HAL
open science

Parallel Search with no Coordination

Amos Korman, Yoav Rodeh

To cite this version:

Amos Korman, Yoav Rodeh. Parallel Search with no Coordination. 24th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Jun 2017, Porquerolles, France. hal-01523506

HAL Id: hal-01523506

https://inria.hal.science/hal-01523506v1

Submitted on 16 May 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Parallel Search with no Coordination*

Amos Korman[1] and Yoav Rodeh[2]

[1]CNRS and University Paris Diderot, Paris, France, amos.korman@irif.fr
[2]Weizmann Institute, Rehovot, Israel, yoav.rodeh@gmail.com

May 16, 2017

Abstract

We consider a parallel version of a classical Bayesian search problem. k agents are looking for a treasure that is placed in one of the boxes indexed by $\mathbb{N}^+$ according to a known distribution p. The aim is to minimize the expected time until the first agent finds it. Searchers run in parallel where at each time step each searcher can "peek" into a box. A basic family of algorithms which are inherently robust is *non-coordinating* algorithms. Such algorithms act independently at each searcher, differing only by their probabilistic choices. We are interested in the price incurred by employing such algorithms when compared with the case of full coordination.

We first show that there exists a non-coordination algorithm, that knowing only the relative likelihood of boxes according to p, has expected running time of at most $10+4(1+\frac{1}{k})^2T$, where T is the expected running time of the best fully coordinated algorithm. This result is obtained by applying a refined version of the main algorithm suggested by Fraigniaud, Korman and Rodeh in STOC'16, which was designed for the context of linear parallel search.

We then describe an optimal non-coordinating algorithm for the case where the distribution p is known. The running time of this algorithm is difficult to analyse in general, but we calculate it for several examples. In the case where p is uniform over a finite set of boxes, then the algorithm just checks boxes uniformly at random among all non-checked boxes and is essentially 2 times worse than the coordinating algorithm. We also show simple algorithms for Pareto distributions over M boxes. That is, in the case where $p(x) \sim 1/x^b$ for $0 < b < 1$, we suggest the following algorithm: at step t choose uniformly from the boxes unchecked in $\{1, \ldots, \min(M, \lfloor t/\sigma \rfloor)\}$, where $\sigma = b/(b + k - 1)$. It turns out this algorithm is asymptotically optimal, and runs about $2 + b$ times worse than the case of full coordination.

*This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 648032).

1 Introduction

We consider a parallel variant of the classical Bayesian search problem, typically attributed to Blackwell [6]. A treasure is placed in one of the boxes indexed by $\mathbb{N}^+$ according to some known distribution p. As p is known, we can assume that the boxes are ordered so that p is non-increasing. Denote $M = \max\{x \mid p(x) > 0\}$, which can be ∞. There are k agents that search for the treasure, aiming to minimize the expected time until the first one finds it, where looking into a box takes one unit of time. We shall assume that algorithms know the number of searchers k.

If coordination is allowed, a simple application of the rearrangement inequality shows that letting agent i peek into box $(t-1)k+i$ at time t is an optimal algorithm (see Appendix A). Denote this algorithm $\mathtt{A}_{cord}$, and note that its expected running time is $\sum_x p(x)\lceil x/k \rceil$, giving a speedup of essentially k compared to just one searcher. However, as simple as this algorithm is, it 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[1].

A class of search algorithms which is of particular interest is *non-coordinating* algorithms [1, 17]. In such an algorithm, all searchers operate independently, executing the same protocol, differing only in the outcome of the flips of their random coins. With such a strong restriction on the coordination, one cannot expect that many search problems could be efficiently parallelized. However, when such a 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, assume that an oblivious adversary is allowed to crash at most f out of the k searchers at arbitrary points in time during the execution. To overcome the presence of at most f faults, one can simply run the non-coordinating algorithm that is designed for the case of $k-f$ searchers. If the running time of a non-coordinating algorithm without crashes is $T(k)$, then the running time of the new robust algorithm would be at most $T(k-f)$. This is because the correct operation as well as the running time of a non-coordinating algorithm can only improve if more searchers than planned are actually being used. Note that even when coordination is allowed, one cannot expect to obtain robustness at a cheaper price since the number of searchers that remain alive is in the worst case $k-f$.

For an algorithm A, and $k \geq 2$, denote by $\mathbb{T}_k(A, x)$ the expected running time if the treasure is placed at x, when running algorithm A with k searchers. Note that by "running time" we actually mean the expected number of boxes peeked into by each searcher, as we are mostly interested in query complexity. Further, for a distribution p over the boxes, denote the expected time to find the treasure when it is placed in one of the boxes according to p:

$$\mathbb{T}_{p,k}(A) = \sum_x p(x)\mathbb{T}_k(A, x)$$

In this notation, the expected running time of the optimal coordinating algorithm is $\mathbb{T}_{p,k}(\mathtt{A}_{cord})$. We are interested in the connection between these two terms, and specifically in identifying non-coordinating algorithms that minimize the additive and multiplicative factors a and b such that:

$$\mathbb{T}_{p,k}(A) \leq a + b\mathbb{T}_{p,k}(\mathtt{A}_{cord})$$

We remark, for readability's sake, the subscripts above, as well as most subscripts in the text that follows, will be dropped when clear from context. Also, the number of agents $k \geq 2$ will be fixed

[1] It is actually an interesting and non trivial question to find efficient and robust algorithms that are allowed to coordinate [8]. Of course, our non-coordinating algorithms fall under this category, but one may potentially improve the running time by allowing coordination.

and so omitted from formal statements. This will many times go for p as well. Also note that there are distributions where no algorithm can achieve finite running time, such as $p(x) = c/x^2$, where the expected placement of the treasure is unbounded. We shall therefore always assume that $\sum_x p(x)x < \infty$, and so, for example, $\mathbb{T}(\mathtt{A}_{cord})$ is always defined.

1.1 Our Results

We first show that there exists a simple and highly efficient algorithm, denoted $\mathtt{A}_{universal}$, which for large k enjoys a multiplicative factor that tends to 4. In this algorithm, each agent, at phase t, checks two different uniformly chosen boxes of those it did not check yet in $\{1, \ldots, t(k+1)\}$. This algorithm is universal in the sense that it does not depend on the details of the distribution p, and assumes only the knowledge of the relative likelihood of the boxes, that is, their order.

Theorem 1. $\mathbb{T}(\mathtt{A}_{universal}) \leq 10 + 4\left(1 - \frac{1}{k+1}\right)^2 \mathbb{T}(\mathtt{A}_{cord})$

Note that this gives improvement over the trivial one searcher for every k. Even for $k = 2$ we get that for large enough x, this runs at 8/9's the time of the lone searcher.

Algorithm $\mathtt{A}_{universal}$ remembers all the boxes it checked and so needs memory which is linear in its running time. We also consider $\mathtt{A}_{memory}$ which at phase t chooses uniformly two boxes in $\{1, \ldots, tk\}$. This algorithm uses only logarithmic memory in its running time, and for large number of searchers performs almost as well:

Theorem 2. $\mathbb{T}(\mathtt{A}_{memory}) \leq 2 + 4\mathbb{T}(\mathtt{A}_{cord})$

Both algorithms $\mathtt{A}_{universal}$ and $\mathtt{A}_{memory}$ where actually given in [17] to tackle the setting of linear search with an adversarially placed treasure. We note, however, that when applied in our context, the bounds established in [17] only guarantee that the additive term is some unknown, possibly large constant. To prove that this constant is in fact small we had to refine the upper bound analysis of [17], and prove tighter bounds on the Gamma function.

We next present Algorithm $\mathtt{A}^\star$, that given access to the the exact distribution p (and not only the order of the boxes), gives the optimal expected running time:

Theorem 3. *For every non-coordinating algorithm A, $\mathbb{T}(\mathtt{A}^\star) \leq \mathbb{T}(A)$.*

An interesting property satisfied by this algorithm, is that at any time during the execution, all boxes that previously received a positive probability to be checked, are now going to be checked with equal probability.

Calculating the running time of $\mathtt{A}^\star$ can become challenging for specific distributions, and the rest of the paper shows a few interesting examples. A simple one is when p is the uniform distribution over a finite domain. In this case, running $\mathtt{A}^\star$, at each step each agent chooses a box uniformly among those it did not check yet. This natural choice for an algorithm therefore turns out to be optimal, and yields a multiplicative factor of essentially 2 when compared to $\mathtt{A}_{cord}$.

On the other extremity there are exponential distributions. Such distributions strongly concentrate the probability on the first few boxes, and so a good algorithm would invest in optimizing the parallel performance on a constant number of boxes. As we are concerned with non-trivial behavior over many boxes, we turn our attention to investigate Pareto distribution, which spread the distribution more gradually.

Specifically, we consider the family of Pareto distributions over M boxes, thinking of M as large. Here, for some $0 < b < 1$, for all $x \leq M$, $p(x) = I/x^b$, where I is the normalization factor, and $p(x) = 0$ for larger x. While $\mathtt{A}^\star$ is optimal, it is quite complex and difficult to analyse. We

present a simple algorithm $\mathtt{A}_{pareto}$ that is asymptotically optimal. In $\mathtt{A}_{pareto}$, at step t, an agent chooses uniformly from one of the boxes it did not check yet in $\{1, \ldots, \min(M, \lfloor t/\sigma \rfloor)\}$, where $\sigma = b/(b+k-1)$.

Theorem 4. *For* $0 < b < 1$, $\lim_{M\to\infty} \frac{\mathbb{T}_{r_{M,b}}(\mathtt{A}_{pareto})}{\mathbb{T}_{r_{M,b}}(\mathtt{A}_{cord})} = k\sigma(2-\sigma) + \frac{k}{k+1}(2-b)(1-\sigma)^2$. *Furthermore, no non-coordinating algorithm can achieve a better limit bound.*

When b is close to 1, then $\sigma \approx 1/k$ and the factor becomes $(3k-1)/(k+1)$. For $k = 2$ this is 5/3 compared to 16/9 achieved by $\mathtt{A}_{universal}$, and for large k this tends to 3 as opposed to 4. For smaller b's the result is not as clean, but assuming k is large, then $\sigma \approx b/k$, and we get that the ratio is about $2 + b$. This makes sense, as when b approaches 0, the distribution becomes uniform, where we already know that this factor is 2 for large k.

Finally, we note that most of our algorithms are very simple and hence applicable. From the technical point of view, our results illustrate deep connections between the general probabilistic parallel search setting considered here, and the setting of parallel *linear* search studied in [17].

