Weighted online search

Spyros Angelopoulos, Konstantinos Panagiotou

▶ **To cite this version:**

Spyros Angelopoulos, Konstantinos Panagiotou. Weighted online search. Journal of Computer and System Sciences, 2023, 138, pp.103457. 10.1016/j.jcss.2023.05.002 . hal-04248699

HAL Id: hal-04248699
https://hal.science/hal-04248699

Submitted on 18 Oct 2023

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.

Weighted Online Search

Spyros Angelopoulos[1] and Konstantinos Panagiotou[2]

[1]CNRS and LIP6-Sorbonne University, Paris, France, spyros.angelopoulos@lip6.fr

[2]LMU Munich, Munich, Germany, kpanagio@lmath.lmu.de

Abstract

We study the general setting of *weighted search* in which a number of weighted targets are hidden in a star-like environment, and a mobile searcher must locate a subset of targets with aggregate weight at least a given value W. The cost of the strategy is the distance traversed by the searcher, and its performance is measured by the worst-case ratio of the cost incurred by the searcher over the cost of an on optimal, offline strategy. This is the first study of a setting that generalizes several problems in search theory such as searching for a single target and searching for unit-weighted targets. We present and analyze a near-optimal strategy using an approach based on parameterized analysis. This problem formulates settings of resource allocation among parallel tasks under uncertainty; specifically, we demonstrate further applications in the design of interruptible systems based on *adaptive* scheduling of contract algorithms.

Keywords: Search problems, competitive analysis, ray search, contract scheduling.

1 Introduction

We introduce and study the following general search problem. We are given a star-like environment that consists of m concurrent *rays* of infinite length, with a common origin O. For each ray $i \in \{0, \ldots, m-1\}$, there is a *target* of weight $w_i \geq 0$ that is hidden at some distance $d_i \geq 0$ from O. Note that the setting allows cases such as $d_i = \infty$ (i.e., there is no target hidden on ray i) and $w_i = 0$ (i.e., the target has no weight). A mobile searcher is initially at O, and its objective is to locate a subset of targets whose aggregate weight is at least a specified value W. The searcher

knows the number of rays m, however, it has no further knowledge about the instance, namely the weights of the targets and their distances from O.

A *search strategy* Σ for this problem is specified by determining, at every point in time that the searcher is at the origin, a ray i and a depth ℓ_i, such that the searcher will next traverse ray i to depth ℓ_i and then will return back to O. If during this search a target was found, and as long as the weight accrued is less than W, the searcher will immediately return to O and will never again traverse ray i. In general, we assume that the search strategies have *memory*; the pairs (i, ℓ_i) may depend, among other things, on targets already discovered by Σ and the already explored depths.

As common for searching in unbounded environments, we evaluate the performance of a search strategy Σ by means of the well-established *competitive ratio*, which can be traced to early work in (Beck and Newman, 1970) in the context of searching on the line. Informally, the competitive ratio compares the performance of a search strategy that is oblivious of the instance (that is, the exact position of the targets and their weights) to an omniscient searcher that has full information on the instance. To formalize this concept, given a number of rays $m \geq 2$, let us denote by $\mathcal{I}_m$ the set of all instances

$$\mathcal{I}_m = \Big\{ \big(W, (d_i, w_i)_{0 \leq i \leq m-1\}}\big) : W \geq 0, w_i \geq 0, d_i \geq 1, \text{ and } \sum_{0 \leq i \leq m-1} w_i \geq W \Big\}.$$

We make two standard assumptions, namely that all targets are at least a unit distance from O – otherwise, no strategy can have a bounded competitive ratio – and that there is a feasible solution to the instance. Given $I \in \mathcal{I}_m$ the *cost of* Σ *on* I, denoted by $c(\Sigma, I)$ is defined as the total distance traversed by the searcher until the first time it discovered targets of aggregate weight at least W. We also denote by $\mathrm{opt}(I)$ the *optimal cost of* I, namely the cost of an ideal strategy that has complete knowledge of the instance I (i.e., the positions of the targets and their weights, as well as W). A strategy Σ is called ρ-*competitive* for some $\rho \geq 1$, if $c(\Sigma, I) \leq \rho \cdot \mathrm{opt}(I)$, for all $I \in \mathcal{I}_m$. The competitive ratio of Σ is then defined as

$$\rho(\Sigma) = \sup_{I \in \mathcal{I}_m} \frac{c(\Sigma, I)}{\mathrm{opt}(I)}, \tag{1}$$

and reflects the overhead of Σ due to lack of information. A strategy of minimum competitive ratio is called *optimal*. Figure 1 illustrates an example of an instance.

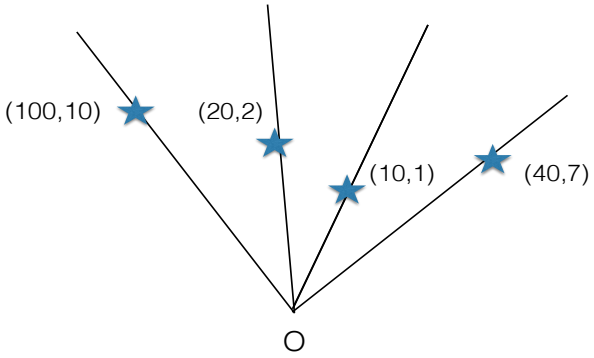

Figure 1: Illustration of the weighted star search problem, for an instance I with $m = 4$ rays. The target at ray i is represented by the pair (d_i, w_i), where d_i denotes its distance from O and w_i it's weight. Assume an indexing of rays as $0 \dots 3$, from left to right, and a value of W equal to 10. Then an optimal strategy will either search the ray 0 at a cost of 100 or the rays 1,2,3 (in this order), which also yields the optimal cost of $2 \cdot 20 + 2 \cdot 10 + 40 = 100$.

Our setting is motivated by many factors. First, it generalizes several well-studied problems in Theoretical Computer Science and Operations Research, most notably the (unweighted) *ray search* or *star search* problem in (Gal, 1974; Baeza-Yates et al., 1993; Jaillet and Stafford, 1993), in which a single target is present on some ray. Note that star search is a generalization of another well-known problem, namely the *linear search* problem, originally studied in (Bellman, 1963; Beck, 1964). In addition, the weighted setting subsumes the multi-target search problem of (Angelopoulos et al., 2014), in which each target has unit weight, and the objective is to locate t targets, for some $t \in \mathbb{N}^+$ that is known to the strategy. Star search problems have a long history of study; see Section 1.3 for a discussion of related work.

More importantly, weighted search provides a useful abstraction of the setting in which one has to allocate resources to parallel tasks without knowing, in advance, the degree to which each task will prove itself fruitful. This is a fundamental aspect of many decision-making processes. For instance, a researcher may want to determine how to allocate time among m different projects, without knowing ahead of time how useful the outcome of each project will be towards a combined publication of the potential results. Here, the projects can be modeled as weighted targets, with the distance to a target representing the time needed to the successful completion of the corresponding project.

Connections between ray searching and resource allocation among parallel tasks have been

shown even for unweighted search, in the context of studies in TCS, AI and OR. Notable examples include: the design of *interruptible* algorithms, i.e., algorithms that return acceptable solutions even if interrupted during their execution (see (Bernstein et al., 2003; Angelopoulos, 2015)); the synthesis of *hybrid* algorithms, in which the best algorithm must be chosen among a suite of different algorithms, without knowing in advance which one is the best on a given instance; and database query optimization (in particular, pipeline filter ordering as studied in (Condon et al., 2009), which has an equivalent search-related formulation over a star graph as discussed in (Angelopoulos et al., 2019)). Thus, solutions to the weighted search problem can provide a framework for solving weighted variants of the above settings. For instance, one may define hybrid algorithms based on the model of (Kao et al., 1998), in which each individual algorithm execution is assigned a score related to the quality of the returned solution. In this sense, our setting is a generalization akin to the study of weighted objectives in scheduling theory.

1.1 Contribution

We present and analyze a strategy called AdaptiveSearch (or AdSch for short) for weighted ray search that attains a (near-)optimal competitive ratio. Define the function ϕ by

$$\phi(x) = 1 + 2\,(1+x)\left(1+\frac{1}{x}\right)^x,\ x > 0.$$

Note that $\phi(x) = \Theta(x)$. As we will see, ϕ plays a central role in the analysis and reveals connections to previous studies of the simpler variants of search problems. Our first theorem provides a tight, worst-case analysis for weighted search.

Theorem 1. *The competitive ratio of* AdSch *is* $\phi(m-1)$*, and this is the optimal competitive ratio for weighted search.*

Some remarks are in place. Let us consider a special case, namely the classic problem of searching on m rays. Here, there is solely one target that hides in some ray unknown to the searcher. Using our notation, this corresponds to the case $d_i = \infty$ for all rays $0 \leq i \leq m-1$ except for exactly one ray i^*, and $W = w_{i^*}$. The competitive ratio of this problem was determined in (Gal, 1974) to be equal to $\phi(m-1)$. It is then immediate that the competitive ratio of weigthed search is at least $\phi(m-1)$; the main contribution of Theorem 1 is to establish that this is also an upper

bound. In other words, our first result says that in the worst case, weighted search is *at most* as difficult as searching for one target in m rays.

Even though Theorem 1 is tight for some instances, it provides a rather pessimistic guarantee. To see this, let us consider another special case, namely *multi-target* search (Angelopoulos et al., 2014). In this variant we must locate exactly $1 \leq t \leq m-1$ targets that are placed on rays again unknown to the searcher; this corresponds to $d_i = \infty$, for all except of t rays $0 \leq i_1^* < \cdots < i_t^* \leq m-1$ with weights $w_{i_1^*} = \cdots = w_{i_t^*} = W/t$. In (Angelopoulos et al., 2014) it was shown that the competitive ratio of this problem is $\phi(m-t)$, which can be rephrased as "searching for t targets on m rays is as difficult as searching for one target in $m-t+1$ rays". Note that the competitive ratio is a decreasing function of t.

