(t)q(x) = \sum_{x=1}^{\mathtt{ac}(t)}(1-\alpha(t)q(x)) - \sum_{x=1}^{\mathtt{ac}(t-1)} (1-\alpha(t-1)q(x)).$$

By Lemma 14 the first sum is t, and the second is $t-1$, and so the sum of probabilities is indeed 1.

It remains to show that indeed the functional view of $\mathtt{A}^\star$ is L. This is proved by induction on t. Let N be the functional view of $\mathtt{A}^\star$. For $t=0$, $L(x,1)=N(x,1)$ for all $x\ge 1$. Assume equality for $t-1$, and we prove it for t. For $x\le \mathtt{ac}(t-1)$, we have $N(x,t-1)=L(x,t-1)=\alpha(t-1)q(x)$. Using Observation 11:

$$N(x,t) = N(x,t-1)\cdot\frac{\alpha(t)}{\alpha(t-1)} = \alpha(t-1)q(x)\cdot\frac{\alpha(t)}{\alpha(t-1)} = L(x,t).$$

For $\mathtt{ac}(t-1) < x \le \mathtt{ac}(t)$, $N(x,t)=L(x,t)$ is straightforward. It follows that $\mathtt{A}^\star$ is well defined. The optimality of $\mathtt{A}^\star$ directly follows from Lemma 14. □

5 Optimal competitiveness

The purpose of this section is to show that $\mathtt{A}_{order}$ has optimal competitive ratio among order-invariant algorithms, with respect to $\mathtt{A}_{coord}$, by proving the following result:

Theorem 18. *For every $k\ge 1$, and every order-invariant non-coordinating algorithm A, if there exist b and c such that*

$$\mathbb{T}_{\mathtt{avg}}(A) \le c\,\mathbb{T}_{\mathtt{avg}}(\mathtt{A}_{coord}) + b \text{ or } \mathbb{T}_{\mathtt{worst}}(A) \le c\,\mathbb{T}_{\mathtt{worst}}(\mathtt{A}_{coord}) + b,$$

then $c \ge 4\left(1-\frac{1}{k+1}\right)^2$.

The remaining of this section is entirely dedicated to the proof of Theorem 18

5.1 Preliminaries

Instead of the set of functions in $\mathcal{V}$ defined in Section 4.2, we consider a more general class of functions. For a measurable set X, we define

$$\mathcal{F}(X) = \{N : X \times \mathbb{N} \to [0,1] \mid N(\cdot, t) \text{ is measurable for every fixed } t \in \mathbb{N}\}.$$

For a function $N \in \mathcal{F}(X)$, we say that N satisfies the *column requirement* if, for every $t \geq 0$,

$$C_N(t) = \int_X (1 - N(x,t))\,\mathrm{d}x \leq t.$$

Such a function N is called *valid*, and $\mathcal{V}(X)$ is the set of all valid functions in $\mathcal{F}(X)$. Given an integer $k \geq 2$, and some measurable function $\rho : X \to [0, \infty)$, let us define

$$\mathbb{U}(N) = \sum_{t=0}^{\infty} \int_X \rho(x) N(x,t)^k \,\mathrm{d}x.$$

This definition of $\mathbb{U}$ for functions N is equivalent of $\mathbb{T}_{\mathtt{avg}}$ for algorithms A, but is "unnormalized", as ρ is not necessarily a distribution. The following result shows a connection between algorithms and valid functions.

Lemma 19. *For every non-increasing distribution p on $\mathbb{N}^+$, and every algorithm A for $k \geq 2$ searchers, there is a function $L \in \mathcal{V}([1,\infty))$ such that $\mathbb{U}(L) \leq \mathbb{T}_{\mathtt{avg}}(A)$, where $\rho : [1,\infty] \to [0,\infty)$ is any non-increasing measurable function that agrees with p on integer values.*

Proof. For every $x \in [1, +\infty)$, and $t \in \mathbb{N}$, let us define $L(x,t) = N(\lfloor x \rfloor, t)$ where N is the functional view of A. For any $t \in \mathbb{N}$, we have

$$C_L(t) = \int_1^{\infty} (1 - N(\lfloor x \rfloor, t))\,\mathrm{d}x = \sum_{x=1}^{\infty} (1 - N(x,t)) = C_N(t) \leq t.$$

So L satisfies the column requirement. Next,

$$\begin{aligned}
\mathbb{U}(L) &= \sum_{t \geq 0} \int_1^{\infty} \rho(x) N(\lfloor x \rfloor, t)^k \,\mathrm{d}x \\
&\leq \sum_{t \geq 0} \int_1^{\infty} p(\lfloor x \rfloor) N(\lfloor x \rfloor, t)^k \,\mathrm{d}x \\
&= \sum_{t \geq 0} \sum_{x=1}^{\infty} p(x) N(x,t)^k \\
&= \mathbb{T}_{\mathtt{avg}}(A),
\end{aligned}$$

as claimed. □

Lemma 19 shows that lower bounding the "running time" of functions in $\mathcal{V}([1,\infty))$ will lower bound the running time of algorithms.