1.2 Related work

The study of parallel search by non-coordinating algorithms has recently been advocated by Fraigniaud, Korman and Rodeh as a simple way to obtain robustness while avoiding communication overheads [17]. The setting therein, however, differs from ours by two fundamental characteristics: First, they assumed that the treasure is placed by an adversary. The second major difference is that they focused on a *linear search* setting (see also [4, 5, 9]), 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. That is, if the treasure is placed in index x, then the running time of the parallel algorithm should be compared to x/k. Although this linear search setting may seem somewhat specific compared to the setting studied in the current paper, it turns out that there are important connections between the two settings, both in terms of techniques and results. See Section 2 for more details.

The case of a single searcher that searches for a randomly pleased treasure has receives significant amount of attention from the communities of statistics, operational research and computer science, see e.g., [6, 11, 20], and has been studied under various settings, including the case that there are different costs associated with queries, that queries can be noisy, and that the target may be mobile, see the book [22]. As we initiate its parallel version, we consider only the most basic form of the problem, yet, we note that most of our results can be extended to the case in which weighted costs are associated with queries.

In general, when it comes to parallel search, most of the literature deals with mobile agents that search graphs of different topologies, and typically employ some form of communication between themselves. The literature on this subject is vast, and some good references can be found, e.g., in [2, 3, 10, 16, 21]. 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*.

Multiple random walkers are a special case of non-coordinating searchers. In a series of papers [1, 7, 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. Non-coordinating searchers have also been studied in the context of the ANTS problem, a parallel variant of the cow-path problem on the grid [4, 19], which was introduced in [14, 15] motivated by applications to central search foraging by desert ants. For example, it was shown in [14, 15] 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.

Finally, BOINC [18] (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 a potential abstraction for projects operated at platforms similar to BOINC with hundreds of thousands distributed searchers.

2 Ordering of Boxes is Known

In [17], the authors consider a somewhat different scenario. The boxes are ordered linearly by some predefined importance, and the treasure is placed in one of them by an adversary. In such a situation, a lone searcher will check the boxes according to their order, and so box x will be checked by time x. They present algorithm $\mathtt{A}_{universal}$, in which each agent, at phase t, checks two different uniformly chosen boxes of those it did not check yet in $\{1, \ldots, t(k+1)\}$. It is shown there that:

$$\limsup_{x\to\infty} \frac{\mathbb{T}(\mathtt{A}_{universal}, x)}{x} = \frac{4k}{(k+1)^2}$$

and that it is in fact optimal in this way. That is, in that setting, it has the best speedup compared to the lone searcher when taking large enough x.

If $\mathtt{A}_{universal}$ would give this result for all x and not only large ones, it will solve the case of a randomly placed treasure with surprising efficiency. All one has to do is set the importance of the boxes according to the likelihood of the treasure being placed there. The following claim is proved in Appendix C.1 via a refined analysis of that done in [17], and shows that the limsup only hides a small additive term:

Claim 5. *For all x, $\mathbb{T}(\mathtt{A}_{universal}, x) \leq 10 + \frac{4k}{(k+1)^2}x$.*

A major ingredient in the proof is the following lemma:

Lemma 6. *For integers $b \geq a \geq 1$, and $0 < \phi \leq 1$, $\prod_{i=a}^{b} \frac{i}{i+\phi} \leq \left(\frac{a}{b}\right)^{\phi}$.*

Using properties of the Gamma function it is easy to see that the two sides of the equation are asymptotically equal, but this is not enough to prove our result as we need the inequality for small a and b as well. Using Claim 5 the following is straightforward:

Theorem 1. $\mathbb{T}(\mathtt{A}_{universal}) \leq 10 + 4\left(1 - \frac{1}{k+1}\right)^2 \mathbb{T}(\mathtt{A}_{cord})$

Proof.

$$\mathbb{T}(\mathtt{A}_{universal}) = \sum_x p(x)\mathbb{T}_k(\mathtt{A}_{universal}, x) \leq 10 + \frac{4k}{(k+1)^2}\sum_x p(x)x$$
$$\leq 10 + \frac{4k^2}{(k+1)^2}\sum_x p(x)\left\lceil\frac{x}{k}\right\rceil = 10 + 4\left(1 - \frac{1}{k+1}\right)^2 \mathbb{T}(\mathtt{A}_{cord})$$

□

At [17], the authors introduce a memory efficient version of $\mathtt{A}_{universal}$, which we present here, slightly altered, as $\mathtt{A}_{memory}$. In it, each agent, at phase t, checks two uniformly chosen boxes of those in $\{1, \ldots, kt\}$. The following is proved in Appendix C.2:

Claim 7. *For all x, $\mathbb{T}(\mathtt{A}_{memory}, x) \leq 2 + 4\lceil \frac{x}{k} \rceil$.*

Note that for $k \leq 4$ this is of no use, as running the trivial one searcher will do better. This claim immediately proves,

Theorem 2. $\mathbb{T}(\mathtt{A}_{memory}) \leq 2 + 4\mathbb{T}(\mathtt{A}_{cord})$

While $\mathtt{A}_{memory}$ is not optimal as $\mathtt{A}_{universal}$ is, as k grows the difference between them grows smaller, and $\mathtt{A}_{memory}$'s simplicity and efficiency make it an outstanding candidate for real life purposes.

3 Exact Distribution is Known

Ignoring the small additive term in Theorem 1, as k grows larger we get that $\mathtt{A}_{universal}$ is about 4 times worse than the best coordinating algorithm. In the remainder of the paper we show it is possible to improve on this if the exact distribution is known.

3.1 Preliminaries

Consider a non-coordinated algorithm A that is running on k agents. Focusing on just one agent, denote by $A(x,t)$ the probability that by time t, box x was not already checked by this agent. Hence, the probability that none of the k agents checked x by time t is $A(x,t)^k$. In fact, as we shall soon see, the information encoded in this functional view of A is all that is needed to assess its running time. First note:

Observation 8. The function corresponding to algorithm A satisfies $A(x,0) = 1$ for all x. Also, for all x and $t \geq 1$:

$$A(x,t) = A(x,t-1) \cdot \mathtt{Pr}\left[\begin{array}{c|c} x \text{ wasn't checked} & x \text{ wasn't checked} \\ \text{at time } t & \text{prior to time } t \end{array}\right]$$

Let us now consider such functions on their own, possibly without a corresponding algorithm. Let[2] $N : \mathbb{N}^+ \times \mathbb{N} \to [0,1]$. For time t, denote:

$$C_N(t) = \sum_x 1 - N(x,t)$$

In the case of an algorithm A, $C_A(t)$ 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 $C_N(t) \leq t$. Also, define the set of *valid* functions as:

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

and so functions corresponding to algorithms are always valid. Finally, the "running time" of N:

$$\mathbb{T}_{p,k}(N) = \sum_x p(x) \sum_t N(x,t)^k = \sum_t \sum_x p(x) N(x,t)^k$$

The sum on t is from 0 to ∞, and these limits will be omitted whenever clear from context. This is clearly defined so that $\mathbb{T}(A)$ is indeed the expected running time of algorithm A, as $\mathbb{T}(A,x) = \sum_t \mathtt{Pr}\left[x \text{ wasn't found by time } t\right] = \sum_t A(x,t)^k$.

[2] The letter N stands for "probability of *not* being checked up to time".

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 a generalized version of the main Lemma of [17] which we prove in Appendix D. At this point we only need a very simple version of the lemma, yet we present it in its full glory, as we will need it later in the paper. The current version improves on the original lemma of [17] as it applies to general measurable functions, instead of only continuous and bounded ones. In addition, the measure theoretic proof is much more elegant and concise than the original one.

3.2 Main Lemma

The notation that follows is in measure theory style. Fix some $k \geq 2$ and let $(X, \mathcal{X}, \mu)$ be a measure space. For $T \geq 0$, denote by $V(T)$ the set of measurable functions $f : X \to [0,1]$ such that $\int 1 - f \, \mathrm{d}\mu \leq T$. For a measurable function $c : X \to [0, \infty)$, and $\alpha \geq 0$ define the function $f_{c,\alpha} : X \to [0,1]$ as:

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

Lemma 9. *For a given c and T as above, if there is some $h \in V(T)$ such that $\int ch^k \, \mathrm{d}\mu < \infty$, then there exists $\alpha \geq 0$, such that $f_{c,\alpha} \in V(T)$, and for every $g \in V(T)$, $\int cf_{c,\alpha}^k \, \mathrm{d}\mu \leq \int cg^k \, \mathrm{d}\mu$. Furthermore, this α is minimal among those satisfying $f_{c,\alpha} \in V(T)$.*

Towards finding the optimal $N \in \mathcal{V}$, fix some t, and then $N \in \mathcal{V}$, means $\sum_x 1 - N(x,t) \leq t$, and the aim is to minimize $\sum_x p(x)N(x,t)^k$. As this can be done for each t completely separately, Lemma 9 comes into play.

Claim 10. *The following function L is in $\mathcal{V}$, and achieves minimal $\mathbb{T}(\cdot)$ over all valid functions.*

$$L_{p,k}(x,t) = \begin{cases} 1 & p(x) = 0 \\ \min(1, \alpha(t)q(x)) & \text{otherwise} \end{cases}$$

Where $q(x) = p(x)^{-\frac{1}{k-1}}$, and for all t, $\alpha(t) \geq 0$ is the minimal such that $L_{p,k} \in \mathcal{V}$.

Proof. Fix t. Setting $X = \mathbb{N}^+$ with the trivial measure $\mu(x) = 1$ for all x, $T = t$ and $c = p$, Lemma 9 gives the values of the optimal N for this specific t. To check the condition of the lemma, take the constant function $h(x) = 1$. Clearly $h \in V(t)$, and $\int ch^k \, \mathrm{d}\mu = \sum_x p(x) = 1 < \infty$. □

The following basically says that L, if thought of as an algorithm, never rechecks a box. See the proof in Appendix B.

Observation 11. For every $t < M$, $C_L(t) = t$, and for $t \geq M$, $L(x,t) = 0$ everywhere.

As an illustration consider a simple example: $k = 2$, $p(1) = 1/2$, $p(2) = 1/3$, and $p(3) = 1/6$. 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 $\alpha(3) = 0$. From these we get the matrix L on the right. Note that Observation 11 holds, as the sum of column t is indeed equal to $M - t$.

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

3.3 Optimal Algorithm

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 8 quickly become complicated for general N. However, it turns out that because of the specific structure L has, there is in fact an algorithm that has it as its function.