The results in (Angelopoulos et al., 2014) open up the possibility that more refined guarantees than $\phi(m-1)$ may be attainable for weighted search; in particular, we can expect that if many targets have to be located, in the sense that *any* solution needs to collect many targets, then the competitive ratio should decrease. In order to make this precise, we follow a *parameterized* approach, in which we study the performance of a strategy not only in terms of m, but also in terms of some natural parameters that reflect the "hardness" of the instance. More specifically, given $I = \{(W, (d_i, w_i)_{0 \leq i \leq m-1})\}$, define $W_I = W$ and $w_{\max,I} = \max_{0 \leq i \leq m-1} w_i$. Then any strategy, including the offline optimal one that knows I, must locate at least $\lceil W_I / w_{\max,I} \rceil$ targets. The question then is: what is the competitiveness of weighted search, in terms of both m and the bound on the number of targets that must be found? Our next result gives a tight bound, and generalizes the known results on multi-target search.

Theorem 2. *Let $m \in \mathbf{N}$. Let $I \in \mathcal{I}_m$ be such that $w_{\max,I} > 0$. If $\lceil W_I / w_{\max,I} \rceil < m$, then*

$$c(\textsc{AdSch}, I) \leq \phi\left(m - \lceil W_I / w_{\max,I} \rceil\right) \cdot \mathrm{opt}(I).$$

Moreover, for every strategy Σ, there exists $I \in \mathcal{I}_m$ with $\frac{W_I}{w_{\max,I}} < m$ such that

$$c(\Sigma, I) \geq \phi\left(m - \lceil W_I / w_{\max,I} \rceil\right) \cdot \mathrm{opt}(I).$$

In other words, the competitive ratio of AdSch is optimal for instances with a fixed value of

$\lceil W_I/w_{\max,I} \rceil$. However, we can obtain an even stronger performance guarantee. As we show in the next theorem, we can, rather surprisingly, relate the competitiveness of weighted search to the *maximum* number of targets that an ideal solution of optimal cost must locate; in contrast, in Theorem 2, the competitive ratio is expressed in terms of the *minimum* number of targets in any ideal solution (more precisely: a lower bound on this number). For a general instance $I \in \mathcal{I}_m$, let s_I denote the maximum value of this parameter, see Section 2 for a formal definition. To illustrate this concept, consider the instance of Figure 1. There are two optimal solutions: one locating the target $(100, 10)$, and one that locates all other targets. Thus, in this instance, $s_I = 3$.

Theorem 3. *Let $I \in \mathcal{I}_m$. If $s_I < m$, then*

$$c(\textsc{AdSch}, I) \leq \phi(m - s_I) \cdot \mathrm{opt}(I).$$

Moreover, for every strategy Σ and every $s \in \{1, \ldots, m-1\}$ there is $I \in \mathcal{I}_m$ with $s_I = s$ such that

$$c(\Sigma, I) \geq \phi(m - s) \cdot \mathrm{opt}(I).$$

The extreme cases $W_I/w_{\max,I} = m$ and $s_I = m$ in the statements of Theorems 2 and 3 respectively, describe the outlier cases in which the optimal strategy must locate all m targets. In this extreme case, we can prove a very small gap between the upper and the lower bounds, as expressed in the following theorem.

Theorem 4. *Let $I \in \mathcal{I}_m$. If $s_I = m$, then*

$$c(\textsc{AdSch}, I) \leq (3 + 2e)\mathrm{opt}(I).$$

Moreover, for every strategy Σ and $\epsilon > 0$ there is $I \in \mathcal{I}_m$ with $s_I = m$ such that

$$c(\Sigma, I) \geq (2 - \epsilon)\mathrm{opt}(I).$$

Last, we demonstrate that our results and techniques have further applications in resource allo-

cation under uncertainty. In Section 5 we study an application related to the the design of interruptible algorithms for solving parallel instances of problems, based on contract scheduling (Russell and Zilberstein, 1991). Previous studies of this problem assumed a *static* setting, in which the schedule is determined ahead of time. In our setting, instead, we consider the *adaptive* setting in which some of the instances may be deemed solved if the system has made sufficient progress towards them. We show how to obtain an interruptible system of optimal performance, as measured by the *acceleration ratio* (Russell and Zilberstein, 1991).

1.2 Techniques and analysis

In this work we rely on a strategy that searches the rays in a *cyclic*, i.e., round-robin manner; this is motivated by the fact that, for the special case of the m-ray search problem, (Gal, 2010) showed that any optimal search strategy must be cyclic. There are two main challenges that one needs to address. The first challenge pertains to the design of the strategy itself, and more precisely in determining the appropriate sequence of search lengths. Unlike most previous work in which for an optimal strategy it suffices to increase the search lengths by the same factor at each step, in our setting we show later that such simple rules are very inefficient. We thus introduce an *adaptive* strategy in which the search lengths increase depending on the total number of targets found by the end of each step. The second challenge lies in analyzing this strategy, since the setting is substantially more complex than the unweighted one. To this end, we relate the competitiveness of the weighted search problem to that of a related problem which we term *subset search*. The objective in the latter problem is, informally, to locate a certain subset of the targets, without advance knowledge of the specific subset. For this problem, which is of independent interest, we show strategies that achieve the same competitiveness as strategies that search for *any* subset of targets that has the same cardinality as the subset that is sought. This, perhaps surprising, result establishes a strong upper bound on the competitiveness of the problem and is the main technical challenge.

To arrive at this result, we first prove essentially tight bounds on the optimal cost, by identifying appropriate parameters related to the search, and by relaying the cost of our strategy to precisely those parameters. We then apply an inductive argument on the number of targets discovered by the strategy in the last m rays that have been explored by the searcher. Some technical difficulties arise

from the fact that, unlike the single-target variant, there is no simple expression that describes the optimal cost, and that the cost of the searcher involves several parameters in a trade-off relation.

Note that the vast majority of previous works on ray searching applies to deterministic algorithms (i.e., pure strategies), and so does ours. In addition, as with previous works, the objective is not merely to give an asymptotic expression on the competitive ratio (which is a trivial task), but rather find the *exact* competitive ratio.

1.3 Further related work

The m-ray search problem, also known as *star search*, is a generalization of the *linear search* problem (Beck, 1964; Beck and Newman, 1970) and has been studied in many variants and settings. It is known that any optimal search strategy must be defined as a geometric sequence of search lengths (Gal, 1972; Gal and Chazan, 1976). We refer the reader to Chapters 8 and 9 in the book (Alpern and Gal, 2003) for results related to search games on a star. In Computer Science, star search has been studied in parallel (López-Ortiz and Schuierer, 2004; Angelopoulos et al., 2016), fault-tolerant (Czyzowicz et al., 2019; Kupavskii and Welzl, 2018), randomized (Kao et al., 1998; Schuierer, 2003; Kao et al., 1996) and stochastic (Jaillet and Stafford, 1993; Kao and Littman, 1997; Bonato et al., 2020) settings. Multi-target searching beyond the competitive ratio was studied in (Kirkpatrick, 2009; McGregor et al., 2009; Angelopoulos et al., 2014). Other variants include: searching with turn cost (Demaine et al., 2006; Angelopoulos et al., 2017); searching with an upper bound on the target distance (Hipke et al., 1999; Bose et al., 2015) or a general hint (Angelopoulos, 2021); and expanding search (Angelopoulos et al., 2019; Angelopoulos and Lidbetter, 2020). These are only some representative works.

2 Notation & m-ray search

We begin with some useful definitions and facts concerning the problem we study. Given an instance $I \in \mathcal{I}_m$ of WeightedSearch (WS), we denote by $T_I = \{0, \ldots, m-1\}$ the set of the m targets of I. Given $S \subseteq T_I$, we define the *total weight of* S, denoted by w_S as $\sum_{i \in S} w_i$, and the *optimal*

search cost of S, denoted by d_S as

$$d_S = 2\sum_{i\in S} d_i - \max_{i\in S} d_i.$$

The latter is the minimum cost required by the mobile searcher initially located at the origin to visit all targets in S assuming full information of I (note that the ray with the most distant target among targets in S is visited last, and that the searcher does not need to return to the origin after having located all targets). With this notation the optimal cost of the instance I is

$$\mathrm{opt}(I) = \min_{S\subseteq T_I}\{d_S : w_S \geq W\}.$$

Moreover, we denote by s_I the largest number of targets in an optimal solution, that is,

$$s_I = \max\big\{|S| : S \subseteq T_I, w_S \geq W, \text{and } d_S = \mathrm{opt}(I)\big\}.$$

The following quantities will appear several times in the forthcoming analysis. Define

$$b_x = 1 + \frac{1}{x} \quad \text{and} \quad b_{m,t} = b_{m-t} = 1 + \frac{1}{m-t}. \tag{2}$$

Note that for the function ϕ defined in the introduction we obtain the useful relation

$$\phi(x) = 1 + 2b_x^{x+1}/(b_x - 1).$$

Let us now turn our attention to a special case of weighted search considered in the introduction, namely the classical m-ray search problem in which there is exactly one target to be found. For this problem, it was shown in (Gal, 1974) that there exists an optimal cyclic strategy with geometrically increasing lengths. More specifically, in the i-th step, the strategy searches ray $i \pmod m$ up to length $b_{m,1}^i$. As already mentioned, the (optimal) competitive ratio equals

$$1 + 2\frac{b_{m,1}^m}{b_{m,1} - 1} = \phi(m-1).$$

Concerning the problem in which there are m unweighted targets, one per ray, and the searcher seeks to locate any subset of exactly t targets, it was shown in (Angelopoulos et al., 2014) that there is an optimal cyclic strategy, which in the i-th step searches the corresponding ray up to length $b^i_{m-t+1,1}$. The optimal competitive ratio is equal to

$$1 + 2\frac{b^{m-t+1}_{m-t+1,1}}{b_{m-t+1,1} - 1} = \phi(m-t),$$

namely the same as the competitive ratio of searching for a single target in a star consisting of $m-t+1$ rays. Note that in the unweighted setting, the searcher benefits from knowing the number t of targets that are sought. This allows (Angelopoulos et al., 2014) to give an non-adaptive optimal strategy, i.e., one in which in the i-th step the searcher goes up to distance b^i from O, for some appropriate b. As we will argue in the next section, this type of non-adaptive strategy cannot be efficient for weighted search, because the "right" number of targets that needs to be located is a parameter that is unknown to the searcher.