5.2 The Plan

The plan is to show that for every $\epsilon > 0$ there is some distribution p_ϵ on $\mathbb{N}^+$ such that for any algorithm A,

$$\liminf_{\epsilon \to 0} \frac{\mathbb{T}_{\texttt{avg}}(A)}{\mathbb{T}_{\texttt{avg}}(\texttt{A}_{coord})} \geq 4\left(1 - \frac{1}{k+1}\right)^2,$$

and $\lim_{\epsilon\to 0} \mathbb{T}_{\texttt{avg}}(\texttt{A}_{coord}) = \infty$. This will prove the theorem. So, let us fix some $\epsilon > 0$, and set

$$I = \left(\sum_{x=1}^{\infty} \frac{1}{x^{2+\epsilon}}\right)^{-1} \quad \text{and} \quad J = \left(\sum_{x=1}^{\infty} \frac{1}{x^{1+\epsilon}}\right)^{-1}.$$

Also set $p(x) = p_\epsilon(x) = I/x^{2+\epsilon}$. We have

$$\mathbb{T}_{\texttt{avg}}(\texttt{A}_{coord}) = \sum_{x=1}^{\infty} \frac{I}{x^{2+\epsilon}} \left\lceil \frac{x}{k} \right\rceil \leq \sum_{x=1}^{\infty} \frac{I}{x^{2+\epsilon}} \left(\frac{x}{k} + 1\right) = \frac{I}{k} \sum_{x=1}^{\infty} \frac{1}{x^{1+\epsilon}} + \sum_{x=1}^{\infty} \frac{I}{x^{2+\epsilon}} = \frac{I}{kJ} + 1 < \infty. \tag{9}$$

However, as ϵ tends to 0, I tends to some constant, and J goes to 0, hence $\mathbb{T}_{\texttt{avg}}(\texttt{A}_{coord})$ tends to infinity, as desired.

Define $\rho : [1,\infty) \to [0,\infty)$ by $\rho(x) = I/x^{2+\epsilon}$. So ρ agrees with p_ϵ on $x \in \mathbb{N}^+$. By Lemma 19, there is some $L \in \mathcal{V}([1,\infty))$ such that $\mathbb{U}(L) \leq \mathbb{T}_{\texttt{avg}}(A)$. Therefore, it is enough to prove that for any $N \in \mathcal{V}([1,\infty))$,

$$\liminf_{\epsilon \to 0} \frac{kJ}{I} \mathbb{U}(N) \geq 4\left(1 + \frac{1}{k+1}\right)^2, \tag{10}$$

where the $+1$ in Eq. (9) is omitted as it is insignificant in the limit. The question is then: among all functions N satisfying

$$\int_1^\infty (1 - N(x,t))\, \mathrm{d}x \leq t \text{ for every } t \in \mathbb{N}, \tag{11}$$

what is the minimal value of

$$\mathbb{U}(N) = \sum_{t=0}^{\infty} \int_1^\infty \rho(x) N(x,t)^k \, \mathrm{d}x \text{ ?} \tag{12}$$

5.3 Using the Presentation Lemma

To find the N that minimizes $\mathbb{U}(N)$ as in Eq. (12) it is enough to find an N that minimizes the inner integral for each t separately while obeying the restriction in Eq. (11). Let us again use Lemma 13 for each t separately. Fix such a $t \in \mathbb{N}$, set $X = [1,\infty]$ with the standard measure, $c = \rho$, and $T = t$. By the Presentation Lemma 13, an N minimizing the inner integral is

$$N(x,t) = \min\left\{1, \alpha\rho(x)^{-\frac{1}{k-1}}\right\} \tag{13}$$

where α is a function of t (and yet, for readability, we don't write $\alpha(t)$). To check the condition of the lemma, take $h(x) = 1$. In this case, $\int_1^\infty (1 - h(x))\, \mathrm{d}x = 0 \leq t$, and

$$\int_1^\infty \rho(x) h(x)^k \, \mathrm{d}x = \int_1^\infty \frac{I}{x^{2+\epsilon}} \, \mathrm{d}x < \infty,$$

as desired. From this point onwards, N will refer to this specific function.

To calculate α, we use what we know from Eq. (13) about what N looks like, and we rely on the refinement of Eq. (11) given by the Presentation Lemma, i.e., that for all t, $\int_1^\infty (1 - N(x,t))\, \mathrm{d}x = t$. Again, fix a t and examine the function $f(x) = \alpha\rho(x)^{-1/(k-1)}$. We observe the following:

- α cannot be zero, as otherwise $N(x,t) = f(x) = 0$ for all x, and then the constraint of Eq. (11) is not satisfied (no matter what t is).
- ρ is strictly decreasing in x, and it goes to 0 as x goes to infinity. Therefore, f is strictly increasing, and goes to infinity.
- Taking $\alpha = \rho(1)^{1/(k-1)}$, means that $f(1) = 1$, and, by monotonicity, $N(x,t) = \min\{1, f(x))\} = 1$ for all $x \geq 1$. This α clearly satisfies the constraint of Eq. (11), and the minimality guaranteed by Lemma 13 means that the "real" α is at most this value. So $f(1) \leq 1$.
- From the above, and since f is continuous, there is some unique $\gamma \geq 1$ (which is a function of t), where $f(\gamma) = 1$. Such a γ satisfies
$$\alpha\rho(\gamma)^{-\frac{1}{k-1}} = 1 \quad \iff \quad \alpha = \rho(\gamma)^{\frac{1}{k-1}}.$$