For instance, a corresponding algorithm for the example above 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 8, $(2/11)/0.4 = (3/11)/0.6 = 5/11$, which is the probability of not checking B given that it was not checked up to this point.

In this section we present Algorithm $\mathtt{A}^\star$, which given p, calculates the function L, and randomly chooses boxes so as to get L as its function. We describe the ideas behind it here, and the formal proof appears in appendix E.

Theorem 3. *For every non-coordinating algorithm A, $\mathbb{T}(\mathtt{A}^\star) \leq \mathbb{T}(A)$.*

Algorithm $\mathtt{A}^\star$

```
for t ← 1 to M do
    for y ← ac(t − 1) + 1 to ∞ do                          ▷ Calculate ac(t), α(t)
        if Σ_{x=1}^{y} 1 − q(x)/q(y) > t then
            ac(t) ← y − 1
            α(t) ← (ac(t) − t)/Σ_{x≤ac(t)} q(x)
    from unchecked boxes x ≤ ac(t)                         ▷ Choose one box
        if x ≤ ac(t − 1) then
            Check x w.p. 1 − α(t)/α(t − 1)
        else
            Check x w.p. 1 − α(t)q(x)
```

At step t, the first thing $\mathtt{A}^\star$ does is calculate the values of $L(x,t)$ for all x, so that it can recreate them with its random choices. For that it needs to calculate $\alpha(t)$, which by Observation 11 means solve the equation:

$$t = \sum_x 1 - L(x,t) = \sum_x 1 - \min(1, \alpha(t)q(x)) \tag{1}$$

The first step is to figure out which x's actually contribute something to this sum. 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. The point is, x is active when $\alpha(t) < 1/q(x)$, and so to see who is active, it needs to only check $\alpha(t) = 1/q(1), 1/q(2), \ldots$. Once $\mathtt{ac}(t)$ is found, solving (1) and finding $\alpha(t)$ is straightforward.

Now that $L(x,t)$ is calculated, $\mathtt{A}^\star$ randomly chooses a box to check according to it, using the fact that up to this point, 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 8 means it should be checked with probability $1 - \alpha(t)/\alpha(t-1)$. Fortunately, all these probabilities sum up to 1.

As an interesting side note, observe that at each step, all previously active yet unchecked boxes get the same probability of being checked. 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've already seen.

An 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.

How does algorithm $\mathtt{A}^\star$ look for example distributions, and how does it compare to $\mathtt{A}_{universal}$? In general it is quite difficult to analyse the exact running time of this algorithm, but sometimes it can be done, as we shall see.

3.4 Uniform Distribution

The first example that comes to mind is when the treasure is uniformly placed in one of the boxes $\{1, \ldots, M\}$. As $q(x)$ is equal for all boxes, an agent 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. This algorithm is the most natural choice in this case, and indeed, by Theorem 3 it is optimal. Analysis is simple:

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

Note that with coordination, the expected running time would be about $M/2k$, so we lose about a factor of 2 by non-coordination as opposed to 4 in the case of Algorithm $\mathtt{A}_{universal}$. 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 Pareto Distributions

$\mathtt{A}^\star$ is optimal, but it is a complex algorithm. For a large family of Pareto distributions we present a simplified algorithm that approximates the performance of $\mathtt{A}^\star$ well. Let $r_{b,M}$ be the Pareto distribution with parameter $b > 0$ on M boxes. Denote $b(x) = 1/x^b$, and then $r_{b,M}(x) = I/b(x)$, where $I = 1/\sum_{x=1}^{M} b(x)$ is the normalization factor. Note that the function $b(\cdot)$ will be important on its own right. We will especially be interested in the case[3] where $b < 1$, as when M grows, the fraction of the weight any specific box has goes to 0. For $b > 1$ that is not true, and so we are left with too little leeway for simplifying $\mathtt{A}^\star$.

In Algorithm $\mathtt{A}_{pareto}$, each agent, at its t-th step, chooses uniformly from one of the boxes it did not check yet in $\{1, \ldots, \min(M, \lfloor t/\sigma \rfloor)\}$, where $\sigma = b/(b+k-1)$. While $\mathtt{A}_{pareto}$ is not optimal, asymptotically it is. Practically all proofs of the section appear in Appendix F.

[3] In fact, our lower bound result also hold for $b = 1$, but our upper bound proof does not work for this case. However, we strongly believe the theorem to be true for $b = 1$ as well.

Theorem 4. *For $0 < b < 1$, $\lim_{M\to\infty} \frac{\mathbb{T}_{r_{M,b}}(\mathtt{A}_{pareto})}{\mathbb{T}_{r_{M,b}}(\mathtt{A}_{cord})} = k\sigma(2-\sigma) + \frac{k}{k+1}(2-b)(1-\sigma)^2$. Furthermore, no non-coordinating algorithm can achieve a better limit bound.*

In what follows, $o(1)$ means an expression that tends to 0 as M goes to infinity.

4.1 Lower Bound

The lower bound part of Theorem 4 is proved for all non-coordinating algorithms. For that, instead of the set of functions in $\mathcal{V}$, we consider a more general class of functions and so lower bound the original question. For a measurable set X denote:

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

For an $N \in \mathcal{F}(X)$, we say that N satisfies the *column requirements* if for all t: $C_N(t) = \int_X 1 - N(x,t)\,\mathrm{d}x \le t$. Such a function is called *valid*, and $\mathcal{V}(X)$ is the set of all valid functions. Given an integer $k \ge 2$ and some measurable function $p : X \to [0,\infty)$, define:

$$\mathbb{U}_{p,k}(N) = \int_0^\infty \int_X p(x)N(x,t)^k \,\mathrm{d}x\,\mathrm{d}t$$

This is a sort of equivalent of the $\mathbb{T}$ of algorithms, but is "unnormalized", as p is not necessarily a distribution. The following claim shows a connection between algorithms and functions:

Claim 12. *For every distribution p on $\{1, 2, \ldots, M\}$ and algorithm A on the M boxes, there is a function $N \in \mathcal{V}([1, M+1])$ such that $\mathbb{U}_{p',k}(N) \le \mathbb{T}_{p,k}(A)$, where $p' : [1, M+1] \to [0,\infty)$ is any non-increasing measurable function that agrees with p.*

It is proved quite directly by taking $N(x,t) = A(\lfloor x \rfloor, \lfloor t \rfloor)$. This shows that lower bounding the "running time" of functions in $\mathcal{V}([1, M+1])$ will lower bound the running time of algorithms on M boxes. Next, fix some $0 < b < 1$, and so the function $b(x) = 1/x^b$.

Observation 13. Let X be a finite interval of $\mathbb{R}^+$. Among all functions of $N \in \mathcal{V}(X)$ there is one that minimizes $\mathbb{U}_{b,k}(N)$. Denote it $\mathtt{OPT}_{b,X}$.

The proof of this observation uses the full power of Lemma 9 by finding the optimal function of x for each specific t, in a very similar way to the optimality proof of $\mathtt{A}^\star$. Next, we introduce the important tool of *zooming*, which is used a couple of times in what follows.

Definition 14. Given some $N \in \mathcal{F}(X)$ and $u, v > 0$, define the zooming of N by (u, v) as: $N_{\overrightarrow{u,v}}(x,t) = N(x/u, t/v)$, where $N_{\overrightarrow{u,v}}(x,t) \in \mathcal{F}(uX)$.

The intuitive meaning of it is that the algorithm is expanded to work on a domain of size u times the original one, and slowed down by a factor of v. What happens to the column requirement integrals and to the time?

Lemma 15. *For $N \in \mathcal{F}(X)$ and $u, v > 0$, $\mathbb{U}(N_{\overrightarrow{u,v}}) = u^{1-b}v\mathbb{U}(N)$, and for all t, $C_{N_{\overrightarrow{u,v}}}(t) = uC_N\left(\frac{t}{v}\right)$.*

This Lemma reduces our question to the running time of a specific set of optimal functions:

Claim 16. *For any Algorithm A that works on M boxes, denoting $r = r_{b,M}$, and $\epsilon = 1/(M+1)$:*

$$\frac{\mathbb{T}_r(A)}{\mathbb{T}_r(\mathtt{A}_{cord})} \ge (1 - o(1)) \cdot k(2-b) \cdot \mathbb{U}_b(\mathtt{OPT}_{[\epsilon,1]})$$

To use Claim 16 one should figure out who is $\texttt{OPT}_{[\epsilon,1]}$. This is possible using Lemma 9, but the equations that calculate $\alpha(t)$ are differential and it is not clear how to solve them. However, assuming M is large, we can trick our way out of this via a clever use of zooming, and so reduce the problem to calculating $\texttt{OPT}_{(0,1]}$ which is much simpler. Denote $\texttt{OPT} = \texttt{OPT}_{(0,1]}$. Then:

Claim 17. $\lim_{\epsilon\to 0}\left(\mathbb{U}(\texttt{OPT}_{[\epsilon,1]})/\mathbb{U}(\texttt{OPT})\right) = 1$

All that is left to do, is figure out $\texttt{OPT}$ and calculate its running time:

Claim 18. *Denote $\sigma = b/(b+k-1)$. Then, $\mathbb{U}(\texttt{OPT}) = \frac{\sigma(2-\sigma)}{2-b} + \frac{(1-\sigma)^2}{k+1}$.*

Finally, we can prove the lower bound and optimality part of Theorem 4. By Claim 16, Claim 17 and Claim 18, for every algorithm A:

$$\lim_{M\to\infty}\frac{\mathbb{T}(A)}{\mathbb{T}(\texttt{A}_{cord})} \geq k(2-b)\lim_{M\to\infty}\mathbb{U}(\texttt{OPT}_{[1/(M+1),1]}) = k(2-b)\mathbb{U}(\texttt{OPT}) = k\sigma(2-\sigma) + \frac{k(2-b)(1-\sigma)^2}{k+1}$$

4.2 Upper Bound

Below we describe the high level structure of the proof of the upper bound part of Theorem 4. The missing proofs of this section appear in Appendix G. A simple analysis of $\texttt{A}_{cord}$ and gives:

Claim 19.

$$\frac{\mathbb{T}(\texttt{A}_{pareto})}{\mathbb{T}(\texttt{A}_{cord})} \leq \frac{k(2-b)}{M^{2-b}}\sum_{t}\sum_{x=1}^{M}\frac{1}{x^b}\texttt{A}_{pareto}(x,t)^k$$