3 Strategies and the overall approach

In this section we present the strategy for weighted star search, and the main approach to the analysis, namely the relation between the competitiveness of WS and SubsetSearch (SS). We formally define the latter problem as follows. The instance to the problem consists of m unweighted targets (one per ray) with the target at ray i being at distance d_i from the origin of the star, as well as a subset $S \subseteq T_I$ of the targets. The distances, as well as the subset S are not known to the searcher. The search terminates when all targets in S have been discovered; we can assume the presence of an oracle that announces this event to the searcher and thus prompts the termination. The cost of the search is defined as the total cost incurred at termination, whereas the cost of the optimal solution to the instance is the cost for locating all targets in S assuming full information of the instance, i.e., equal to d_S. The following lemma establishes a useful association between the two problems.

Lemma 5. *Suppose there is a strategy Σ_s for SS such that for any instance $I_s = (S, d_0, \dots d_{m-1})$ we have that $c(\Sigma_s, I_s) \leq \rho(|S|, m) \cdot d_S$, where $\rho(|S|, m)$ is a function of $|S|, m$. Then there is a strategy Σ_w for* WS *such that for any instance I_w of this problem, $c(\Sigma_w, I_w) \leq \rho(s_I, m) \cdot opt(I_w)$.*

Proof. Let $I_w = \{W, (d_0, w_0), \ldots, (d_{m-1}, w_{m-1})\}$ denote an instance of WS. Consider the instance I_s for SS in which m targets are at distances $d_0, \ldots d_{m-1}$, and S is defined to be any subset of targets such that $d_S = \text{opt}(I_w)$ and $|S| = s_I$. From the definition of s_I, such a set exists. Define Σ_w for WS as follows: Σ_w executes Σ_s until an aggregate target weight of at least W has been located. Since $w_S \geq W$, it follows that $c(\Sigma_w, I_w) \leq c(\Sigma_s, I_S)$. Furthermore, since Σ_S has competitive ratio $\rho(s_I, m)$, we have that $c(\Sigma_s, I_S) \leq \rho(s_I, m) d_S$. Thus, $c(\Sigma_w, I_w) \leq \rho(s_I, m)\text{opt}(I_w)$. □

Lemma 5 demonstrates that the competitiveness of SS is directly related to that of WS. We thus focus on SS; in particular, we will analyze a strategy which we call AdaptiveSubset (AdSub for brevity), and which is described by the statement of Algorithm 1. The strategy explores rays in cyclic order (line 13), and keeps track of the found targets, denoted by f (line 11). On each iteration, the strategy will search a ray to a length equal to $b_{m,f}$ times the length of the last ray exploration that did not reveal a target (line 8), until a new target is discovered. Once a target is found on a ray, the ray is not considered again; namely, the remaining rays are relabeled so as to maintain a cyclic order (line 12).

Algorithm 1: Strategy AdSub for SS

```
1  Input: m rays labeled {0, ..., m − 1}, subset oracle O
2  f ← 1, r ← 0, D ← 1
3  repeat
4      repeat
5          explore the rth ray up to distance D · b_{m,f} or until a target is found
6          if no target was found then
7              r ← r + 1 (mod m − f + 1)
8              D ← D · b_{m,f}
9          end
10     until a target was found
11     f ← f + 1
12     remove the rth ray and relabel the rays canonically from 0 ... m − f
13     r ← r (mod m − f + 1)
14 until all targets, according to O, were found
```

Before we proceed with the analysis, it is instructive to point out that the need for an adaptive strategy that modifies the search lengths as a function of the number of targets that have been found seems to be, in a sense, unavoidable. To see this, consider a cyclic strategy that uses geometrically increasing lengths with a fixed base that may depend only on m. Consider also an instance in

which $|S|-1$ targets from S are very close to the origin, then the competitive ratio of this strategy is (more or less) at least the competitive ratio of a strategy that searches one target in $m-|S|+1$ rays; crucially, this ratio is only achieved if the base equals $b_{m-|S|+1,1}$; see Section 2. Since $|S|$ is not known in advance, the wrong choice of a base has a detrimental effect on performance.

The above example demonstrates the need for an adaptive strategy that modifies the search lengths as a function of the number of targets that have been found. However, it is not immediately clear how to adapt the search lengths. To see this, let us consider an obvious candidate, which also turns out to be inefficient. Namely, a cyclic, geometric strategy that changes the base of search lengths once a target is found; more precisely, a strategy that on the i-th step searches the corresponding ray up to depth $b^i_{m,t}$, where $t-1$ represents the number of targets found at the beginning of the step. This strategy has unbounded competitive ratio. To see this, suppose that $|S|=1$, and that in iteration i, the strategy finds on ray r a target that is not in S. Suppose also that the unique target in S lies on ray $(r-1) \bmod m$ and at distance $b^{i-1}_{m,1}+\epsilon$, for some $\epsilon > 0$. The strategy will discover this target after having spent cost at least $2b^{i+m-2}_{m,2}$ (namely, the cost for searching ray $(r-2) \bmod m$) on iteration $m+i-2$). It follows that the competitive ratio is at least $(b_{m,2}/b_{m,1})^i$, which, since i is arbitrary, is unbounded.

By combining the insights from both previous examples, we see that not only we have to adapt the search lengths, but we also have to ensure a smooth transition when such a modification takes place. This is precisely accomplished in AdSub; when a target is found, say in iteration i, the search depths in the following rounds do not jump 'abruptly' from b^{i+j} to $\tilde{b}^{i+j}$, where $\tilde{b} \neq b$, but instead to $b^i \cdot \tilde{b}^j$, $j \geq 0$.

The following lemma, which bounds the competitive ratio of AdSub, is the main technical result of this work. Its proof is given in Section 4.

Lemma 6. *Let $m \geq 2$. Let I be an instance of SS in which we seek a non-empty set $S \subseteq \{0,\ldots,m-1\}$. Then*

$$c(\textsc{AdSub}, I) \leq \phi(m-|S|) \cdot d_S, \quad \textit{if} \ \ |S| \leq m-1, \tag{3}$$

and $c(\textsc{AdSub}, I) \leq (3+2e) \cdot d_S$, if $|S|=m$.

Assuming Lemma 6 we show how to obtain Theorem 1. This establishes a tight, albeit worst-case bound on the competitive ratio.

Proof of Theorem 1. From Lemmas 5 and 6 and since ϕ is increasing, for all $I \in \mathcal{I}_m$,

$$c(\textsc{AdSch}, I) \leq \max\{3 + 2e, \phi(m-1)\} \cdot \text{opt}(I).$$

Moreover, $\phi(1) = 9 > 3 + 2e$, hence the upper bound follows. This bound is tight for the instance I in which there is only one target of weight w, and $W_I = w$ (i.e., standard star search for a single target in (Gal, 1974)). □

In order to prove Theorem 4 we will need the following lemma that addresses the extreme cases.

Lemma 7. *Given an instance I of* WS *for which $s_I = m$ or $W/w_{\max,I} = m$, we have $c(\textsc{AdSch}, I) \leq (3 + 2e)opt(I)$. Moreover, for every strategy Σ, there exists I such that $c(\Sigma, I) \geq (2 - \epsilon)opt(I)$, for arbitrarily small ϵ.*

Proof. The upper bound follows directly from the upper bounds in Lemmas 5 and 6. Remains thus to show the second part of the lemma. To this end, consider any strategy Σ that has to locate all m targets (as stipulated by the assumption). Consider any snapshot of the execution of the strategy at the moment the searcher returns to the origin, and let l_i denote the depth at which ray i has been searched. Consider an instance I in which m targets of unit weights are placed such that there is a target at distance $l_i + \epsilon$, for arbitrarily small ϵ, and $W_I = m$. Then

$$c(\Sigma, I) \geq 2 \sum_{0 \leq i \leq m-1} l_i + \text{opt}(I),$$

and for sufficiently small ϵ, we have that $2\sum_{i=0}^{m-1} l_i \geq (1 + \epsilon)\text{opt}(I)$, which yields the result. □

Having established bounds for the extreme cases, we can now proceed with the proofs of Theorems 2 and 3. Recall that these theorems establish our main parametric results on the competitive ratio of weighted search.