Since $N(x,t) = \min\{1, f(x)\}$, it follows that $N(x,t)$ is monotonically increasing until $x = \gamma$, and it remains constant equal to 1 from that point on. By Lemma 13, the constraint of Eq. (11) is an equality. That is, we have

$$t = \int_1^\infty (1 - N(x,t))\,\mathrm{d}x = \int_1^\gamma (1 - \alpha\rho(x)^{-\frac{1}{k-1}})\,\mathrm{d}x = \gamma - 1 - \int_1^\gamma \alpha\rho(x)^{-\frac{1}{k-1}}\,\mathrm{d}x.$$

It follows that

$$\begin{aligned} \gamma - t - 1 &= \alpha \int_1^\gamma \rho(x)^{-\frac{1}{k-1}}\,\mathrm{d}x && (14)\\ &= \rho(\gamma)^{\frac{1}{k-1}} \int_1^\gamma \rho(x)^{-\frac{1}{k-1}}\,\mathrm{d}x = \int_1^\gamma \left(\frac{\rho(\gamma)}{\rho(x)}\right)^{\frac{1}{k-1}} \mathrm{d}x. && (15) \end{aligned}$$

This restriction on γ, if solved, will give us α. Let us first simplify the inner integral of $\mathbb{U}(N)$ as follows:

$$\begin{aligned} \int_1^\infty \rho(x)N(x,t)^k\,\mathrm{d}x &= \int_1^\gamma \rho(x)\alpha^k\rho(x)^{-\frac{k}{k-1}}\,\mathrm{d}x + \int_\gamma^\infty \rho(x)\,\mathrm{d}x \\ &= \alpha^{k-1}\int_1^\gamma \alpha\rho(x)^{-\frac{1}{k-1}}\,\mathrm{d}x + \int_\gamma^\infty \rho(x)\,\mathrm{d}x \qquad (16)\\ &= \rho(\gamma)(\gamma - t - 1) - R(\gamma), \end{aligned}$$

where we use Eq. (14) and the fact that $\alpha^{k-1} = \rho(\gamma)$ for the last equality, and we denote $R(\gamma) = \int_\gamma^\infty \rho(x)\,\mathrm{d}x$ the indefinite integral. (Note that $R(\infty) = 0$ because of the way ρ is defined).

5.4 Approximations and Calculations

Denote $a = 2 + \epsilon$, and let us find γ from Eq. (15). We have

$$\begin{aligned} \gamma - t - 1 &= \int_1^\gamma \left(\frac{\rho(\gamma)}{\rho(x)}\right)^{\frac{1}{k-1}} dx = \int_1^\gamma \left(\frac{x}{\gamma}\right)^{\frac{a}{k-1}} dx = \gamma \int_{\frac{1}{\gamma}}^1 x^{\frac{a}{k-1}}\,dx \\ &= \gamma \frac{1}{1 + \frac{a}{k-1}} \left[x^{1+\frac{a}{k-1}}\right]_{\frac{1}{\gamma}}^1 = \gamma \frac{k-1}{a+k-1}\left(1 - \gamma^{-\frac{a+k-1}{k-1}}\right) \end{aligned}$$

Since $\gamma \geq 1$, and $\frac{a+k-1}{k-1} > 1$, it follows that

$$0 < \gamma^{-\frac{a+k-1}{k-1}} < \frac{1}{\gamma},$$

and therefore

$$\gamma \frac{k-1}{a+k-1} > \gamma - t - 1 > \gamma \frac{k-1}{a+k-1} \left(1 - \frac{1}{\gamma}\right).$$

The left hand side in the above inequality bounding can be rewritten as

$$\gamma \frac{a}{a+k-1} < t+1 \quad \Longrightarrow \quad \gamma < \frac{a+k-1}{a}(t+1).$$

Similarly, the right hand side can be rewritten

$$\gamma - t - 1 > \frac{k-1}{a+k-1}(\gamma - 1) \quad \Longrightarrow \quad (\gamma-1)\frac{a}{a+k-1} > t \quad \Longrightarrow \quad \gamma > \frac{a+k-1}{a} t + 1.$$

Now examining Eq. (16), we get

$$\begin{aligned}
\int_1^\infty \rho(x) N(x,t)^k \, \mathrm{d}x &= \rho(\gamma)(\gamma - t - 1) - R(\gamma) = I\left(\frac{\gamma - t - 1}{\gamma^a} + \frac{\gamma^{1-a}}{a-1}\right) \\
&= \frac{I}{\gamma^a}\left(\gamma - t - 1 + \frac{\gamma}{a-1}\right) = \frac{I}{\gamma^a}\left(\frac{a}{a-1}\gamma - t - 1\right) \\
&\geq \frac{I}{\gamma^a}\left(\frac{a}{a-1}\left(\frac{a+k-1}{a}t + 1\right) - t - 1\right) = \frac{I}{(a-1)\gamma^a}(kt+1) \\
&= \frac{Ik}{a-1} \cdot \frac{t + \frac{1}{k}}{\gamma^a} \geq \frac{Ik}{a-1} \cdot \frac{t + \frac{1}{k}}{\left(\frac{a+k-1}{a}(t+1)\right)^a} = \frac{Ik}{a-1}\left(\frac{a}{a+k-1}\right)^a \cdot \frac{t + \frac{1}{k}}{(t+1)^a}.
\end{aligned}$$

Therefore,

$$\mathbb{U}(N) \geq \frac{Ik}{a-1}\left(\frac{a}{a+k-1}\right)^a \sum_{t=0}^{\infty} \frac{t + \frac{1}{k}}{(t+1)^a}.$$