Since $\texttt{A}_{pareto}$ chooses uniformly from a set of unopened boxes at each stage, by Observation 8, when x is in this set then:

$$\texttt{A}_{pareto}(x,t) = \texttt{A}_{pareto}(x,t-1)\cdot\left(1-\frac{1}{|\text{interval chosen from}|-(t-1)}\right)$$

Applying generously and then using Lemma 6, one gets:

Claim 20.

$$\texttt{A}_{pareto}(x,t) \leq (1+o(1))\cdot\begin{cases}1 & t<\lceil\sigma x\rceil\\ \left(\frac{\lceil\sigma x\rceil}{t}\right)^{\frac{b}{k-1}} & \lceil\sigma x\rceil\leq t<\lceil\sigma M\rceil\\ \frac{1}{1-\sigma}\left(1-\frac{t}{M}\right)\left(\frac{\lceil\sigma x\rceil}{\sigma M}\right)^{\frac{b}{k-1}} & \lceil\sigma M\rceil\leq t<M\\ 0 & t\geq M\end{cases}$$

The point of this is that completely ignoring the rounding up operations, this is exactly $\texttt{OPT}(x/M, t/M)$, which appears in explicit form in (8) of Claim 18. Indeed, using very careful needlework math to get rid of these roundings, we show what would otherwise be a simple claim:

Claim 21.

$$\frac{1}{M^{2-b}}\sum_{t=0}^{M}\sum_{x=1}^{M}\frac{1}{x^b}\texttt{A}_{pareto}(x,t)^k \leq (1+o(1))\mathbb{U}(\texttt{OPT})$$

Plugging this into Claim 19 gives:

$$\frac{\mathbb{T}(\texttt{A}_{pareto})}{\mathbb{T}(\texttt{A}_{cord})} \leq (1+o(1))k(2-b)\mathbb{U}(\texttt{OPT})$$

Claim 18 gives the value of $\mathbb{U}(\texttt{OPT})$, and concludes the upper bound proof of Theorem 4 in exactly the same fashion as the end of the lower bound proof of this theorem.

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

[8] 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.

[9] Jurek Czyzowicz, Evangelos Kranakis, Danny Krizanc, Lata Narayanan, Jaroslav Opatrny, and Sunil M. Shende. Linear search with terrain-dependent speeds. *CoRR*, abs/1701.03047, 2017.

[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] 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.

[15] 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.

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

[17] Pierre Fraigniaud, Amos Korman, and Yoav Rodeh. Parallel exhaustive search without coordination. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 312–323, 2016.

[18] https://boinc.berkeley.edu/. BOINC.

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

[21] 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.

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

Appendix

A Coordinating Agents

Observation 22. $\mathtt{A}_{cord}$ is optimal among coordinating algorithms, and $\mathbb{T}(\mathtt{A}_{cord}) = \sum_x p(x)\lceil \frac{x}{k} \rceil$.

Proof. Coordinating k agents can be viewed as one algorithm that works in phases, where in each phase it can check k boxes. The aim is then to minimize the expected number of phases until the treasure is found. In this scenario, as there is no feedback from the algorithm's choices until the treasure is found, any randomized strategy can be seen as a distribution over deterministic algorithms. It follows then, that it is enough to consider deterministic algorithms.

W.l.o.g, as this cannot harm the running time, each box is checked once and in each phase there are exactly k boxes that are being checked. Now,

$$\mathbb{T}(A) = \sum_x p(x)T(x)$$

Where $T(x)$ is the phase where x is checked. By the assumption above, the sequence $T(1), T(2), \ldots$ contains exactly k copies of each positive integer (representing a phase number). Since $p(x)$ is a non-increasing sequence, then by the rearrangement inequality, $\mathbb{T}(A)$ is minimized when the $T(x)$ are arranged in a non-decreasing order, which is exactly algorithm $\mathtt{A}_{cord}$. Its running time is clear from definitions. □

B Observation 11

Observation 11. For every $t < M$, $C_L(t) = t$, and for $t \geq M$, $L(x,t) = 0$ everywhere.

Proof. For $t \geq M$, $\alpha(t) = 0$ satisfies the column requirement, and is as required. Next assume that $0 < t < M$. Clearly, in this case $\alpha(t) \neq 0$, as otherwise the column requirement is violated. Assume by contradiction that $C_L(t) \neq t$, and since L is valid this means that $C_L(t) < t$.

Note that since $p(x)$ goes to zero, $q(x)$ goes to infinity, and so there are only a finite number of x's where $\alpha(t)q(x) < 2$. As $\alpha(t) > 0$, we can reduce it slightly, and this will only affect the value of L at these x's. Making this change small enough, will maintain the inequality $C_L(t) < t$, and keep L valid. As this change can only decrease $L(x,t)$ at these points, $\mathbb{T}(L)$ does not increase. Contradicting the minimality of $\alpha(t)$. □

C Refined Analysis of the Algorithms of [17]

C.1 Efficiency of $\mathtt{A}_{universal}$

Claim 5. *For all x, $\mathbb{T}(\mathtt{A}_{universal}, x) \leq 10 + \frac{4k}{(k+1)^2}x$.*

Proof. Count the time in steps of size 2, so at each step $\mathtt{A}_{universal}$ chooses two new boxes. The algorithm might actually end mid-step, but this just means that this is an over approximation.

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:

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

Where the last step is by Claim 23 proved in Appendix C.3 below. Denoting $a = 2k/(k-1)$ the total running time for x is then at most (times 2):

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

As $((s+1)/t)^a$ is decreasing, 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:

$$\begin{aligned}
&s + 2 + \int_{s+1}^{\infty} \left(\frac{s+1}{t}\right)^a \mathrm{d}t = s + 2 + (s+1)\int_1^{\infty} t^{-a}\,\mathrm{d}t = s + 2 + \frac{s+1}{a-1} \\
&= 1 + (s+1)\left(1 + \frac{1}{\frac{2k}{k-1} - 1}\right) = 1 + \left(\left\lceil \frac{x}{k+1} \right\rceil + 1\right)\left(1 + \frac{k-1}{k+1}\right) \\
&\le 1 + \left(\frac{x}{k+1} + 2\right)\left(\frac{2k}{k+1}\right) \le 5 + \frac{2k}{(k+1)^2} s
\end{aligned}$$

Multipying by 2 gives the result. □

C.2 Efficiency of $\mathtt{A}_{memory}$

Claim 7. *For all x, $\mathbb{T}(\mathtt{A}_{memory}, x) \le 2 + 4\lceil \frac{x}{k} \rceil$.*

Proof. The proof proceeds in very similar to that of Theorem 5, yet is in fact a little simpler. Count the time in steps of size 2.

Box x starts to have some probability of being checked at time $s = \lceil x/k \rceil$, and for $t \ge s$ the probability of x not being checked by time t is:

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

Where the last step is by Lemma 6 below. The total 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, 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:

$$s + 1 + \int_s^{\infty} \left(\frac{s}{t}\right)^2 \mathrm{d}t = s + 1 + s\int_1^{\infty} \frac{1}{t^2}\,\mathrm{d}t = 2s + 1$$

Multipying by 2 gives the result. □

C.3 Gamma Function Property

Claim 23. *For integers $b \ge a \ge 1$, and $k \ge 2$,*

$$\prod_{i=a}^{b} \frac{i}{i + \frac{2}{k-1}} \le \left(\frac{a+1}{b}\right)^{\frac{2}{k-1}}$$

Proof. Let us start with the case $k = 2$. In this case, $2/(k-1) = 2$. If $a = b$ then this is clearly true, as left side is at most 1, and right side at least 1. Otherwise, the product is telescopic and we get:

$$\prod_{i=a}^{b} \frac{i}{i+2} = \frac{a(a+1)}{(b+1)(b+2)}$$

It is indeed at most $(a+1)^2/b^2$, as its numerator is smaller, and its denominator larger.

Regarding $k \geq 3$. In this case, $2/(k-1) \leq 1$, and so we can use Lemma 6 below, to get that the product in the claim is at most:

$$\left(\frac{a}{b}\right)^{\frac{2}{k-1}} \leq \left(\frac{a+1}{b}\right)^{\frac{2}{k-1}}$$

□

Lemma 6. *For integers $b \geq a \geq 1$, and $0 < \phi \leq 1$, $\prod_{i=a}^{b} \frac{i}{i+\phi} \leq \left(\frac{a}{b}\right)^{\phi}$.*

Proof. By induction on a (somehow on b it doesn't work..). If $b = a$, then we should show $a/(a+\phi) \leq 1$, which is true. We therefore assume that the result holds for $a+1$ and prove it for a:

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

We want to show:

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

Take $b = \frac{1}{a} < 1$ and $x = \frac{1}{\phi} \geq 1$ the above is equivalent to:

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

If we show that the left side is increasing with x when $x \geq 1$ then we are done. We take the derivative (using an internet site):

$$\left(1 + \frac{b}{x}\right)^{x} \cdot \left(\ln\left(1 + \frac{b}{x}\right) - \frac{b}{x\left(1 + \frac{b}{x}\right)}\right)$$

This is positive if

$$\left(1 + \frac{b}{x}\right) \ln\left(1 + \frac{b}{x}\right) > \frac{b}{x}$$

Take $y = \frac{b}{x} \leq 1$. We want to show that

$$(1+y)\ln(1+y) > y$$

We use the equality

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

So

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

as desired. □

D Proof of Main Lemma

Recall that $V(T)$ the set of measurable functions $f : X \to [0,1]$ such that $\int 1 - f \, \mathrm{d}\mu \leq T$. Also, for a measurable function $c : X \to [0, \infty)$, and $\alpha \geq 0$ the function $f_{c,\alpha} : X \to [0,1]$ is:

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

In what follows we will drop the subscript c when it is clear from context.

Lemma 9. *For a given c and T as above, if there is some $h \in V(T)$ such that $\int ch^k \, \mathrm{d}\mu < \infty$, then there exists $\alpha \geq 0$, such that $f_{c,\alpha} \in V(T)$, and for every $g \in V(T)$, $\int cf_{c,\alpha}^k \, \mathrm{d}\mu \leq \int cg^k \, \mathrm{d}\mu$. Furthermore, this α is minimal among those satisfying $f_{c,\alpha} \in V(T)$.*

Proof. A few of points to begin with:

1. All of the functions below are measurable, either by definition or by straightforward proof. Also, as all of them are positive, they will all have a defined Lebesgue integral (though its value may be ∞.