Proof of Theorem 2. The upper bound in the general case follows from Lemmas 5 and 6. For the lower bound (the general case), consider an instance I in which all targets have weight $w > 0$, and $W = tw$, for some $t \in \mathbb{R}^+$. Then any strategy must locate $\lceil t \rceil$ targets; from (Angelopoulos et al., 2014), the competitive ratio is at least $\phi(m - t) = \phi(m - \lceil W/w_{\max,I} \rceil)$. □

Proof of Theorem 3. The upper bound in the general case follows from Lemmas 5 and 6. For the lower bound in the general case, given $s < m$, consider an instance I in which $W = sw$, for some $w > 0$, and there are s targets of weight w, while all other targets have weight 0. Suppose that $s-1$ of positive-weight targets are very close to the origin, and can be found at negligible cost. Thus, on this instance, weighted search is as hard as locating one target in $m-(s-1)$ rays, and thus has competitive ratio

$$1 + 2\frac{b_{m-s+1,1}^{m-s+1}}{b_{m-s+1,1}-1} = 1 + 2\frac{b_{m-s}^{m-s+1}}{b_{m-s}-1} \overset{(2)}{=} \phi(m-s).$$

□

4 The competitive ratio of AdSub

In this section we determine the competitive ratio of AdSub, by proving Lemma 6. The following two technical lemmas provide some useful properties of the functions ϕ and b. Their proofs are based on Taylor series expansions and convexity arguments, and are given in Appendix A.

Lemma 8. *The function ϕ is increasing and $\phi(q) - \phi(q-1) \geq 2e$, for all $q \in \mathbb{N}^+$, where we define $\phi(0) = \lim_{x\to 0} \phi(x) = 3$.*

Lemma 9. *For $q, l \in \mathbb{N}$ let*

$$h_{q,\ell} = b_{q+1}^{-\ell-1}\left(\frac{b_q^\ell}{b_q-1} + 1 + b_{q+1}\right)(b_{q+1}-1).$$

Then $h_{q,l} \leq 1$ for all $1 \leq \ell \leq q+1$.

We obtain the following consequence of Lemma 8.

Corollary 10. *For every $x, y \in \mathbb{N}^+$*

$$\frac{b_x^{x+1}}{b_x-1} + 2y \leq \frac{b_x^{x+y+1}}{b_{x+y}-1}.$$

Proof. From Lemma 8 we know that for every $x \in \mathbb{N}^+$

$$\frac{b_x^{x+1}}{b_x - 1} - \frac{b_{x-1}^{x}}{b_{x-1} - 1} \geq e > 2.$$

Using a simple inductive argument, it follows that for every $y \in \mathbb{N}^+$,

$$\frac{b_x^{x+y+1}}{b_{x+y} - 1} - \frac{b_x^{x+1}}{b_x - 1} \geq e \cdot y > 2y.$$

□

We fix some notation that we will use throughout the proof of Lemma 6. Let $s = |S|$ denote the cardinality of the sought set of targets, and let t denote the total number of targets discovered by AdSub at the moment in which all targets in S have been found. That is, $f = t + 1$ upon termination. Clearly, we have $s \leq t \leq m$. For each $1 \leq j \leq t$ we say that the strategy in in *Phase* j if the number of targets discovered at that point in time is equal to $j - 1$. That is, the strategy starts in Phase 1, repeats lines 4–10 in the statement of Algorithm 1 until eventually a target is found, proceeds to Phase 2, repeats lines 4–10 in the statement until the second target is found, and so forth. Let $\ell_j \geq 1$ be the number of iterations of the loop in lines 4–10 performed in the j-th phase, with $1 \leq j \leq t$. Note that in Phase j, the number of rays explored unsuccessfully (i.e., without finding a target) equals $\ell_j - 1$, and in the ℓ_j-th iteration a target was discovered. Moreover, let D_j denote the distance of the j-th discovered target, $1 \leq j \leq t$. Since the last discovered target must be in S, we obtain the straightforward bound

$$d_S \geq D_t. \tag{4}$$

A short roadmap to the proof of Lemma 6 is as follows. We will assume first that $|S| < m$; at the end we argue how to handle the remaining case $|S| = m$. In the first step, we derive an upper bound on the algorithm's cost; see Lemma 11. This is a function of several parameters, in particular the distances of all the targets that have been discovered, and the cost of the unsuccessful explorations made by the searcher. In the second step we derive upper and lower bounds on the distances of the discovered targets; this is accomplished in Lemma 12. Note that the lower bound is particularly useful, since the last discovered target has to be in S and provides a lower bound for d_S. In the

third (and most technical) step, we combine all these bounds, using an inductive argument, to arrive at the desired conclusion. This will be accomplished in Lemma 13.

We begin by deriving a bound on the total distance traversed by the searcher, and we write $c(I) = \mathrm{c}(\textsc{AdSub}, I)$ for brevity. Define $Y_j = \prod_{1 \le i < j} b_{m,i}^{\ell_i - 1}$.

Lemma 11. *With the notation in this section*

$$c(I) \le \frac{2Y_{t+1} b_{m,t}}{b_{m,t} - 1} + 2 \sum_{1 \le j < t} (D_j + Y_{j+1}) + D_t. \tag{5}$$

Proof. In Phase 1 the strategy explores the rays in a cyclic fashion to distances $b_{m,1}, b_{m,1}^2, \ldots, b_{m,1}^{\ell_1 - 1}$ (going through each ray twice), and then discovers a target at distance D_1. Similarly, in Phase 2 it explores the rays to distances $b_{m,1}^{\ell_1 - 1} b_{m,2}, \ldots, b_{m,1}^{\ell_1 - 1} b_{m,2}^{\ell_2 - 1}$ and then discovers a target at distance D_2. More generally, in Phase j, $1 \le j < t$, the distance traversed equals

$$2 \cdot \prod_{1 \le i < j} b_{m,i}^{\ell_i - 1} \cdot \sum_{1 \le i < \ell_j} b_{m,j}^i + 2D_j = 2 \cdot \prod_{1 \le i < j} b_{m,i}^{\ell_i - 1} \cdot \frac{b_{m,j}^{\ell_j} - b_{m,j}}{b_{m,j} - 1} + 2D_j.$$

(As usual, the empty product equals 1.) Similarly, and because after locating the t-th target the searcher does not return to the origin, the distance traversed in the final phase equals

$$2 \cdot \prod_{1 \le i < t} b_{m,i}^{\ell_i - 1} \cdot \frac{b_{m,t}^{\ell_t} - b_{m,t}}{b_{m,t} - 1} + D_t.$$

Using the fact that $Y_j = \prod_{1 \le i < j} b_{m,i}^{\ell_i - 1}$, where $1 \le j \le t + 1$, we obtain

$$c(I) = 2 \sum_{1 \le j \le t} Y_j \frac{b_{m,j}^{\ell_j} - b_{m,j}}{b_{m,j} - 1} + 2 \sum_{1 \le j < t} D_j + D_t. \tag{6}$$

Note that for $1 \le j \le t$

$$Y_j \frac{b_{m,j}^{\ell_j} - b_{m,j}}{b_{m,j} - 1} = (Y_{j+1} - Y_j) \frac{b_{m,j}}{b_{m,j} - 1} = (Y_{j+1} - Y_j)(m - j + 1),$$

and consequently

$$\begin{aligned}\sum_{1\le j\le t} Y_j \frac{b_{m,j}^{\ell_j} - b_{m,j}}{b_{m,j}-1} &= \sum_{1\le j\le t} (Y_{j+1}-Y_j)(m-j+1) \\ &= Y_{t+1}\frac{b_{m,t}}{b_{m,t}-1} + \sum_{1\le j<t} Y_{j+1} - Y_1\cdot m \\ &\le Y_{t+1}\frac{b_{m,t}}{b_{m,t}-1} + \sum_{1\le j<t} Y_{j+1}.\end{aligned}$$

Substituting this into (6) yields the statement of the lemma. □

Having accomplished the task of providing an appropriate upper bound for AdSub we proceed with deriving explicit bounds for the distances of the targets located by the strategy.

Lemma 12. *Let D_j be the distance of the j-th discovered target by* AdSub*. Then*

$$D_j \le Y_j b_{m,j}^{\ell_j} = Y_{j+1} b_{m,j}, \quad \textit{for all } \ 1\le j\le t. \tag{7}$$

Moreover, suppose that the j-th discovered target is on ray $0\le r\le m-j$, and let j' with $1\le j'\le j$ be such that ray r was last explored in Phase j', prior to discovering the target in Phase j. Then

$$D_j \ge Y_{j'}\cdot b_{m,j'}^{-m+j+\sum_{j'\le i\le j}(\ell_i-1)}. \tag{8}$$

Proof. In Phase j, with $1\le j\le t$, the largest distance to which a ray is explored is $Y_j\cdot b_{m,j}^{\ell_j}$. Thus $D_j \le Y_j b_{m,j}^{\ell_j} = Y_{j+1} b_{m,j}$. For the lower bound, note that there exists x_j with $1\le x_j\le \ell_{j'}-1$ such that this ray was explored up to a depth of $Y_{j'} b_{m,j'}^{x_j}$, and so $D_j \ge Y_{j'} b_{m,j'}^{x_j}$. The number of targets discovered between the last time ray r was explored before discovering the target at distance D_j, and the time at which this target was discovered is equal to $j-j'+1$ (including the said target). Since the number of remaining rays in Phase j' equals $m-j'+1$, we obtain

$$m-j'+1 = (j-j'+1) + \sum_{i=j'+1}^{j} (\ell_i-1) + (\ell_{j'}-1-x_j).$$

Thus, $\sum_{j' \le i \le j}(\ell_i - 1) - m + j = x_j$, and so $D_j \ge Y_{j'} \cdot b_{m,j'}^{-m+j+\sum_{j' \le i \le j}(\ell_i - 1)}$. □

Moreover, for x_j as defined in the proof of Lemma 12, since $1 \le x_j \le \ell_{j'} - 1$ we obtain the bounds

$$\sum_{j' \le i \le j} (\ell_i - 1) \ge m - j + 1 \text{ and } \sum_{j' < i \le j} (\ell_i - 1) \le m - j, \tag{9}$$

that will be useful later. We also obtain a simpler lower bound on D_j, namely

$$D_j \ge Y_j \cdot b_{m,j}^{\ell_j - m + j - 1} \quad \text{for all} \quad \ell_j \ge 1. \tag{10}$$

To see why (10) holds, note that from definition of Y_j, we have that

$$Y_{j'} = Y_j \prod_{j' \le i < j} b_{m,i}^{1-\ell_i} \ge Y_j \prod_{j' \le i < j} b_{m,j}^{1-\ell_i} = Y_j b_{m,j}^{\ell_j - 1} \prod_{j' \le i \le j} b_{m,j}^{1-\ell_i} \ge Y_j b_{m,j}^{\ell_j - 1} b_{m,j}^{j-m},$$

where the first inequality follows from the fact that $b_{m,h}$ is increasing in h, and the last inequality follows from (9).