Focusing on the sum only, and shifting it by 1, we get:

$$\sum_{t=1}^{\infty} \frac{t - \frac{k-1}{k}}{t^a} = \sum_{t=1}^{\infty} \frac{1}{t^{a-1}} - \frac{k-1}{k} \sum_{t=1}^{\infty} \frac{1}{t^a} \geq \frac{1}{J} - \frac{k-1}{k} \sum_{t=1}^{\infty} \frac{1}{t^2} = \frac{1 - o(1)}{J},$$

where the term $o(1)$ goes to 0 as $a \to 2$. (This is because the second sum converges, and $1/J$ goes to infinity). Plugging this back in $\mathbb{U}(N)$, we get

$$\mathbb{U}(N) \geq (1 - o(1)) \frac{Ik}{J(a-1)} \left(\frac{a}{a+k-1}\right)^a.$$

Coming back to Eq. (10), we obtain

$$\liminf_{a \to 2} \frac{kJ}{I} \mathbb{U}(N) \geq \lim_{a \to 2} \frac{k^2}{a-1}\left(\frac{a}{a+k-1}\right)^a = \left(\frac{2k}{k+1}\right)^2 = 4\left(1 - \frac{1}{k+1}\right)^2.$$

This concludes the proof of Theorem 18.

6 Conclusion

In this paper, we have solved the *non-coordinating* Bayesian search problem by designing the *optimal* non-coordinating algorithm $\mathtt{A}^\star$. Given any distribution p over the integers, and given k searchers, $\mathtt{A}^\star$ executes a number of queries by these k searchers which is minimum in expectation among all non-coordinating algorithms for p. We have also designed the *order-invariant* non-coordinating algorithm $\mathtt{A}_{order}$. The latter is far simpler than $\mathtt{A}^\star$, and has optimal competitive ratio with respect to the best coordinating algorithm, $\mathtt{A}_{coord}$, among all non-coordinating order-invariant algorithms. This competitive ratio turns out to be at most 4, demonstrating that non-coordinating algorithms are a viable alternative to coordinating algorithms, especially in a context where searchers can crash or slightly misbehave. This also holds for *linear* search, where the order of importance on the boxes is given a priori, and the treasure is placed at an arbitrary position.

A few questions remain open, as far as Bayesian search is concerned. In particular, we know that $\mathtt{A}^\star$ is optimal, which implies that it cannot performs worse than $\mathtt{A}_{order}$, whose running time is essentially $4(1-\frac{1}{k+1})^2$ times the one of $\mathtt{A}_{coord}$ for k searchers. However, we do not know if $\mathtt{A}^\star$ is often or rarely significantly better than $\mathtt{A}_{order}$. For the uniform distribution over a finite domain, we have seen that $\mathtt{A}^\star$ performs roughly twice faster than $\mathtt{A}_{order}$, but we do not know whether such a factor 2 acceleration holds for many distributions. (It does not hold for all distributions, as illustrated by the distributions p_ϵ described in Section 5.2). Comparing $\mathtt{A}_{order}$ and $\mathtt{A}^\star$ for a given distribution p is an important issue because $\mathtt{A}^\star$ is significantly more complex to implement than $\mathtt{A}_{order}$. The question whether implementing $\mathtt{A}^\star$ is worth the effort in terms of performances with respect to $\mathtt{A}_{order}$ is therefore an important open problem in the context of non-coordinating Bayesian search.

More generally, we have focussed our interest on non-coordinating search algorithms motivated by their inherent fault-tolerance. The price to pay in terms of performances is rather limited (again, for k searchers, at most $4(1-\frac{1}{k+1})^2$ times the optimal performance with full coordination). An interesting, and non-trivial question is to find efficient *and* robust algorithms that are allowed to coordinate. Our non-coordinating algorithms fall under this category, but one may potentially improve the running time by allowing coordination. Finding tradeoffs between efficiency, robustness, and amount of coordination is an intriguing open research direction.

References

[1] Noga Alon, Chen Avin, Michal Koucky, Gady Kozma, Zvi Lotker, and Mark R. Tuttle. Many Random Walks Are Faster Than One. In *Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures*, SPAA '08, pages 119–128, New York, NY, USA, 2008. ACM.

[2] Steve Alpern, Robbert Fokkink, Leszek Gasieniec, Roy Lindelauf, and V.S. Subrahmanian. Search theory: A game theoretic perspective. *Springer*, 2013.

[3] Steve Alpern and Shmuel Gal. The theory of search games and rendezvous. *International Series in Operations Research & Management Science, Springer*, 2003.

[4] R.A. Baezayates, J.C. Culberson, and G.J.E. Rawlins. Searching in the Plane. *Inf. Comput.*, 106(2):234–252, October 1993.

[5] A. Beck. On the linear search problem. *Israel J. of Math*, 2(4):221– 228, 1964.

[6] David Blackwell. Notes on dynamic programming. *Unpublished notes, University of California, Berkeley*, 1962.

[7] Lihi Cohen, Yuval Emek, Oren Louidor, and Jara Uitto. Exploring an infinite space with finite memory scouts. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 207–224, 2017.

[8] Colin Cooper, Alan M. Frieze, and Tomasz Radzik. Multiple Random Walks in Random Regular Graphs. *SIAM J. Discrete Math.*, 23(4):1738–1761, 2009.

[9] Jurek Czyzowicz, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, and Jaroslav Opatrny. Search on a line with faulty robots. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 405–414, New York, NY, USA, 2016. ACM.