2. Starting from the end, assuming that the existence of α is proved, we claim that $\alpha = \min\{\beta \geq 0 \mid f_\beta \in V(T)\}$. Existence proves that this set is not empty. By monotonicity, if $\beta < \beta'$ then $\int cf_\beta^k \, \mathrm{d}\mu \leq \int cf_{\beta'}^k \, \mathrm{d}\mu$, so all that remains to show is that this minimum exists. If not, then there is some sequence $\{\beta_n\}_{n=1}^\infty$ that approaches an 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.

3. Denote $S = \{x \in X \mid c(x) > 0\}$, the support of c. If $T \geq \mu(S)$, then take $\alpha = 0$. We get $f_0(x) = 0$ on S and 1 elsewhere, so $\int 1 - f_0 \, \mathrm{d}\mu = \mu(S) \leq T$, and so $f_0 \in V(T)$. Also, $\int cf_0^k \, \mathrm{d}\mu = 0$ and is therefore optimal, so we are done. We will therefore always assume that $T < \mu(S)$. Specifically, $\mu(S) > 0$ and $T < \mu(X)$.

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

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

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

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

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

The proof proceeds as usual in these cases, by a gradual increase of the generality of the function c that we handle.

Indicator functions. First assume $c = 1_A$ is the indicator function of some set $A \subseteq X$. By Item 4, $\mu(A) < \infty$, and by Item 3, we can assume $T < \mu(A)$. For any $g \in V(T)$:

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

Where we used Jensen's inequality for the case where the total measure is not necessarily 1. Take $\alpha = (\mu(A) - T)/\mu(A)$. We get $0 < \alpha < 1$, and $f_\alpha(x) = \alpha$ for every $x \in A$ and 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,

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

And so f_α is optimal. Note that it is a constant function on A.

Simple functions. In this case, $c = \sum_{i=1}^n c_i 1_{X_i}$, where all $c_i > 0$, and the X_i are pair-wise disjoint. Also, by Item 4, 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. Denote $T_i = \int_{X_i} 1 - g \,\mathrm{d}\mu$. Restricted to X_i, according to the case of indicator functions, there is some constant $g_i \geq 0$, such that $\int_{X_i} 1 - g_i \,\mathrm{d}\mu \leq T_i$, and $\int_{X_i} g_i^k \,\mathrm{d}\mu \leq \int_{X_i} g^k \,\mathrm{d}\mu$.

We therefore define $g' = 1_Y + \sum_{i=1}^n g_i 1_{X_i}$, where $Y = X \setminus \cup_{i=1}^n X_i$. According to the above,

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

So $g' \in V(T)$. Also,

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

So g' is a better candidate than g, and we can therefore assume that g is constant on each of the X_i's, and can be written as $g = \sum_{i=1}^n g_i 1_{X_i}$.

Our question can now be viewed as follows. Given $c_1, \ldots, c_n$ and $\mu_1, \ldots \mu_n > 0$, find the $g_1, \ldots, g_n \in [0, 1]$ among those satisfying $\sum_{i=1}^n \mu_i(1 - g_i) \leq T$, that minimize $\sum_{i=1}^n c_i g_i^k$. As the solution space is compact and the function to minimize is continues, 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. According to Lemma 24 below, these two values must satisfy:

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

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

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 $f_n = f_{c_n,\alpha_n}$ gives minimal $\int c_n f_n^k \,\mathrm{d}\mu$ among all functions of $V(T)$.

If this sequence of α_n 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 the α_n, and denote its limit by α. Now, 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.

Examine the sequence of functions $1 - f_n$. They all satisfy $\int 1 - f_n \,\mathrm{d}\mu \le T$, and so by Fatou's lemma:

$$\int 1 - f \,\mathrm{d}\mu \le \liminf_{n\to\infty} \int 1 - f_n \,\mathrm{d}\mu \le T$$

And so $f \in V(T)$. For all x, the function cf^k is the pointwise limit of $c_n f_n^k$. As f_n is optimal for c_n,

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

So all these integrals are jointly bounded and so their lim inf exists. Therefore, by Fatou's lemma:

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

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

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

Take large enough n, and use the fact that f_n is optimal for c_n,

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

and we get a contradiction.

The only thing left to show is that $f = f_{c,\alpha}$ for some α. As we've seen there are two cases, and we have to deal with the case where $f = 1$.

By (3) there is some $\epsilon > 0$ such that the set $A = \{x \in X \mid c(x) > \epsilon\}$ satisfies $\mu(A) > 0$, and by (4), $\mu(A) < \infty$. If $T < \mu(A)$, then take the function $g(x) = 1 - T/\mu(A)$ on this set and 1 elsewhere. Clearly $g \in V(T)$. Also,

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

As f is optimal, it cannot be the function 1 and must be of the required form. If $T > \mu(A)$, proceed in the same way, except $g(x) = 0$ on A and 1 elsewhere. □

D.1 Lemma for 2

Lemma 24. *Let $k \ge 2$, $c_1, c_2, \mu_1, \mu_2 > 0$, and $M \le \mu_1 + \mu_2$. 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:*

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

Proof. Let $c = c_2/c_1$, and $\mu = \mu_2/\mu_1$. Setting $N = M/\mu_1$, we can write the lemma equivalently as follows. Assuming $N \leq 1 + \mu$, knowing that $g_1 + \mu g_2 = N$, the $g_1, g_2 \in [0, 1]$ minimizing $g_1^k + c\mu g_2^k$ satisfy $g_1 = \min\left(1, c^{1/(k-1)} g_2\right)$.

Denoting $g_2 = (N - g_1)/\mu$, we want to minimize:

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

We take the derivative w.r.t. g_1:

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

This is zero exactly when:

$$g_1 = c^{\frac{1}{k-1}} \cdot \frac{N - g_1}{\mu} \left(= c^{\frac{1}{k-1}} g_2\right) \tag{3}$$

We get:

$$g_1 = \frac{c^{\frac{1}{k-1}}}{\mu + c^{\frac{1}{k-1}}} N \tag{4}$$

We take the second derivative (the first was (2)),

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

If we look at g_1's in the range $[0, N]$, this is always strictly positive, meaning our function is U shaped there. Also, by (4) the minimum is somewhere in $[0, N]$. Recall that $g_1 \in [0, 1]$. If the bottom of the U is in $[0, 1]$ then as we've seen in (3) we get the lemma. Otherwise it must be somewhere in $(1, N]$, and so our minimum would be at $g_1 = 1$. Note that it is unique. □

E Optimality proof of $\mathtt{A}^\star$

Theorem 3. *For every non-coordinating algorithm A, $\mathbb{T}(\mathtt{A}^\star) \leq \mathbb{T}(A)$.*

Proof. First, $\mathtt{A}^\star$ calculates $\alpha(t)$ and $\mathtt{ac}(t)$. Note that $y \leq \mathtt{ac}(t)$ iff $\alpha(t) < 1/q(y)$, 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 Observation 11:

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

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 $t-1$, and were not checked yet at time t is $\mathtt{ac}(t-1) - (t-1)$. So, summing all the probabilities of the different boxes:

$$(\mathtt{ac}(t-1) - t + 1)\left(1 - \frac{\alpha(t)}{\alpha(t-1)}\right) + \sum_{\mathtt{ac}(t-1) < x \leq \mathtt{ac}(t)} 1 - \alpha(t)q(x) \tag{5}$$

By Observation 11:

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

Plugging this is (5):

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

By Observation 11 the first sum is t and the second is $t-1$, and so the sum of probabilities is indeed 1.

The last bit is to show that indeed $\mathtt{A}^\star = L$. This is proved by induction on t. For $t = 0$, $L(x, 1) = \mathtt{A}^\star(x, 1)$ for all x. Assume equality for $t-1$ and we prove it for t. For $x \le \mathtt{ac}(t-1)$, $\mathtt{A}^\star(x, t-1) = L(x, t-1) = \alpha(t-1)q(x)$. Using Observation 8:

$$\mathtt{A}^\star(x, t) = \mathtt{A}^\star(x, t-1) \cdot \frac{\alpha(t)}{\alpha(t-1)} = \alpha(t-1)q(x) \cdot \frac{\alpha(t)}{\alpha(t-1)} = L(x, t)$$

For $\mathtt{ac}(t-1) < x \le \mathtt{ac}(t)$, it is straightforward. □

F Lower Bounding Pareto Distributions

F.1 Claim 12

Claim 12. *For every distribution p on $\{1, 2, \ldots, M\}$ and algorithm A on the M boxes, there is a function $N \in \mathcal{V}([1, M+1])$ such that $\mathbb{U}_{p',k}(N) \le \mathbb{T}_{p,k}(A)$, where $p' : [1, M+1] \to [0, \infty)$ is any non-increasing measurable function that agrees with p.*

Proof. Define $N(x, t) = A(\lfloor x \rfloor, \lfloor t \rfloor)$. For any t:

$$C_N(t) = \int_1^{M+1} 1 - A(\lfloor x \rfloor, \lfloor t \rfloor)\,\mathrm{d}x = \sum_{x=1}^{M} 1 - A(x, \lfloor t \rfloor) = C_A(\lfloor t \rfloor) \le \lfloor t \rfloor \le t$$

So N satisfies the column requirements. Next,

$$\mathbb{U}_{p',k}(N) = \int_0^\infty \int_1^{M+1} p'(x) A(\lfloor x \rfloor, \lfloor t \rfloor)^k \,\mathrm{d}x\,\mathrm{d}t \le \int_0^\infty \int_1^{M+1} p(\lfloor x \rfloor) A(\lfloor x \rfloor, \lfloor t \rfloor)^k \,\mathrm{d}x\,\mathrm{d}t$$
$$= \sum_{x=1}^{M} \sum_{t=0}^{\infty} p(x) A(x, t) = \mathbb{T}_{p,k}(A)$$

□

F.2 Observation 13

Observation 13. Let X be a finite interval of $\mathbb{R}^+$. Among all functions of $N \in \mathcal{V}(X)$ there is one that minimizes $\mathbb{U}_{b,k}(N)$. Denote it $\mathtt{OPT}_{b,X}$.