We introduce some additional notation to facilitate the further analysis. Combining (10) with (7) we infer that for every $1 \le j \le t$ there exist γ_j such that

$$D_j = \gamma_j \cdot Y_{j+1}, \quad \text{where} \quad b_{m,j}^{-m+j} \le \gamma_j \le b_{m,j}.$$

Moreover, let J be the set of indexes in $\{0, \ldots, t-1\}$ such that for each $j \in J$ we have that the target discovered in Phase j is in S. That is, we have $|J| = |S| - 1$ and moreover, strengthening the bound in (4)

$$d_S \ge \sum_{j \in J} D_j + D_t = \sum_{j \in J} \gamma_j \cdot Y_{j+1} + D_t. \tag{11}$$

Let also $\overline{J} = \{1, \ldots, t-1\} \setminus J$.

With this notation in place, we can now bound the competitive ratio of AdSub, towards the proof of Lemma 6. We will use the simple fact that for $a_1, \ldots, a_N, b_1, \ldots, b_N > 0$

$$\frac{\sum_{i=1}^N a_i}{\sum_{i=1}^N b_i} \le \max_{1 \le i \le n} \left\{ \frac{a_i}{b_i} \right\}.$$

Specifically, by applying Lemma 11 and (11), we obtain that

$$\begin{aligned}\frac{c(I)}{d_S} &\leq \frac{\frac{2Y_{t+1}b_{m,t}}{b_{m,t}-1} + 2\sum_{1\leq j<t}(1+\gamma_j)Y_{j+1} + D_t}{\sum_{j\in J}\gamma_j \cdot Y_{j+1} + D_t} \\ &= \frac{\frac{2Y_{t+1}b_{m,t}}{b_{m,t}-1} + 2(\sum_{j\in J}(1+\gamma_j)Y_{j+1} + \sum_{j\in \overline{J}}(1+\gamma_j)Y_{j+1}) + D_t}{\sum_{j\in J}\gamma_j \cdot Y_{j+1} + D_t} \\ &\leq 2\max\Big\{\max_{j\in J}\frac{1+\gamma_j}{\gamma_j}, H\Big\} + 1,\end{aligned}$$

where H is defined as

$$H = \frac{\frac{Y_{t+1}b_{m,t}}{b_{m,t}-1} + \sum_{j\in\overline{J}} Y_{j+1}(1+\gamma_j)}{D_t}. \tag{12}$$

Note that the function $(1+x)/x$ is decreasing in x. Therefore, as $\gamma_j \geq b_{m,j}^{-m+j}$, for any $j \in J$ we have

$$\frac{1+\gamma_j}{\gamma_j} \leq 1 + b_{m,j}^{m-j} = 1 + \left(1+\frac{1}{m-j}\right)^{m-j} \leq 1 + e < 4. \tag{13}$$

Therefore, (4) gives

$$c(I) \leq \max\{9, 1+2H\}\ d_S. \tag{14}$$

Recall that we assume that $|S| < m$; at the end we will argue how to handle the remaining case $|S| = m$. The crucial step in the proof of Lemma 6 will be to show that

$$H \leq \frac{b_q^{q+1}}{b_q - 1},\ \text{where } q = m - |S|. \tag{15}$$

This will suffice to prove the lemma in the case $|S| < m$. This is because, from Lemma 8, it follows that $1+2H \geq 9$, therefore from (14), the competitive ratio of AdSub is at most $1+2H \leq \phi(m-|S|)$, for $|S| < m$. Note that (14) confirms what one expects intuitively, namely that the contribution from targets in S to the competitive ratio is not significant, and bounded by 9, whereas the contribution of targets that are not in S to the competitive ratio is more substantial, and more challenging to bound. In order to bound H, we first define $L_{a,b}$ and $Y_{a,b}$ as

$$L_{a,b} = \sum_{a\leq i\leq b}(\ell_i - 1) \ \text{ and } \ Y_{a,b} = \frac{Y_{b+1}}{Y_a}.$$

Using the definition of H (12) , the lower bound on D_t (8) and the fact $\gamma_j \leq b_{m,j}$, we obtain that

for some $t' \leq t$,

$$H \leq b_{m,t'}^{m-t-L_{t',t}} \left(\frac{b_{m,t} Y_{t',t}}{b_{m,t} - 1} + \sum_{j \in \overline{J}} (1 + b_{m,j}) Y_{t',j} \right). \tag{16}$$

This expression is central in our analysis and will indeed help us bound the contribution of targets that are not in S. We first define a partition of $\overline{J}$ into two sets which contain elements in $\overline{J}$ that are greater than, and respectively smaller than or equal to $t-1$; namely we define $\overline{J}_> = \overline{J} \cap \{t', \ldots, t-1\}$ and $\overline{J}_\leq = \overline{J} \cap \{1, \ldots, t'-1\}$. Regarding any $j \in \overline{J}_\leq$, note that $Y_{t',j} \leq 1$ and hence $b_{m,t'}^{m-t-L_{t',t}}(1 + b_{m,j}) Y_{t',j} \leq b_{m,t'}^{m-t-L_{t',t}} \cdot (1 + b_{m,j})$. From (9) we infer that $L_{t',t} \geq m - t + 1$ and so the previous expression is bounded by at most $(1 + b_{m,j})/b_{m,t'}$. Thus, we obtain that

$$b_{m,t'}^{m-t-L_{t',t}}(1 + b_{m,j}) Y_{t',j} < \frac{1 + b_{m,t'}}{b_{m,t'}} \leq 2, \quad j \in \overline{J}_\leq. \tag{17}$$

In words, (17) shows that each element in $\overline{J}_\leq$ contributes at most an additive 2 to the bound in (16). Moreover, note that for every $j \in \overline{J}_\leq$

$$(1 + b_{m,j}) Y_{t',j} \leq 1 + b_{m,j} = 1 + \frac{1}{m-j} < 1 + \frac{1}{m-t'},$$

whereas for every $j \in \overline{J}_>$

$$(1 + b_{m,j}) Y_{t',j} \geq 1 + b_{m,j} = 1 + \frac{1}{m-j} \geq 1 + \frac{1}{m-t'}.$$

Therefore, the contribution to H of every $j \in \overline{J}_\leq$ is smaller than the corresponding contribution of every $j \in \overline{J}_>$. Thus, in order to bound H we can assume, without loss of generality, that $\overline{J}_>$ has maximal cardinality.

Combining (16), (17) and the observation on the maximality of $\overline{J}_>$, we have that

$$H \leq G + 2|\overline{J}_\leq|, \text{ where } G = b_{m,t'}^{m-t-L_{t',t}} \left(\frac{b_{m,t} Y_{t',t}}{b_{m,t} - 1} + \sum_{j \in \overline{J}_>} (1 + b_{m,j}) Y_{t',j} \right). \tag{18}$$

The following is the main technical lemma of this section. The lemma will help us bound G, which in turn will help us prove (15).

Lemma 13.

$$G \leq \frac{b_{Q+|\overline{J}_>|}^{Q+|\overline{J}_>|+1}}{b_{Q+|\overline{J}_>|}-1}, \text{ where } Q = m-t.$$

Proof. For ease of notation, let us denote $|\overline{J}_>|$ by d. Since $\overline{J}_> = \overline{J} \cap \{t', \ldots, t-1\}$, and since we can assume that $\overline{J}_>$ has maximal cardinality, we have that $d = t - t'$.