[10] Shantanu Das. Mobile agents in distributed computing: Network exploration. *Bulletin of the European Association for Theoretical Computer Science (EATCS), No. 109*, pages 54–69, 2013.

[11] Assaf David and Zamir Shmuel. Optimal sequential search: A bayesian approach. *The Annals of Statistics*, 13(3):1213–1221, 1985.

[12] Klim Efremenko and Omer Reingold. How Well Do Random Walks Parallelize? In Irit Dinur, Klaus Jansen, Joseph Naor, and José Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, pages 476–489. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[13] Robert Elsässer and Thomas Sauerwald. Tight bounds for the cover time of multiple random walks. *Theor. Comput. Sci.*, 412(24):2623–2641, 2011.

[14] Yuval Emek, Tobias Langner, Jara Uitto, and Roger Wattenhofer. Solving the ANTS problem with asynchronous finite state machines. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 471–482, 2014.

[15] Ofer Feinerman and Amos Korman. Memory Lower Bounds for Randomized Collaborative Search and Implications for Biology. In *Distributed Computing - 26th International Symposium, DISC 2012, Salvador, Brazil, October 16-18, 2012. Proceedings*, pages 61–75, 2012.

[16] Ofer Feinerman, Amos Korman, Zvi Lotker, and Jean-Sébastien Sereni. Collaborative search on the plane without communication. In *ACM Symposium on Principles of Distributed Computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012*, pages 77–86, 2012.

[17] Helmut Finner. A Generalization of Holder's Inequality and Some Probability Inequalities. *The Annals of Probability*, 20(4):1893–1901, October 1992.

[18] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by oblivious mobile robots. *Morgan & Claypool Publishers*, 2012.

[19] Ming-Yang Kao, John H. Reif, and Stephen R. Tate. Searching in an Unknown Environment: An Optimal Randomized Algorithm for the Cow-path Problem. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 441–447, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[20] Christoph Lenzen, Nancy A. Lynch, Calvin C. Newport, and Tsvetomira Radeva. Trade-offs between selection complexity and performance when searching the plane without communication. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 252–261, 2014.

[21] Chew Milton, C. A sequential search procedure. *The Annals of Mathematical Statistics*, 38(2):494–502, 1967.

[22] University of California Berkeley. Boinc. `https://boinc.berkeley.edu/`, 2017.

[23] Giuseppe Prencipe. Autonomous mobile robots: A distributed computing perspective. *Algorithms for Sensor Systems - 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics, ALGOSENSORS 2013, Sophia Antipolis, France, September 5-6, 2013, Revised Selected Papers*, pages 6–21, 2013.

[24] Lawrence Stone, D. Theory of optimal search, 2nd edition. *Topics in Operations Research Series*, 2001.

APPENDIX

A Proof of Lemma 3.1

Lemma 8 ***(restated).*** *For any integers $b \geq a \geq 1$, and every real ϕ with $0 < \phi \leq 1$, $\prod_{i=a}^{b} \frac{i}{i+\phi} \leq \left(\frac{a}{b}\right)^{\phi}$.*

Let $b \geq 1$. The proof is by induction on a, from $a = b$ down to $a = 1$. If $a = b$, then $a/(a+\phi) \leq 1$, as required. Assuming the result holds for $a+1$, we wish to prove that it holds for a. The l.h.s. is:

$$\prod_{i=a}^{b} \frac{i}{i+\phi} = \frac{a}{a+\phi} \cdot \prod_{i=a+1}^{b} \frac{i}{i+\phi} \leq \frac{a}{a+\phi} \cdot \left(\frac{a+1}{b}\right)^{\phi}.$$

So the aim is to prove:

$$\frac{a}{a+\phi} \cdot \left(\frac{a+1}{b}\right)^{\phi} \leq \left(\frac{a}{b}\right)^{\phi} \iff \frac{a}{a+\phi} \leq \left(\frac{a}{a+1}\right)^{\phi} \iff \left(\frac{a+\phi}{a}\right)^{\frac{1}{\phi}} \geq \frac{a+1}{a}.$$

Take $y = \frac{1}{a} \leq 1$ and $x = \frac{1}{\phi} \geq 1$. The above is equivalent to:

$$\left(1 + \frac{y}{x}\right)^{x} \geq 1 + y.$$

It is enough to show that the left side is increasing with x when $x \geq 1$. It is increasing iff $(1 + y/x)^{x/y}$ is increasing. Setting $z = y/x \leq 1$, as z is decreasing in x, the question is whether $(1+z)^{1/z}$ is decreasing, which is the same as showing that $\frac{1}{z}\ln(1+z)$ is decreasing. Deriving, we want

$$-\frac{\ln(1+z)}{z^2} + \frac{1}{z(1+z)} < 0 \iff (1+z)\ln(1+z) > z.$$

Using the equality $\ln(1+z) = \int_0^z \frac{1}{1+t}\,\mathrm{d}t$, we get

$$(1+z)\ln(1+z) = \int_0^z \frac{1+z}{1+t}\,\mathrm{d}t > \int_0^z 1\,\mathrm{d}t = z,$$

as desired. □

B Proof of Presentation Lemma

Lemma 13 ***(restated).***(Presentation Lemma) *Let $c : X \to [0, \infty)$ be a measurable function, and let $T \geq 0$. If there is some $h \in V(T)$ such that $\int c\, h^k \,\mathrm{d}\mu < \infty$, then $\alpha = \min\{\beta \geq 0 \,|\, f_{c,\beta} \in V(T)\}$ exists, and, for every $g \in V(T)$, $\int c\, f_{c,\alpha}^k \,\mathrm{d}\mu \leq \int c\, g^k \,\mathrm{d}\mu$. Furthermore, if $\alpha \neq 0$ then $\int (1 - f_{c,\alpha})\,\mathrm{d}\mu = T$.*

Proof. In what follows we will drop the subscript c in f when it is clear from context. We start by a sequences of preliminary remarks, that we label for further references.