Proof. Fix t. Setting $c(x) = b(x)$ and $T = t$, Lemma 9 gives a function $f_t(x)$ minimizing $\int_X b(x) f_t(x)^k \,\mathrm{d}x$, under the condition $\int_X 1 - f_t(x)\,\mathrm{d}x \le t$. To fulfil the condition of the lemma, take $h(x) = 0$ for $x \le t$, and 1 elsewhere. Then $\int_X 1 - h(x)\,\mathrm{d}x \le t$, and as X is a finite interval. Also,

$$\int_X \frac{1}{x^b} h(x)^k \,\mathrm{d}x = \int_{X \setminus [0,t]} \frac{1}{x^b} \,\mathrm{d}x < \infty$$

Putting all these t's together by setting $\mathtt{OPT}_{b,X}(x,t) = f_t(x)$, we get that $\mathtt{OPT}_{b,X} \in \mathcal{V}(X)$. Also:

$$\mathbb{U}(\mathtt{OPT}_{b,X}) = \int_0^\infty \int_X \frac{1}{x^b} f_t(x)^k \,\mathrm{d}x$$

And as the f_t minimize the inner integral for any function in $\mathcal{V}(X)$, we get that if this integral exists it is minimal. To show it exists, we note that the function $F(t) = \int_X \frac{1}{x^b} f_t(x)^k \,\mathrm{d}x$ is non-increasing in t, and so is measurable. This is because, if $t < t'$, $f_t \in V(t')$, and so by minimality of $f_{t'}$, $F(f_{t'}) \leq F(f_t)$. This means that $\mathbb{U}(\mathtt{OPT}_{b,X})$ is defined, although it might be ∞. □

F.3 Zooming Lemma

Lemma 15. *For $N \in \mathcal{F}(X)$ and $u, v > 0$, $\mathbb{U}(N_{\overrightarrow{u,v}}) = u^{1-b} v \mathbb{U}(N)$, and for all t, $C_{N_{\overrightarrow{u,v}}}(t) = uC_N\left(\frac{t}{v}\right)$.*

Proof. First:

$$\mathbb{U}(N_{\overrightarrow{u,v}}) = \int_0^\infty \int_{uX} \frac{1}{x^b} N(x/u, t/v)^k \,\mathrm{d}x \,\mathrm{d}t = uv \int_0^\infty \int_X \frac{1}{(xu)^b} N(x,t)^k \,\mathrm{d}x \,\mathrm{d}t = u^{1-b} v \mathbb{U}(N)$$

The column integrals:

$$C_{N_{\overrightarrow{u,v}}}(t) = \int_{uX} 1 - N(x/u, t/v) \,\mathrm{d}x = u \int_X 1 - N(x, t/v) \,\mathrm{d}x = uC_N\left(\frac{t}{v}\right)$$

□

F.4 Claim 16

Claim 16. *For any Algorithm A that works on M boxes, denoting $r = r_{b,M}$, and $\epsilon = 1/(M+1)$:*

$$\frac{\mathbb{T}_r(A)}{\mathbb{T}_r(\mathtt{A}_{cord})} \geq (1 - o(1)) \cdot k(2-b) \cdot \mathbb{U}_b(\mathtt{OPT}_{[\epsilon,1]})$$

Proof. Recall $r(x) = I/x^b$, where $I = 1/\sum_{i=1}^M 1/x^b$. We take r' to be the extension of this on all of $[1, M+1]$. Then, by Claim 12 there is some $N \in \mathcal{V}([1, M+1])$ such that:

$$\mathbb{T}_r(A) \geq \mathbb{U}_{r'}(N) = I \cdot \mathbb{U}_b(N)$$

Where the last step is trivial when examining the definition of $\mathbb{U}$. Consider $N' = N_{\overrightarrow{\epsilon,\epsilon}}$. By Lemma 15 and the fact that N satisfies the column requirements,

$$C_{N'}(t) = \epsilon C_N\left(\frac{t}{\epsilon}\right) \leq \epsilon \cdot \frac{t}{\epsilon} = t$$

So $N' \in \mathcal{V}([\epsilon, 1])$. Also, by the same lemma,

$$\mathbb{U}_b(N') = \frac{1}{(M+1)^{2-b}} \mathbb{U}_b(N)$$

Together with the fact that $\mathbb{U}_b(N') \geq \mathbb{U}_b(\mathtt{OPT}_{[\epsilon,1]})$, we obtain:

$$\mathbb{T}_r(A) \geq (M+1)^{2-b} \cdot I \cdot \mathbb{U}_b(\mathtt{OPT}_{[\epsilon,1]})$$

The running time of $\mathtt{A}_{cord}$ is:

$$\mathbb{T}_r(\mathtt{A}_{cord}) = \sum_{x=1}^{M} \frac{I}{x^b}\left\lceil \frac{x}{k} \right\rceil \leq \sum_{x=1}^{M} \frac{I}{x^b}\left(\frac{x}{k}+1\right) = \frac{I}{k}\sum_{x=1}^{M} x^{1-b} + 1$$
$$\leq 1 + \frac{I}{k}\int_1^{M+1} x^{1-b}\,\mathrm{d}x = 1 + \frac{I}{k}\frac{(M+1)^{2-b}}{2-b}$$

Where we used the fact that x^{1-b} is monotonically non-decreasing. Together:

$$\frac{\mathbb{T}_r(A)}{\mathbb{T}_r(\mathtt{A}_{cord})} \geq \frac{(M+1)^{2-b}\cdot I\cdot \mathbb{U}_b(\mathtt{OPT}_{[\epsilon,1]})}{1+\frac{I}{k}\frac{(M+1)^{2-b}}{2-b}} = \frac{1}{\frac{k(2-b)}{I(M+1)^{2-b}}+1}\cdot k(2-b)\mathbb{U}_b(\mathtt{OPT}_{[\epsilon,1]})$$

As $b > 0$, then $I^{-1} = \sum_{i=1}^{M} 1/x^b = o(M)$, and so the first factor tends to 1 as M tends to infinity. □

F.5 Getting Rid of ϵ

Claim 17. $\lim_{\epsilon\to 0}\left(\mathbb{U}(\mathtt{OPT}_{[\epsilon,1]})/\mathbb{U}(\mathtt{OPT})\right) = 1$

Proof. For the sake of this proof, denote $E = \mathtt{OPT}_{[\epsilon,1]}$. To show that the limit is at most 1, define $\mathtt{OPT}'$ to be $\mathtt{OPT}$ restricted to $[\epsilon, 1]$. It is easy to see that $\mathtt{OPT}' \in \mathcal{V}([\epsilon,1])$, and so $\mathbb{U}(\mathtt{OPT}') \geq \mathbb{U}(E)$. Also, it is clear that $\mathbb{U}(\mathtt{OPT}') \leq \mathbb{U}(\mathtt{OPT})$, which concludes this side.

To show that the limit is indeed 1, we construct a new function E' that will span the whole range of x's from 0 to 1, with little change to $\mathbb{U}(E)$. This will be done by slowing E down, and using what we saved in the column integrals to visit the x's between 0 and ϵ using our optimal solution, running it fast enough so it does not incur a big difference in $\mathbb{U}(E)$.

Fix some $a < 1$ to be determined later. Define:

$$E'(x,t) = \begin{cases} \mathtt{OPT}_{\overrightarrow{\epsilon,\epsilon/(1-a)}} & x \leq \epsilon \\ E_{\overrightarrow{1,1/a}} & x > \epsilon \end{cases}$$

Since the zoomed version of $\mathtt{OPT}$ here is defined on the x's in $(0,\epsilon]$ and the zoomed E is on those in $[\epsilon, 1]$, we get that for all t:

$$C_{E'}(t) = C_{\mathtt{OPT}_{\overrightarrow{\epsilon,\epsilon/(1-a)}}}(t) + C_{E_{\overrightarrow{1,1/a}}}(t) = \epsilon C_{\mathtt{OPT}}\left(\frac{1-a}{\epsilon}t\right) + C_E(at) \leq \epsilon\frac{1-a}{\epsilon}t + at = t$$

where we used Lemma 15 and the fact that both $\mathtt{OPT}$ and E satisfy the column requirements. So $E' \in \mathcal{V}((0,1])$ and by the optimality of $\mathtt{OPT}$, $\mathbb{U}(E') \geq \mathbb{U}(\mathtt{OPT})$. Again, by Lemma 15:

$$\mathbb{U}(\mathtt{OPT}) \leq \mathbb{U}(E') = \mathbb{U}(\mathtt{OPT}_{\overrightarrow{\epsilon,\epsilon/(1-a)}}) + \mathbb{U}(E_{\overrightarrow{1,1/a}}) = \frac{\epsilon^{2-b}}{1-a}\mathbb{U}(\mathtt{OPT}) + \frac{1}{a}\mathbb{U}(E)$$

And therefore, denoting $\epsilon' = \epsilon^{2-b}$,

$$\mathbb{U}(E) \geq a\left(1 - \frac{\epsilon'}{1-a}\right)\mathbb{U}(\mathtt{OPT})$$

Taking $a = 1-\sqrt{\epsilon'}$, the factor becomes: $\left(1-\sqrt{\epsilon'}\right)^2$, which goes to 1 as ϵ goes to 0. □

F.6 Analysing OPT

Claim 18. *Denote $\sigma = b/(b+k-1)$. Then,* $\mathbb{U}(\texttt{OPT}) = \frac{\sigma(2-\sigma)}{2-b} + \frac{(1-\sigma)^2}{k+1}$.

Proof. We proceed as in Observation 13, and use Lemma 9 to figure out the exact structure of OPT. For each fixed t it gives the function $f_t(x)$ minimizing $\int_0^1 \frac{1}{x^b} f_t(x)^k \, \mathrm{d}x$, from all those satisfying the column requirement $\int_0^1 1 - f_t(x) \leq t$. As shown in the observation's proof, $\texttt{OPT}(x,t) = f_t(x)$.

The first step is to understand what is $f_t(x)$. For $t \geq 1$, the optimal f_t is obviously $f_t(x) = 0$, since this satisfies the column requirement and has an integral of 0. For $t < 1$, we have $\int_0^1 1 - f_t(x) \, \mathrm{d}x \leq t$, and that $f_t(x) = \min(1, \alpha_t x^{b/(k-1)})$. So, given t, it is possible to deduce what its corresponding α is (we drop the subscript). For each t, denote by γ (a function of t) the smallest x where $f_t(x) = 1$, and in case this does not happen, set $\gamma = 1$. So:

$$\gamma = \min\left(1, \frac{1}{\alpha^{\frac{k-1}{b}}}\right) \tag{6}$$

for every $t < 1$, to minimize our target function, we would like f_t to be the smallest possible and so the column requirement will actually be an equality:

$$t = \int_0^1 1 - \min\left(1, \alpha x^{\frac{b}{k-1}}\right) \mathrm{d}x = \gamma - \int_0^\gamma \alpha x^{\frac{b}{k-1}} \, \mathrm{d}x$$

We have two cases:

1. for all t where $\gamma < 1$ this equation is:

$$\gamma - t = \int_0^\gamma \alpha x^{\frac{b}{k-1}} \, \mathrm{d}x \tag{7}$$

   From (6), and using the assumption that $\gamma < 1$, we get $\alpha = 1/\gamma^{b/(k-1)}$. Plugging this in:

$$\gamma - t = \int_0^\gamma \left(\frac{x}{\gamma}\right)^{\frac{b}{k-1}} \mathrm{d}x = \gamma \int_0^1 x^{\frac{b}{k-1}} \, \mathrm{d}x = \gamma \frac{1}{\frac{b}{k-1} + 1} = \frac{k-1}{b+k-1}\gamma$$

   Recall $\sigma = b/(b+k-1)$, and so $\gamma = t/\sigma$. This means, that for all $t < \sigma$, $\gamma < 1$ and then $\alpha = (\sigma/t)^{b/(k-1)}$. For all other t, $\gamma = 1$.