We will prove the lemma by induction on d. The base case, $d = 0$, follows trivially by direct substitution. To give some intuition into the inductive step, let us also show the lemma in the case $d = 1$, namely when $t' = t - 1$. By substituting into the expression of G (18), we have that

$$G = b_{Q+1}^{Q-(\ell_t-1)} \left(\frac{b_Q^{\ell_t}}{b_Q - 1} + (1 + b_{Q+1}) \right),$$

and since $\ell_t \geq 1$, by applying Lemma 9,

$$G \leq b_{Q+1}^{Q-(\ell_t-1)} \frac{b_{Q+1}^{\ell_t+1}}{b_{Q+1}-1} = \frac{b_{Q+1}^{Q+2}}{b_{Q+1}-1}.$$

For the inductive step, let us denote by G_d the value of G given that $t - t' = d$. By substituting into (18), we obtain that G_d can be expressed as

$$G_d = b_{Q+d}^{Q-\sum_{x=0}^{d}(\ell_{t-x}-1)} \left(\frac{b_Q \prod_{x=0}^{d} b_{Q+x}^{\ell_{t-x}-1}}{b_Q - 1} + \sum_{y=t-d}^{t-1} (1 + b_{m-y}) \prod_{x=t'}^{y} b_{m,x}^{\ell_x - 1} \right). \tag{19}$$

We also have that

$$\prod_{x=0}^{d} b_{Q+x}^{\ell_{t-x}-1} = b_{Q+d}^{\ell_{t-d}-1} \prod_{x=0}^{d-1} b_{Q+x}^{\ell_{t-x}-1}, \tag{20}$$

and

$$\begin{aligned} \sum_{y=t-d}^{t-1} (1 + b_{m-y}) \prod_{x=t-d}^{y} b_{m,x}^{\ell_x-1} &= \sum_{y=t-(d-1)}^{t-1} (1 + b_{m-y}) \prod_{x=t-d}^{y} b_{m,x}^{\ell_x-1} + (1 + b_{Q+d}) b_{Q+d}^{\ell_{t-d}-1} \\ &= b_{Q+d}^{\ell_{t-d}-1} \left(\sum_{y=t-(d-1)}^{t-1} (1 + b_{m-y}) \prod_{x=t-(d-1)}^{y} b_{m,x}^{\ell_x-1} + (1 + b_{Q+d}) \right). \end{aligned} \tag{21}$$

Denote by F_d the quantity $b_{Q+d}^{Q-\sum_{x=0}^{d}(\ell_{t-x}-1)}$. By substituting (20) and (21) into (19) we have that

$$G_d = F_d \cdot b_{Q+d}^{\ell_{t-d}-1}\left(\sum_{y=t-(d-1)}^{t-1}(1+b_{m-y})\prod_{x=t-(d-1)}^{y} b_{m,x}^{\ell_x-1} + (1+b_{Q+d})\right),$$

and we can thus obtain a recursive expression for G_d, namely

$$G_d = F_d \cdot b_{Q+d}^{\ell_{t-d}-1}\left(\frac{G_{d-1}}{F_{d-1}} + 1 + b_{Q+d}\right). \tag{22}$$

From the induction hypothesis, we have

$$G_{d-1} \le \frac{b_{Q+d-1}^{Q+d}}{b_{Q+d-1}-1}.$$

Let L_i denote the expression $\sum_{x=0}^{i}(\ell_{t-x}-1)$. Then $F_{d-1} = b_{Q+d-1}^{Q-L_{d-1}}$, and $F_d = b_{Q+d}^{Q-L_d}$. Therefore, $\frac{G_{d-1}}{F_{d-1}} = b_{Q+d-1}^{L_{d-1}+d}/b_{Q+d-1}-1$, and from (22) we obtain

$$\begin{aligned} G_d &\le b_{Q+d}^{Q-L_d} b_{Q+d}^{\ell_{t-d}-1}\left(\frac{b_{Q+d-1}^{L_{d-1}+d}}{b_{Q+d-1}-1} + 1 + b_{Q+d}\right) \\ &\le b_{Q+d}^{Q-L_{d-1}}\frac{b_{Q+d}^{L_{d-1}+d+1}}{b_{Q+d}-1} && \text{(Lemma 9)} \\ &= \frac{b_{Q+d}^{Q+d+1}}{b_{Q+d}-1}, \end{aligned}$$

where the conditions for applying Lemma 9 follow from the fact that $L_{d-1} \le Q$, from (10). This completes the inductive step and the proof of the lemma. □

We are now ready to prove the competitiveness of AdSub using the above lemma.

Proof of Lemma 6. Consider first the case $|S| < m$. Recall that it suffices to show (15). We have $|\overline{J}| = |\{1,\ldots,t-1\}\setminus J| = t-1-(|S|-1) = t-|S|$. Moreover, $|\overline{J}_{>}| = d$. Therefore, $|\overline{J}_{\le}| = t-|S|-d$. Since $H \le G + 2|\overline{J}_{\le}|$, from Lemma 13 we obtain that

$$H \le \frac{b_{m-t+d}^{m-t+d+1}}{b_{m-t+d}-1} + 2(t-|S|-d) \le \frac{b_{m-|S|}^{m-|S|+1}}{b_{m-|S|}-1},$$

where the second inequality follows from Corollary 10. We conclude that (15) holds.

It remains to consider the case $|S| = m$. Here, we adapt (5). Recall that J is the set of indexes in $\{0, \ldots, t-1\}$ such that for each $j \in J$ we have that the target discovered in Phase j is in S. Thus $J = \{0, \ldots, t-1\}$ and $\overline{J} = \emptyset$ in this case. Moreover, note that $\ell_t = \ell_m = 1$, as the last remaining ray is searched until a target is found without the searcher returning to the origin. Then, the bound in (5) changes to

$$c(I) \leq \frac{2Y_m b_{m,m-1}}{b_{m,m-1} - 1} + 2\sum_{j=1}^{m-2}(D_j + Y_{j+1}) + 2D_{m-1} + D_m.$$

Note that $b_{m,m-1} = 2$. Proceeding as in (4) we get that

$$\frac{c(I)}{d_S} \leq 1 + 2\max\left\{\max_{j\in J}\frac{1+\gamma_j}{\gamma_j}, \frac{2D_{m-1}}{2/3D_{m-1}}, \frac{4Y_m}{D_{m-1}/3 + D_m}\right\}.$$

From (13), we know that $\frac{1+\gamma_j}{\gamma_j} \leq 1 + e$. Moreover, we have that $D_m \geq Y_m$ and $D_{m-1} \geq Y_m/2$ by (10). This implies that $\frac{c(I)}{d_S} \leq 1 + 2(1+e)$, and the proof is completed. □

5 Adaptive contract scheduling

In this section, we illustrate an application of SubsetSearch in the context of a classic topic in AI and real-time systems: the design and performance analysis of *interruptible* algorithms. Such algorithms have the appealing property that their performance improves gradually as function of the available computation time, and can thus output a reasonably efficient solution even if interrupted at some point in time. Note that not all algorithms have this property. For example, dynamic programming algorithms may very well output a meaningless solution if, say, they are interrupted before they fill in the last important column or row in their DP table. Russell and Zilberstein (Russell and Zilberstein, 1991) call such algorithms *contract* algorithms, in the sense that the required computation time is given as part of the input to the algorithm; the algorithm is guaranteed to output a meaningful, and correct output only if it is allowed an execution time at least as large as this required computation time.

A natural problem that arises is converting a contract algorithm to an interruptible equivalent. This was first studied in (Russell and Zilberstein, 1991), who gave a simple, yet general

approach: repeat the contract algorithm with progressively increasing execution times, so that, if an interruption occurs, the system can benefit from the "longest" execution that has been completed; this technique is known as *contract scheduling*. Since then, this problem has been studied in many variants and settings, e.g., (Bernstein et al., 2002; Zilberstein et al., 2003; Angelopoulos and López-Ortiz, 2009; Bernstein et al., 2003; López-Ortiz et al., 2014; Angelopoulos et al., 2008; Angelopoulos, 2015; Angelopoulos and Jin, 2019; Angelopoulos and Kamali, 2021). One of the generalizations of this problem involves n different *problem instances* and a single contract algorithm (Zilberstein et al., 2003), and the objective is to obtain an interruptible algorithm over all n instances. This can be accomplished by a *schedule* of the form $X = ((p_i, x_i)_{i=0}^{\infty}$. Here, the i-th execution of the contract algorithm (to which we will refer as the i-th *contract*) applies to the instance $p_i \in \{0, \ldots, n-1\}$, and is run for time (length) $x_i \in \mathbb{R}^+$.

In all previous work on contract scheduling, the performance of a schedule X is evaluated by means of the *acceleration ratio* (Russell and Zilberstein, 1991). This is a worst-case measure, akin to the competitive ratio, that captures the overhead incurred by the repeated executions, and which is defined as follows. Given the schedule X, and $p_i \in \{0, \ldots, n-1\}$, let $l(X, p)$ denote the length of the longest contract for p_i that is completed by time t in X. We also define

$$\ell(X, t) = \min_{p \in \{0, \ldots, n-1\}} l(X, p),$$

i.e., $\ell(X, t)$ is a measure of the progress X has achieved on its least worked instance. The *acceleration ratio* of X is defined as

$$\text{acc}(X) = \sup_t \frac{t}{\ell(X, t)}. \tag{23}$$

For contract scheduling with n problem instances, (Zilberstein et al., 2003) showed an optimal acceleration ratio equal to $(\phi(n) - 1)/2$, which is achieved using the cyclic schedule $X = ((i \mod n, ((n+1)/n)^i)_{i=0}^{\infty}$. One may observe similarities with the ray search problem; indeed the two problems share connections, as demonstrated in (Bernstein et al., 2003; Angelopoulos, 2015).

All previous studies of contract scheduling considered a *static* setting. Namely, the schedule is determined ahead of time, and consists of an infinite number of contract executions for each of the n instances. This is to model the requirement that interruptions may appear arbitrarily far in the future. In practice, however, the system will reach a point at which sufficient progress

has been made in one or more problem instances (i.e., the instance has been *solved*), and thus would be preferable to allocate computational resources only to the remaining instances, from that point onwards. This motivates a *dynamic* setting, in which the schedule must adapt according to instances that have been solved.

More precisely, suppose that the schedule completes the execution of a contract $c_i = (p_i, x_i)$, for instance p_i. If the system deems the instance p_i as solved, then no future contract need to be assigned to problem p_i, i.e., for all $c_j = (p_j, x_j)$, with $j > i$, we may assume that $p_j \neq p_i$. Let S_t denote the set of solved instances by time t. Then, we define $\ell(X, t)$ as $\min_{p \notin S_t} l(X, p)$ (in words, the least progress X has made at time t among yet unsolved instances). The definition of the acceleration ratio then follows again from (23). We refer to this problem as the *adaptive contract scheduling* problem. We show how to obtain an optimal schedule for this problem, using ideas from the analysis of the Subset Search problem.