20. Note that all of the functions below are measurable, either by definition, or by straightforward proof. Also, as all of these functions are positive, they all have a defined Lebesgue integral, although its value may be ∞.

21. The fact that α can be taken to be the minimal among all β's assuming we have already proved that there is some $f_\beta \in V(T)$ that is optimal in the above respect is quite simple: $\alpha = \min\{\beta \geq 0 \mid f_\beta \in V(T)\}$, and from the assumption, this set is not empty. By monotonicity, if $\beta < \beta'$ then $\int c f_\beta^k \,\mathrm{d}\mu \leq \int c f_{\beta'}^k \,\mathrm{d}\mu$, so all what remains to be shown is that this minimum exists. If not, then there is some sequence $\{\beta_n\}_{n=1}^\infty$ that approaches the infimum α. By the definitions, f_{β_n} converges pointwise to f_α. By Fatou's lemma:

$$\int (1 - f_\alpha) \,\mathrm{d}\mu \leq \liminf_{n\to\infty} \int (1 - f_{\beta_n}) \,\mathrm{d}\mu \leq T$$

So $f_\alpha \in V(T)$, proving this point.

22. Denote by $S = \{x \in X \mid c(x) > 0\}$ the support of c. If $T \geq \mu(S)$, then take $\alpha = 0$. We get $f_0(x) = 0$ on S, and $f_0(x) = 1$ elsewhere, so $\int (1 - f_0) \,\mathrm{d}\mu = \mu(S) \leq T$, and so $f_0 \in V(T)$. Also, $\int c f_0^k \,\mathrm{d}\mu = 0$, and is therefore optimal, so we are done. We will therefore always assume that $T < \mu(S)$. Specifically, $\mu(S) > 0$ and $T < \mu(X)$.

23. For any $\epsilon > 0$, examine the set $Y = \{x \in X \mid c(x) > \epsilon\}$. We claim that $\mu(Y) < \infty$. Indeed, denote $Z = \{x \in X \mid h(x) < 1/2\}$. As $h \in V(T)$, we get

$$T \geq \int (1 - h) \,\mathrm{d}\mu \geq \int_Z \frac{1}{2} \,\mathrm{d}\mu \geq \frac{\mu(Z)}{2}.$$

So $\mu(Z) < \infty$, and therefore $\mu(Y \cap Z) < \infty$. Also, we have

$$\infty > \int c\, h^k \,\mathrm{d}\mu \geq \int_{Y \cap \neg Z} c\, h^k \,\mathrm{d}\mu \geq \frac{\epsilon}{2^k} \cdot \mu(Y \cap \neg Z).$$

Together, this means that $\mu(Y) < \infty$.

24. The last preliminary remark is regarding $\alpha \neq 0$. Assume $\int (1 - f_\alpha) \,\mathrm{d}\mu < T' < T$. The idea here is to find some non-null set where f_α can be slightly decreased, and so still be in $V(T)$, yet improve on the target integral, contradicting f_α's minimality. We first claim that there is some $\epsilon > 0$, and some Y with $0 < \mu(Y) < \infty$, such that for all $x \in Y$, $c(x) > \epsilon$, and $f_\alpha(x) > \epsilon$. Let

$$Y_n = \{x \in X \mid 1/n < c(x) < n\}.$$

As $S = \cup_{n\in\mathbb{N}} Y_n$, and as, by 22, $\mu(S) > 0$, it follows by sigma additivity that there is some n such that $\mu(Y_n) > 0$. By 23, $\mu(Y_n) < \infty$. Setting $Y = Y_n$, and taking $\epsilon < 1/n$ guarantees $c(x) > \epsilon$ as required. Now, for every $x \in Y$, either $f_\alpha(x) = 1$, in which case we are fine since we can assume $\epsilon < 1$, or

$$f_\alpha(x) = \alpha\, c(x)^{-1/(k-1)} > \alpha\, n^{1/(k-1)}.$$

Taking ϵ to be smaller than this value allows us to conclude this part regarding $\alpha \neq 0$. Define $g = f_\alpha$ everywhere, except on Y where it is defined as $g = f_\alpha - \delta$, where $0 < \delta < \epsilon$, and $\delta < (T - T')/\mu(Y)$. This means that $g > 0$ everywhere, and that $g \in V(T)$. Also, by this setting, we get

$$\int c f_\alpha^k \,\mathrm{d}\mu - \int c g^k \,\mathrm{d}\mu = \int_Y c \left(f_\alpha^k - (f_\alpha^k - \delta)^k \right) \mathrm{d}\mu \geq \int_Y \epsilon \delta^k \,\mathrm{d}\mu > 0$$

because when $a > b > 0$, then $(a - b)^k \leq a^k - b^k$. This contradicts the optimality of f_α.

Once we are done with these preliminary remarks, the proof now proceeds by a gradual increase of the generality of the function c that we handle. We start with indicator functions.