2. for all t where $\gamma = 1$:

$$1 - t = \int_0^1 \alpha x^{\frac{b}{k-1}} \, \mathrm{d}x = \alpha \frac{1}{\frac{b}{k-1} + 1} = (1-\sigma)\alpha$$

   and so:

$$\alpha = \frac{1-t}{1-\sigma}$$

Putting all this together:

$$\texttt{OPT}(x,t) = \begin{cases} 1 & t \leq \sigma x \\ \left(\frac{\sigma x}{t}\right)^{\frac{b}{k-1}} & \sigma x < t \leq \sigma \\ \frac{1-t}{1-\sigma} x^{\frac{b}{k-1}} & \sigma < t \leq 1 \\ 0 & t > 1 \end{cases} \tag{8}$$

Next, we wish to calculate $\mathbb{U}(\texttt{OPT})$.

$$\begin{aligned}\mathbb{U}(\texttt{OPT}) &= \int_0^1 \int_0^1 \frac{1}{x^b} \texttt{OPT}(x,t)^k \,\mathrm{d}x \,\mathrm{d}t \\ &= \int_0^\sigma \left(\int_0^\gamma \frac{1}{x^b} (\alpha x^{\frac{b}{k-1}})^k \,\mathrm{d}x \right) \mathrm{d}t + \int_0^\sigma \left(\int_\gamma^1 \frac{1}{x^b} 1^k \,\mathrm{d}x \right) \mathrm{d}t + \int_\sigma^1 \left(\int_0^1 \frac{1}{x^b} (\alpha x^{\frac{b}{k-1}})^k \,\mathrm{d}x \right) \mathrm{d}t\end{aligned}$$

Focus on each term separately:

1. Here $\gamma = t/\sigma < 1$. Hence, the inner integral is:

$$\int_0^\gamma \frac{1}{x^b} (\alpha x^{\frac{b}{k-1}})^k \,\mathrm{d}x = \alpha^{k-1} \int_0^\gamma \alpha x^{\frac{b}{k-1}} \,\mathrm{d}x = \alpha^{k-1}(\gamma - t) = \frac{1}{\gamma^b}(\gamma - t)$$

   Where the second equality is due to (7). Then, as $\gamma = t/\sigma$, we get:

$$\left(\frac{\sigma}{t}\right)^b \left(\frac{t}{\sigma} - t\right) = \sigma^b t^{1-b} \frac{1-\sigma}{\sigma}$$

   The whole integral if $b < 1$:

$$(1-\sigma) \int_0^\sigma \left(\frac{t}{\sigma}\right)^{1-b} \mathrm{d}t = \sigma(1-\sigma) \int_0^1 t^{1-b} \,\mathrm{d}t = \frac{\sigma(1-\sigma)}{2-b}$$

   If $b = 1$ then it is $\sigma(1-\sigma)$ which is the same.

2. Here, still, $\gamma < 1$. Two cases:

   (a) If $b = 1$, the inner integral is:

$$\int_\gamma^1 \frac{1}{x} \,\mathrm{d}x = \log(1) - \log(\gamma) = -\log(\gamma)$$

   Plugging in $\gamma = t/\sigma$, and calculating the whole integral:

$$\int_0^\sigma -\log(t/\sigma) \,\mathrm{d}t = -\sigma \int_0^1 \log(t) \,\mathrm{d}t = \sigma$$

   Last bit is because indefinite integral of $\log(x)$ is $x\log(x) - x$.

   (b) If $b < 1$:

$$\int_\gamma^1 \frac{1}{x^b} \,\mathrm{d}x = \frac{1}{1-b}\left(1 - \gamma^{1-b}\right)$$

   Plugging in $\gamma = t/\sigma$, and calculating the whole integral:

$$\frac{1}{1-b} \int_0^\sigma 1 - \left(\frac{t}{\sigma}\right)^{1-b} \mathrm{d}t = \frac{\sigma}{1-b}\left(1 - \int_0^1 t^{1-b} \,\mathrm{d}t\right) = \frac{\sigma}{1-b}\left(1 - \frac{1}{2-b}\right) = \frac{\sigma}{2-b}$$

   So $\sigma/(2-b)$ works for both cases.

3. Here $\gamma = 1$.

$$\int_0^1 \frac{1}{x^b}(\alpha x^{\frac{b}{k-1}})^k \,\mathrm{d}x = \alpha^{k-1}\int_0^1 \alpha x^{\frac{b}{k-1}} \,\mathrm{d}x = \alpha^{k-1}(1-t)$$

Plugging in $\alpha = (1-t)/(1-\sigma)$ and calculating the whole integral:

$$\frac{1}{(1-\sigma)^{k-1}}\int_\sigma^1 (1-t)^k \,\mathrm{d}t = \frac{1}{(1-\sigma)^{k-1}}\int_0^{1-\sigma} t^k \,\mathrm{d}t$$
$$= \frac{1}{(k+1)(1-\sigma)^{k-1}}(1-\sigma)^{k+1} \,\mathrm{d}t = \frac{(1-\sigma)^2}{k+1}$$

In total:

$$\frac{\sigma(1-\sigma)}{2-b} + \frac{\sigma}{2-b} + \frac{(1-\sigma)^2}{k+1} = \frac{\sigma(2-\sigma)}{2-b} + \frac{(1-\sigma)^2}{k+1}$$

□

G Upper Bounding Pareto Distributions

G.1 Bounding the ratio $\mathbb{T}(\mathtt{A}_{pareto})/\mathbb{T}(\mathtt{A}_{cord})$

Claim 19.

$$\frac{\mathbb{T}(\mathtt{A}_{pareto})}{\mathbb{T}(\mathtt{A}_{cord})} \le \frac{k(2-b)}{M^{2-b}}\sum_t\sum_{x=1}^M \frac{1}{x^b}\mathtt{A}_{pareto}(x,t)^k$$

Proof. Setting $I = (\sum_{x=1}^M 1/x^b)^{-1}$, the running time of $\mathtt{A}_{cord}$ is (noting that x^{1-b} is non-decreasing):

$$\mathbb{T}(\mathtt{A}_{cord}) = \sum_{x=1}^M \frac{I}{x^b}\left\lceil\frac{x}{k}\right\rceil \ge \frac{I}{k}\sum_{x=1}^M x^{1-b} \ge \frac{I}{k}\int_0^M x^{1-b}\,\mathrm{d}x = \frac{I}{k}\frac{M^{2-b}}{2-b}$$

So:

$$\frac{\mathbb{T}(\mathtt{A}_{pareto})}{\mathbb{T}(\mathtt{A}_{cord})} = \frac{\sum_t\sum_{x=1}^M \frac{I}{x^b}\mathtt{A}_{pareto}(x,t)^k}{\mathbb{T}(\mathtt{A}_{cord})} \le \frac{k(2-b)}{M^{2-b}}\sum_t\sum_{x=1}^M \frac{1}{x^b}\mathtt{A}_{pareto}(x,t)^k$$

□

G.2 Figuring out $\mathtt{A}_{pareto}$'s Function

Claim 20.

$$\mathtt{A}_{pareto}(x,t) \le (1+o(1)) \cdot \begin{cases} 1 & t < \lceil\sigma x\rceil \\ \left(\frac{\lceil\sigma x\rceil}{t}\right)^{\frac{b}{k-1}} & \lceil\sigma x\rceil \le t < \lceil\sigma M\rceil \\ \frac{1}{1-\sigma}\left(1-\frac{t}{M}\right)\left(\frac{\lceil\sigma x\rceil}{\sigma M}\right)^{\frac{b}{k-1}} & \lceil\sigma M\rceil \le t < M \\ 0 & t \ge M \end{cases}$$

Proof. As mentioned, since $\mathtt{A}_{pareto}$ chooses uniformly from a set of unopened boxes at each stage, by Observation 8, when x is in this set then:

$$\mathtt{A}_{pareto}(x,t) = \mathtt{A}_{pareto}(x,t-1)\cdot\left(1-\frac{1}{|\text{interval chosen from}| - (t-1)}\right)$$

Also,

1. Fix x. When $x > \lfloor t/\sigma \rfloor$ it has no probability of being checked, and as x is an integer, this means $x > t/\sigma$, and so $t < \sigma x$. It therefore starts being checked when $t = \lceil \sigma x \rceil$.

2. The checking is over all unchosen boxes when $M \leq \lfloor t/\sigma \rfloor \leq t/\sigma$. This starts when $t = \lceil M\sigma \rceil$.