Theorem 14. *There is a schedule for adaptive contract scheduling such that, for any given time t, if the interruption occurs after time t, then its acceleration ratio is at most $(\phi(n - s_t) - 1)/2$, where $s_t < n$[1] is the number of problem instances that have been solved by time t. Furthermore, this bound is tight.*

Proof. Consider the schedule obtained by Algorithm 2, and which is motivated by Algorithm 1 for SS, with a notable difference. Since s (the number of solved instances) is initialized to 0, the multiplicative update in lines 5 and 8 of Algorithm 2 is somewhat smaller. For example, the first contract x_0 has length $(n + 1)/n$, whereas in Algorithm 1, the first searched length is $m/(m - 1)$. We note also the that the relabeling in line 13 is solely for the purpose of assigning contract lengths; all problem instances maintain their original naming.

Let $X = ((p_i, x_i))_i$ denote the schedule of Algorithm 2. Consider an interruption at time t, which, in worst case, may occur right before a contract execution, say (p_j, x_j), for some index j, is about to complete. By the cyclic nature of the algorithm, we have that

$$\frac{t}{\ell(X, t)} \leq \frac{\sum_{i=0}^{j} x_i}{x_{j-n+s_t}}. \tag{24}$$

We will argue that this expression does not exceed $(\phi(n - s_t) - 1)/2$. Consider an execution of

[1]Note that in the statement of Theorem 14, it cannot be that $s_t = n$, since in that case all problems would have been solved by time t, and the system would have reached the end of all executions.

Algorithm 2: Adaptive Contract Scheduling

Input: n problem instances labeled $\{0, \ldots, n-1\}$;
$s \leftarrow 0$, $r \leftarrow 0$, $D \leftarrow 1, i \leftarrow 0$
while *there are instances that are yet unsolved* **do**
  repeat
    execute the i-th contract algorithm on the r-th instance and for length $x_i = D \cdot b_{n,s}$
    if *the instance is solved* **then**
      $r \leftarrow r + 1 \pmod{n-s}$
      $D \leftarrow D \cdot b_{n,s}$
    end
    $i \leftarrow i + 1$
  until *an instance is deemed solved*
  $s \leftarrow s + 1$
  relabel the instances canonically from $0 \ldots n-1-s$
  $r \leftarrow r \pmod{n-s}$
end

AdSub (Algorithm 1) for SS, on an instance with $m = n + 1$ rays, constructed as follows. If in the execution of Algorithm 2, a problem instance p_i is deemed solved after a contract of the form (p_i, x_i) is completed, then there is a target in ray p_i+1, and at distance x_i from the origin; however, this target is *not* one of the targets sought. Last, there is only one target sought, namely in ray p_{j-n+s_t}, and at distance infinitessimally larger than x_{j-n+s_t}, for the j specified in (24). Then, AdSub finds the target at cost

$$C = 2\frac{\sum_{i=0}^{j} x_i}{x_{j-n+s_t}} + x_{j-n+s_t}.$$

Note that in the above instance for SS, there are $s_t + 1$ targets sought among $n + 1$ rays. From Lemma 6 we know that for $s_t + 1 < n + 1$, i.e., for $s_t < n$, we have

$$\frac{C}{x_{j-n+s_t}} \leq \phi(n + 1 - (s_t + 1)) \Rightarrow \frac{\sum_{i=0}^{j} x_i}{x_{j-n+s_t}} \leq \frac{\phi(n - s_t) - 1}{2},$$

which establishes, with (24) the upper bound on the acceleration ratio for the schedule of Algorithm 2.

To see that this bound is tight, we know from (Zilberstein et al., 2003) that the acceleration ratio for contract scheduling with n problem instances, in the standard, static variant, is $(\phi(n) - 1)/2$, which is attained for interruptions T with $T \to \infty$. This implies that if $s_t < n$ instances are solved

by some given (and thus finite) time t, then $n - s_t$ problem instances remain. Therefore, for $T \to \infty$, the contract lengths corresponding to the s_t solved instances are infinitesimal in comparison to T, and thus we obtain a lower bound on the acceleration ratio equal to $(\phi(n - s_t) - 1)/2 - \epsilon$, for arbitrarily small $\epsilon > 0$. □

6 Conclusions & Further Research

We introduced and studied a generalization of the classic star search problem in which each ray has a weighted target, and the objective for the searcher is to locate targets with a certain aggregate. We showed that weighted search can be reduced to another problem, namely the subset search problem, and we proposed and analyzed an efficient search strategy for both problems. The crux in the analysis is to apply a parameterized (or adaptive) approach, which incorporates parameters that capture the "hardness" of the instance. We also demonstrated an application of weighted search in the design of interruptible algorithms based on adaptive contract scheduling.

Weighted search has applications in other domains, e.g., in the setting of *caching games* such as the scatter hoarder's problem of (Alpern et al., 2011), in which a hider distributes resources across a number of locations, and a searcher tries to retrieve them given some bound on the total search cost. Another research direction is to study weighted star search under measures beyond the competitive ratio; see e.g., (Kirkpatrick, 2009) for such a study in the unweighted setting. Moreover, it would be compelling to consider search problems in which the searcher has some limited information about the setting. For example, a relevant scenario is when the weights or the distances are known up to a permutation of the rays. Designing optimal algorithms, under parameterized analysis in such settings is an important future research challenge.

7 Acknowledgments

This research benefited from visiting fellowships by the Center for Advanced Studies of the University of Munich (CAS-LMU) and by Sorbonne University. Konstantinos Panagiotou received funding from the European Research Council, ERC Grant Agreement 772606-PTRCSP.

References

Alpern, S., Fokkink, R., Lidbetter, T., and Clayton, N. S. (2011). A search game model of the scatter hoarder's problem. *Journal of the Royal Society Interface*, 9(70):869–879.

Alpern, S. and Gal, S. (2003). *The theory of search games and rendezvous*. Kluwer Academic Publishers.

Angelopoulos, S. (2015). Further connections between contract-scheduling and ray-searching problems. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1516–1522.

Angelopoulos, S. (2021). Online search with a hint. In *Proceedings of the 12th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 51:1–51:16.

Angelopoulos, S., Arsénio, D., and Dürr, C. (2017). Infinite linear programming and online searching with turn cost. *Theoretical Computer Science*, 670:11–22.

Angelopoulos, S., Arsénio, D., Dürr, C., and López-Ortiz, A. (2016). Multi-processor search and scheduling problems with setup cost. *Theory of Computing Systems*, pages 1–34.

Angelopoulos, S., Dürr, C., and Lidbetter, T. (2019). The expanding search ratio of a graph. *Discrete Applied Mathematics*, 260:51–65.

Angelopoulos, S. and Jin, S. (2019). Earliest completion scheduling of contract algorithms with end guarantees. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5493–5499.

Angelopoulos, S. and Kamali, S. (2021). Contract scheduling with predictions. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 11726–11733. AAAI Press.

Angelopoulos, S. and Lidbetter, T. (2020). Competitive search in a network. *European Journal of Operational Research*, 286(2):781–790.

Angelopoulos, S. and López-Ortiz, A. (2009). Interruptible algorithms for multi-problem solving. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 380–386.

Angelopoulos, S., López-Ortiz, A., and Hamel, A. (2008). Optimal scheduling of contract algorithms with soft deadlines. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 868–873.

Angelopoulos, S., López-Ortiz, A., and Panagiotou, K. (2014). Multi-target ray searching problems. *Theoretical Computer Science*, 540:2–12.

Baeza-Yates, R., Culberson, J., and Rawlins, G. (1993). Searching in the plane. *Information and Computation*, 106:234–244.

Beck, A. (1964). On the linear search problem. *Naval Research Logistics*, 2:221–228.

Beck, A. and Newman, D. (1970). Yet more on the linear search problem. *Israel Journal of Mathematics*, 8:419–429.

Bellman, R. (1963). An optimal search problem. *SIAM Review*, 5:274.

Bernstein, D. S., Finkelstein, L., and Zilberstein, S. (2003). Contract algorithms and robots on rays: unifying two scheduling problems. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1211–1217.

Bernstein, D. S., Perkins, T. J., Zilberstein, S., and Finkelstein, L. (2002). Scheduling contract algorithms on multiple processors. In *Proceedings of the Eighteenth AAAI Conference on Artificial Intelligence (AAAI)*, pages 702–706.

Bonato, A., Georgiou, K., MacRury, C., and Pralat, P. (2020). Probabilistically faulty searching on a half-line - (extended abstract). In *LATIN 2020: Theoretical Informatics - 14th Latin American Symposium, São Paulo, Brazils*, volume 12118 of *Lecture Notes in Computer Science*, pages 168–180. Springer.

Bose, P., Carufel, J. D., and Durocher, S. (2015). Searching on a line: A complete characterization of the optimal solution. *Theoretical Computer Science*, 569:24–42.

Condon, A., Deshpande, A., Hellerstein, L., and Wu, N. (2009). Algorithms for distributional and adversarial pipelined filter ordering problems. *ACM Transaction on Algorithms*, 5(2):24:1–24:34.

Czyzowicz, J., Kranakis, E., Krizanc, D., Narayanan, L., and Opatrny, J. (2019). Search on a line with faulty robots. *Distributed Comput.*, 32(6):493–504.

Demaine, E., Fekete, S., and Gal, S. (2006). Online searching with turn cost. *Theoretical Computer Science*, 361:342–355.