(1) *Indicator functions.* Let us assume that $c = \mathbf{1}_A$, the indicator function of some set $A \subseteq X$. By 23, we get $\mu(A) < \infty$, and by 22, we can assume that $T < \mu(A)$. For any $g \in V(T)$:

$$\begin{aligned}\int c\, g^k \,\mathrm{d}\mu = \int_A g^k \,\mathrm{d}\mu \geq \mu(A) \cdot \left(\frac{1}{\mu(A)} \int_A g \,\mathrm{d}\mu \right)^k \\ \geq \frac{1}{\mu(A)^{k-1}} \cdot \left(\mu(A) - \int_A (1-g) \,\mathrm{d}\mu \right)^k \geq \frac{(\mu(A) - T)^k}{\mu(A)^{k-1}},\end{aligned}$$

where we used Jensen's inequality for the case where the total measure is not 1. Let us take

$$\alpha = (\mu(A) - T)/\mu(A).$$

We get $0 < \alpha < 1$, with $f_\alpha(x) = \alpha$ for every $x \in A$, and $f_\alpha(x) = 1$ elsewhere. Also,

$$\int (1 - f_\alpha) \,\mathrm{d}\mu = \int_A (1 - f_\alpha) \,\mathrm{d}\mu = \int_A \frac{T}{\mu(A)} \,\mathrm{d}\mu = T.$$

So $f_\alpha \in V(T)$. Also,

$$\int c f_\alpha^k \,\mathrm{d}\mu = \int_A \alpha^k \,\mathrm{d}\mu = \frac{(\mu(A) - T)^k}{\mu(A)^{k-1}}.$$

It follows that f_α is optimal. (Note that f_α is a constant function on A). We consider separately two cases.

(2) *Simple functions.* Let

$$c = \sum_{i=1}^{n} c_i \mathbf{1}_{X_i},$$

where all $c_i > 0$, and the X_i are pairwise disjoint. By 23, we assume that all the X_i are of finite measure. Given some $g \in V(T)$, let us examine it on each of the X_i's separately. Let

$$T_i = \int_{X_i} (1-g) \,\mathrm{d}\mu.$$

Restricted to X_i, according to the case of indicator functions, there is some constant $g_i \geq 0$, such that

$$\int_{X_i} (1 - g_i) \,\mathrm{d}\mu \leq T_i, \text{ and } \int_{X_i} g_i^k \,\mathrm{d}\mu \leq \int_{X_i} g^k \,\mathrm{d}\mu.$$

Thus, we define

$$g' = \mathbf{1}_Y + \sum_{i=1}^{n} g_i \mathbf{1}_{X_i},$$

where $Y = X \setminus \cup_{i=1}^{n} X_i$. According to the above, we get

$$\int (1 - g') \,\mathrm{d}\mu = \sum_{i=1}^{n} \int_{X_i} (1 - g_i) \,\mathrm{d}\mu \leq \sum_{i=1}^{n} \int_{X_i} (1 - g) \,\mathrm{d}\mu \leq \int (1-g) \,\mathrm{d}\mu \leq T.$$

It follows that $g' \in V(T)$. Also,

$$\int c\, g^k \,\mathrm{d}\mu = \sum_{i=1}^{n} c_i \int_{X_i} g^k \,\mathrm{d}\mu \geq \sum_{i=1}^{n} c_i \int_{X_i} g_i^k \,\mathrm{d}\mu = \int c\, g'^k \,\mathrm{d}\mu.$$

Therefore, g' is a better candidate than g, and we can thus assume that g is constant on each of the X_i's, and hence can be written as

$$g = \sum_{i=1}^{n} g_i \mathbf{1}_{X_i}.$$

Our question can now be viewed as follows. Given $c_1, \ldots, c_n$, and given $\mu_1, \ldots \mu_n > 0$, find $g_1, \ldots, g_n \in [0,1]$ among those satisfying $\sum_{i=1}^{n} \mu_i(1 - g_i) \leq T$, which minimize $\sum_{i=1}^{n} c_i\, g_i^k$. As the solution space is compact, and as the function to minimize are continuous, there exists an optimal solution $g_1, \ldots, g_n$. Take some i such that $1 < i \leq n$. We can rebalance the values of g_1 and g_i as we wish, as long as the sum $\mu_1 g_1 + \mu_i g_i$ remains the same. We use the following claim, whose proof is postponed at the end of the Appendix so that to keep the flow of the current proof.

Claim 25. *Let $k \geq 2$, $c_1, c_2, \mu_1, \mu_2 > 0$, and $m \leq \mu_1 + \mu_2$. The minimal value of $\mu_1 c_1 g_1^k + \mu_2 c_2 g_2^k$, where $g_1, g_2 \in [0,1]$ and $\mu_1 g_1 + \mu_2 g_2 = m$ is achieved only when*

$$g_1 = \min\left\{1, (c_2/c_1)^{\frac{1}{k-1}} \cdot g_2\right\}.$$

According to Claim 25, the values of g_1 and g_i must satisfy:

$$g_i = \min\left\{1, (c_1/c_i)^{\frac{1}{k-1}}\, g_1\right\}.$$

Taking $\alpha = c_1^{1/(k-1)} g_1$, we obtain the desired form $g_i = \min\left\{1, \alpha c_i^{-1/(k-1)}\right\}$ which concludes this case.