Combining all this together gives:

$$\mathtt{A}_{pareto}(x,t) \leq \begin{cases} 1 & t < \lceil \sigma x \rceil \\ \prod_{i=\lceil \sigma x \rceil}^{t} \left(1 - \frac{1}{i/\sigma - i + 1}\right) & \lceil \sigma x \rceil \leq t < \lceil \sigma M \rceil \\ \prod_{i=\lceil \sigma x \rceil}^{\lceil \sigma M \rceil - 1} \left(1 - \frac{1}{i/\sigma - i + 1}\right) \prod_{i=\lceil \sigma M \rceil}^{t} \left(1 - \frac{1}{M - i + 1}\right) & \lceil \sigma M \rceil \leq t < M \\ 0 & t \geq M \end{cases}$$

Where i/σ replaces $\lfloor i/\sigma \rfloor$ in the probabilities, as it only increases the result. Now,

$$\prod_{i=\lceil \sigma x \rceil}^{t} \left(1 - \frac{1}{i/\sigma - i + 1}\right) = \prod_{i=\lceil \sigma x \rceil}^{t} \frac{(1/\sigma - 1)i}{(1/\sigma - 1)i + 1} = \prod_{i=\lceil \sigma x \rceil}^{t} \frac{i}{i + \frac{\sigma}{1-\sigma}} \leq \left(\frac{\lceil \sigma x \rceil}{t}\right)^{\frac{b}{k-1}}$$

Where Lemma 6 is used for last step. Similarly:

$$\prod_{i=\lceil \sigma M \rceil}^{t} \left(1 - \frac{1}{M - i + 1}\right) = \prod_{i=\lceil \sigma M \rceil}^{t} \frac{M - i}{M - i + 1} = \frac{M - \lceil \sigma M \rceil}{M - \lceil \sigma M \rceil + 1} \cdots \frac{M - t}{M - t + 1}$$
$$= \frac{M - t}{M - \lceil \sigma M \rceil + 1} \leq \frac{M - t}{M - \sigma M} = \frac{1}{1 - \sigma}\left(1 - \frac{t}{M}\right)$$

So:

$$\mathtt{A}_{pareto}(x,t) \leq \begin{cases} 1 & t < \lceil \sigma x \rceil \\ \left(\frac{\lceil \sigma x \rceil}{t}\right)^{\frac{b}{k-1}} & \lceil \sigma x \rceil \leq t < \lceil \sigma M \rceil \\ \frac{1}{1-\sigma}\left(1 - \frac{t}{M}\right)\left(\frac{\lceil \sigma x \rceil}{\lceil \sigma M \rceil - 1}\right)^{\frac{b}{k-1}} & \lceil \sigma M \rceil \leq t < M \\ 0 & t \geq M \end{cases}$$

Finally, multiply the third case by $((\lceil \sigma M \rceil - 1)/\sigma M)^{b/(k-1)}$. This will decrease the final result by at most this factor, which tends to 1 as M goes to infinity, giving the result. □

G.3 Relating $\mathtt{A}_{pareto}$ and $\mathtt{OPT}$

Claim 21.

$$\frac{1}{M^{2-b}} \sum_{t=0}^{M} \sum_{x=1}^{M} \frac{1}{x^b} \mathtt{A}_{pareto}(x,t)^k \leq (1 + o(1))\mathbb{U}(\mathtt{OPT})$$

Proof. Our aim is to show:

$$\sum_{t=0}^{M} \sum_{x=1}^{M} \frac{1}{x^b} \mathtt{A}_{pareto}(x,t)^k \leq M^{2-b} \int_0^1 \int_0^1 \frac{1}{x^b} \mathtt{OPT}(x,t)^k \, \mathrm{d}x \, \mathrm{d}t \tag{9}$$

while being quite loose in this comparison, as we can allow additive terms of $o(M^{2-b})$ and still get the result. Since $\mathtt{OPT}$ is non-increasing in t, then so is $\int_0^1 \frac{1}{x^b} \mathtt{OPT}(x,t)^k \, \mathrm{d}x$. The right side is then at least:

$$M^{1-b} \sum_{t=1}^{M} \int_0^1 \frac{1}{x^b} \mathtt{OPT}(x, t/M)^k \, \mathrm{d}x$$

The case $t = 0$ contributes at most an additive $\sum_{x=1}^{M} 1/x^b = o(M)$ to the left side of 9, and so is insignificant. This is in fact true for any particular t. So to prove (9), we will show that for all but a small constant number of t's:

$$\sum_{x=1}^{M} \frac{1}{x^b} \mathtt{A}_{pareto}(x,t)^k \leq M^{1-b} \int_0^1 \frac{1}{x^b} \mathtt{OPT}(x, t/M)^k \, \mathrm{d}x \tag{10}$$

Where here, additive terms of order $o(M^{1-b})$ are considered insignificant.

Putting side by side $\mathtt{A}_{pareto}$'s upper bound as presented in Claim 20 (ignoring the $1+o(1)$ factor which does not bother us), and the explicit form of $\mathtt{OPT}$ of (8), shows their resemblance:

$$\begin{cases} 1 & t < \lceil \sigma x \rceil \\ \left(\frac{\lceil \sigma x \rceil}{t}\right)^{\frac{b}{k-1}} & \lceil \sigma x \rceil \leq t < \lceil \sigma M \rceil \\ \frac{1}{1-\sigma}\left(1 - \frac{t}{M}\right)\left(\frac{\lceil \sigma x \rceil}{\sigma M}\right)^{\frac{b}{k-1}} & \lceil \sigma M \rceil \leq t < M \\ 0 & t \geq M \end{cases} \qquad \begin{cases} 1 & t \leq \sigma x \\ \left(\frac{\sigma x}{t}\right)^{\frac{b}{k-1}} & \sigma x < t \leq \sigma \\ \frac{1-t}{1-\sigma} x^{\frac{b}{k-1}} & \sigma < t \leq 1 \\ 0 & t > 1 \end{cases}$$

Fix some t, and set $f(x) = \mathtt{OPT}(x/\sigma, t/M)^k$. Clearly $\mathtt{OPT}(x, t/M)^k = f(\sigma x)$, but also:

$$\mathtt{A}_{pareto}(x,t)^k \leq \mathtt{OPT}\left(\frac{\lceil \sigma x \rceil}{\sigma M}, \frac{t}{M}\right)^k = f\left(\frac{\lceil \sigma x \rceil}{M}\right)$$

for all but possibly $t = \lceil \sigma M \rceil \pm 1$. So to prove (10), it will be enough to prove that for any function f where $f(\cdot) \in [0,1]$:

$$\sum_{x=1}^{M} \frac{1}{x^b} f\left(\frac{\lceil \sigma x \rceil}{M}\right) \leq M^{1-b} \int_0^1 \frac{1}{x^b} f(\sigma x) \, \mathrm{d}x \tag{11}$$

Assuming the integral above is defined. Again, additive terms of order $o(M^{1-b})$ are considered insignificant. The next step is to approximate the integral by a very specific Riemann sum. This gives a result that is correct up to a multiplicative factor that tends to 1, which is fine for our purpose. The n-th interval is $I_n = (i_{n-1}/M, i_n/M]$, where $i_n = \lceil n/\sigma \rceil$. I_n is sampled at $n/\sigma M$, which is clearly an inner point of I_n.

For example, if $\sigma = 0.3$ then $I_1 = (1,4], I_2 = (4,7], I_3 = (7,10], I_4 = (10,14]$, and the respective sample points are $3\frac{1}{3}, 6\frac{2}{3}, 10$ and $13\frac{1}{3}$. Of course, all of this should be divided by M. Note that the size of the intervals is about $1/M\sigma$ and so tends to 0, as required.

The right hand side of (11) is approximated by:

$$M^{1-b} \cdot \frac{1}{M} \sum_{n=1}^{\lceil \sigma M \rceil} (i_n - i_{n-1}) \frac{1}{(n/\sigma M)^b} f(n/M) = \sum_{n=1}^{\lceil \sigma M \rceil} (i_n - i_{n-1}) \frac{f(n/M)}{(n/\sigma)^b}$$

This can be written as:

$$\sum_{x=1}^{M} \frac{f(n_x/M)}{(n_x/\sigma)^b} \tag{12}$$

Where n_x is defined to satisfy $x \in (i_{n_x - 1}, i_{n_x}]$, so that indeed, each term in the original sum appears exactly $i_n - i_{n-1}$ times in the new sum (of course without the factor of $(i_n - i_{n-1})$). The condition on n_x is actually $\left\lceil \frac{n_x - 1}{\sigma} \right\rceil < x \leq \left\lceil \frac{n_x}{\sigma} \right\rceil$. The following is proved in Appendix G.3.1:

Observation 25. If x and n are natural numbers, then

$$\left\lceil \frac{n-1}{\sigma} \right\rceil < x \leq \left\lceil \frac{n}{\sigma} \right\rceil \quad \iff \quad n = \lceil \sigma x \rceil$$

Using this observation, (12) is:

$$\sum_{x=1}^{M} \frac{1}{(\lceil \sigma x \rceil / \sigma)^b} f\left(\frac{\lceil \sigma x \rceil}{M}\right) \geq \sum_{x=1}^{M} \frac{1}{(x + \frac{1}{\sigma})^b} f\left(\frac{\lceil \sigma x \rceil}{M}\right)$$

Since $b \leq 1$, x^b is sub-linear, and so:

$$\frac{1}{x^b} - \frac{1}{(x+\frac{1}{\sigma})^b} = \frac{(x+\frac{1}{\sigma})^b - x^b}{x^b(x+\frac{1}{\sigma})^b} \leq \frac{\frac{1}{\sigma}}{x^b(x+\frac{1}{\sigma})^b} \leq \frac{1}{\sigma x^{1+b}}$$

Noting that $f(\cdot) \leq 1$, the above is at least:

$$\sum_{x=1}^{M} \frac{1}{x^b} f\left(\frac{\lceil \sigma x \rceil}{M}\right) - \frac{1}{\sigma} \sum_{x=1}^{M} \frac{1}{x^{1+b}}$$

If $b < 1$, then the second term is bounded above by some constant independent of M, and so in this case is $o(M^{1-b})$, proving (11). □

G.3.1 Proof of Rounding Observation

Observation 25. If x and n are natural numbers, then

$$\left\lceil \frac{n-1}{\sigma} \right\rceil < x \leq \left\lceil \frac{n}{\sigma} \right\rceil \quad \iff \quad n = \lceil \sigma x \rceil$$

Proof. The case where $n = \sigma x$ is straightforward. Otherwise,

1. $$n = \lceil \sigma x \rceil \quad \implies \quad n > \sigma x \quad \implies \quad x < \frac{n}{\sigma} \quad \implies \quad x < \left\lceil \frac{n}{\sigma} \right\rceil$$

   Also,

   $$n = \lceil \sigma x \rceil \quad \implies \quad n - 1 < \sigma x \quad \implies \quad \frac{n-1}{\sigma} < x \quad \implies \quad \left\lceil \frac{n-1}{\sigma} \right\rceil \leq x$$

   But equality would mean that $n - 1 = \sigma x$, which cannot be true.

2. $$x < \left\lceil \frac{n}{\sigma} \right\rceil \quad \implies \quad x < \frac{n}{\sigma} \quad \implies \quad \sigma x < n$$

   And,

   $$x > \left\lceil \frac{n-1}{\sigma} \right\rceil \quad \implies \quad x > \frac{n-1}{\sigma} \quad \implies \quad \sigma x > n - 1$$

   So $\lceil \sigma x \rceil = n$.

□