(3) *The general case.* Let $\{c_n\}_{n=1}^{\infty}$ be a non-decreasing family of simple functions that have c as their pointwise limit. According to the simple function case, for each n, there is some α_n such that the function $f_n = f_{c_n,\alpha_n}$ yields minimal $\int c_n f_n^k \,\mathrm{d}\mu$ among all functions of $V(T)$. If this sequence $(\alpha_n)_{n\geq 1}$ is unbounded, we can keep only a sub-sequence where $\alpha_n \to \infty$, and define $f(x) = \lim_{n\to\infty} f_n(x) = 1$ everywhere. Otherwise we can keep only a converging sub-sequence of $(\alpha_n)_{n\geq 1}$, and denote its limit by α. Now, let us define the function $f(x) = f_{c,\alpha}(x) = \lim_{n\to\infty} f_n(x)$. Either way the pointwise limit of the f_n's exists, and we denote it by f. Let us examine the sequence of functions $(1 - f_n)_{n\geq 1}$. Each element of this sequence satisfies $\int (1 - f_n) \,\mathrm{d}\mu \leq T$. Therefore, by Fatou lemma:

$$\int (1 - f) \,\mathrm{d}\mu \leq \liminf_{n\to\infty} \int (1 - f_n) \,\mathrm{d}\mu \leq T.$$

Therefore, $f \in V(T)$. Moreover, the function cf^k is the pointwise limit of $c_n f_n^k$, and hence, as f_n is optimal for c_n, we get

$$\int c_n f_n^k \,\mathrm{d}\mu \leq \int c_n h^k \,\mathrm{d}\mu \leq \int c h^k \,\mathrm{d}\mu < \infty.$$

So all these integrals are jointly bounded, and thus their lim inf exists. Therefore, by Fatou lemma again, we get

$$\int c f^k \,\mathrm{d}\mu \leq \liminf_{n\to\infty} \int c_n f_n^k \,\mathrm{d}\mu < \infty.$$

Let us assume that there is some g that is better than f. That is, there is some $\delta > 0$ such that:

$$\int c\, g^k \,\mathrm{d}\mu < \int c\, f^k \,\mathrm{d}\mu - \delta.$$

Let us then pick n large enough, and let us use the fact that f_n is optimal for c_n, which yields

$$\int c\, f^k \,\mathrm{d}\mu - \frac{\delta}{2} < \int c_n\, f_n^k \,\mathrm{d}\mu \le \int c_n\, g^k \,\mathrm{d}\mu \le \int c\, g^k \,\mathrm{d}\mu,$$

leading to a contradiction.

The only thing left to show is that $f = f_{c,\alpha}$ for some α. By 22, there is some $\epsilon > 0$ such that the set

$$A = \{x \in X \mid c(x) > \epsilon\}$$

satisfies $\mu(A) > 0$. Moreover, by 23, we also have $\mu(A) < \infty$. If $T < \mu(A)$, then let is pick the function $g(x) = 1 - T/\mu(A)$ on this set and $g(x) = 1$ elsewhere. Clearly $g \in V(T)$. Also,

$$\int c\, \mathbf{1}_X^k \,\mathrm{d}\mu - \int c\, g^k \,\mathrm{d}\mu = \int_A c\, \frac{T}{\mu(A)} \,\mathrm{d}\mu \ge \epsilon T > 0.$$

As f is optimal, it cannot be the function $\mathbf{1}_X$, and this it must be of the required form. If $T > \mu(A)$, then we proceed in the same way, except that we set $g(x) = 0$ on A and $g(x) = 1$ elsewhere. This completes the proof of the Presentation Lemma. □

It just remains to prove the technical Claim 25.

Proof (of Claim 25). Let $c = c_2/c_1$, and $\mu = \mu_2/\mu_1$. Setting $N = m/\mu_1$, we can write the claim equivalently as follows. Assuming $N \le 1 + \mu$, and knowing that $g_1 + \mu g_2 = N$, we claim that the values $g_1, g_2 \in [0,1]$ minimizing $g_1^k + c\mu g_2^k$ satisfy $g_1 = \min\{1, c^{1/(k-1)} g_2\}$. Denoting $g_2 = (N - g_1)/\mu$, we want to minimize:

$$g_1^k + \frac{c}{\mu^{k-1}}(N - g_1)^k.$$

Let us take the derivative w.r.t. g_1, yielding

$$k\left(g_1^{k-1} - \frac{c}{\mu^{k-1}}(N - g_1)^{k-1}\right). \tag{17}$$

This is zero exactly when

$$g_1 = c^{\frac{1}{k-1}} \cdot \frac{N - g_1}{\mu} = c^{\frac{1}{k-1}} g_2,$$

from which we get that

$$g_1 = \frac{c^{\frac{1}{k-1}}}{\mu + c^{\frac{1}{k-1}}} N. \tag{18}$$

Let us now take the second derivative w.r.t. g_1 (the first derivative was expressed in Eq. (17)), yielding

$$k(k-1)\left(g_1^{k-2} + \frac{c}{\mu^{k-1}}(N - g_1)^{k-2}\right).$$

If we consider g_1 in the range $[0, N]$, this second derivative is always strictly positive, meaning that our function is U-shaped. Also, by Eq. (18) the minimum is somewhere in $[0, N]$. Recall that $g_1 \in [0,1]$. If the minimum of the U-shape is in $[0,1]$ then we get the lemma. Otherwise this minimum must be somewhere in $(1, N]$, and so our minimum would be at $g_1 = 1$. Note that the minimum is unique. □