Gal, S. (1972). A general search game. *Israel Journal of Mathematics*, 12:32–45.

Gal, S. (1974). Minimax solutions for linear search problems. *SIAM Journal on Applied Mathematics*, 27:17–30.

Gal, S. (2010). Search games. *Wiley Encyclopedia of Operations Research and Management Science*.

Gal, S. and Chazan, D. (1976). On the optimality of the exponential functions for some minimax problems. *SIAM Journal on Applied Mathematics*, 30:324—-348.

Hipke, C., Icking, C., Klein, R., and Langetepe, E. (1999). How to find a point in the line within a fixed distance. *Discrete Applied Mathematics*, 93:67–73.

Jaillet, P. and Stafford, M. (1993). Online searching. *Operations Research*, 49:234–244.

Kao, M.-Y. and Littman, M. (1997). Algorithms for informed cows. In *Proceedings of the AAAI 1997 Workshop on Online Search*.

Kao, M.-Y., Ma, Y., Sipser, M., and Yin, Y. (1998). Optimal constructions of hybrid algorithms. *Journal of Algorithms*, 29(1):142–164.

Kao, M.-Y., Reif, J., and Tate, S. (1996). Searching in an unknown environment: an optimal randomized algorithm for the cow-path problem. *Information and Computation*, 131(1):63–80.

Kirkpatrick, D. G. (2009). Hyperbolic dovetailing. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA)*, pages 616–627.

Kupavskii, A. and Welzl, E. (2018). Lower bounds for searching robots, some faulty. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 447–453.

López-Ortiz, A., Angelopoulos, S., and Hamel, A. (2014). Optimal scheduling of contract algorithms for anytime problems. *Journal of Artificial Intelligence Research*, (51):533–554.

López-Ortiz, A. and Schuierer, S. (2004). On-line parallel heuristics, processor scheduling and robot searching under the competitive framework. *Theoretical Computer Science*, 310(1–3):527–537.

McGregor, A., Onak, K., and Panigrahy, R. (2009). The oil searching problem. In *Proceedings of the 17th European Symposium on Algorithms (ESA)*, pages 504–515.

Russell, S. J. and Zilberstein, S. (1991). Composing real-time systems. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 212–217.

Schuierer, S. (2003). A lower bound for randomized searching on m rays. In *Computer Science in Perspective*, pages 264–277.

Zilberstein, S., Charpillet, F., and Chassaing, P. (2003). Real-time problem-solving with contract algorithms. *Annals of Mathematics and Artificial Intelligence*, 39(1–2):1–18.

A Technical proofs

Proof of Lemma 8. Define the function

$$h(x) = (x+1)\left(1+\frac{1}{x}\right)^x.$$

Then $\phi(x) = 1 + 2h(x)$, and the monotonicity of both ϕ and h follows from the simple fact that $(1+1/x)^x$ is increasing. Moreover, by direct computation we obtain that $\phi(1) - \phi(0) > 2e$ and $\phi(2) - \phi(1) > 2e$. Let

$$H(x) = h(x) - h(x-1).$$

We will argue that

$$H(x) \leq H(x-1) \quad \text{for} \quad x \geq 3. \tag{25}$$

Given (25) the claim that $\phi(q) - \phi(q-1) \geq 2e$, for all $q \in \mathbb{N}^+$, can be shown as follows. Since $(1+1/x)^x$ is monotone increasing, we have that

$$\begin{aligned}
\lim_{x\to\infty} H(x) &= \lim_{x\to\infty} (x+1)\left(1+\frac{1}{x}\right)^x - x\left(1+\frac{1}{x-1}\right)^{x-1} \\
&\leq \lim_{x\to\infty} x\left(1+\frac{1}{x-1}\right)^{x-1} - (x-1)\left(1+\frac{1}{x-2}\right)^{x-2} && \text{(From (25))} \\
&\leq \lim_{x\to\infty} \left(1+\frac{1}{x-2}\right)^{x-2} && \text{(monotonicity of } (1+1/x)^x\text{)} \\
&= e,
\end{aligned}$$

and

$$\lim_{x\to\infty} H(x) \geq \lim_{x\to\infty} \left(1+\frac{1}{x-1}\right)^{x-1} = e.$$

Thus, $\lim_{x\to\infty} H(x) = e$. Moreover, (25) guarantees that $(H(q))_{q\in\mathbb{N}^+}$ is a non-increasing sequence. Summarizing, we obtain that $H(x) \geq e$ for all $x \geq 3$ and thus

$$h(x) - h(x-1) \geq e \implies \phi(x) - \phi(x-1) \geq 2e,$$

as claimed.

In order to show (25), note that it is equivalent to

$$h(x-1) \geq \frac{h(x)+h(x-2)}{2} \quad \text{for} \quad x \geq 3.$$

We will argue that $h''(x) < 0$ for $x \geq 3$, which shows that h is concave for $x \geq 3$ and thus satisfies the above inequality, which completes the proof. Basic calculus reveals that

$$h''(x) = \frac{h(x)t(x)}{x(x+1)}, \quad \text{where} \quad t(x) = x(x+1)\log\left(1+\frac{1}{x}\right)^2 - 1.$$

Since $h(x)$ and $x(x+1)$ are positive, it remains to show that $t(x) < 0$ for $x \geq 3$. Using the Taylor series expansion of the logarithm we infer that

$$\ln(1+y) \leq y - y^2/2 + y^3/3 - y^4/4 + y^5/5, \quad |y| < 1.$$

Thus

$$t(x) \leq x(x+1)\left(\frac{1}{x} - \frac{1}{2x^2} + \frac{1}{3x^3} - \frac{1}{4x^4} + \frac{1}{5x^5}\right)^2 - 1.$$

By expanding the right-hand side we obtain that

$$t(x) \leq \frac{\alpha_7 x^7 + \alpha_6 x^6 + \alpha_5 x^5 + \alpha_4 x^4 + \alpha_3 x^3 + \alpha_2 x^2 + \alpha_1 x + \alpha_0}{3600x^9},$$

with

$$\alpha_7 = -300, \alpha_6 = 300, \alpha_5 = -260, \alpha_4 = 1420,$$

and

$$\alpha_3 = -615, \alpha_2 = 345, \alpha_1 = -216, \alpha_0 = 144.$$

Note that the upper bound for t is less than 0 for $x \in \{3, 4, 5\}$. Moreover, the values of the α_i's guarantee that for all $x \geq 3$

$$\alpha_i x^i + \alpha_{i-1} x^{i-1} = (\alpha_i x + \alpha_{i-1}) x^{i-1} \leq 0, \quad i \in \{7, 3, 1\},$$

and thus $t(x) \leq (-260x^5 + 1420x^4)/3600x^9$. However, for $x \geq 6$, this is negative (since $6 \cdot 260 > 1420$), and the proof is completed. □

Proof of Lemma 9. Substituting the value of b_q and b_{q+1} yields after some straightforward algebraic manipulations

$$h_{q,\ell} = \left(q\Big(1 + \frac{1}{q(q+2)}\Big)^{\ell} + \frac{2q+3}{q+1}\Big(1 - \frac{1}{q+2}\Big)^{\ell}\right)\frac{1}{q+2}. \tag{26}$$

When viewed as a function of ℓ, $h_{q,\ell}$ is of the form $aX^{\ell} + bY^{\ell}$, for some $a, b, X, Y > 0$. The second derivative of this function is $aX^{\ell}\ln(X)^2 + bY^{\ell}\ln(Y)^2$. The positivity of a, b, X, Y implies that this is always non-negative, and thus $h_{q,\ell}$ is convex in ℓ; we obtain that $h_{q,\ell} \leq \max\{h_{q,1}, h_{q,q+1}\}$ for all $\ell \in [1, q+1]$.

It is straightforward to verify that $h_{q,1} = 1$. In the remainder we show that $h_{q,q+1} \leq 1$, which will complete the proof. The cases $q = 1, 2, 3$ are also verified immediately, so we can further assume that $q \geq 4$. To bound $h_{q,q+1}$ we will first prove the auxiliary fact

$$(1+y)^N \leq 1 + yN + y^2N^2/2, \text{ for all } N \in \mathbb{N}_0 \text{ and } y \leq N^{-2}. \tag{27}$$

To show (27), let us fix N and y as required. We will show by induction that $(1+y)^n \leq 1 + yn + y^2n^2/2$ for all $0 \leq n \leq N$. The case $n = 0$ is immediate. Moreover, for $0 \leq n \leq N-1$, by applying the induction hypothesis, we have

$$(1+y)^{n+1} \leq (1+y)(1+yn+y^2n^2/2) = 1 + y(n+1) + (n^2 + 2n + yn^2)y^2/2.$$

Together with $y \leq N^{-2}$ and $n \leq N$ this establishes (27).

Before we proceed with bounding $h_{q,q+1}$ we make two auxiliary observations. First, using the bound in (27) and that obviously $q(q+2) \geq q^2$ we readily obtain

$$\Big(1 + \frac{1}{q(q+2)}\Big)^{q+1} \leq \Big(1 + \frac{1}{q(q+2)}\Big)\Big(1 + \frac{1}{q+2} + \frac{1}{2(q+2)^2}\Big).$$

Second, using that $1 - x \leq e^{-x}$, we obtain

$$\Big(1 - \frac{1}{q+2}\Big)^{q+1} = \Big(1 - \frac{1}{q+2}\Big)^{q+2} \cdot \Big(1 + \frac{1}{q+1}\Big)^{-1} \leq \frac{1}{e} \cdot \Big(1 + \frac{1}{q+1}\Big).$$

By substituting both auxiliary observations in (26) and collecting terms we obtain that there are polynomials P, Q such that

$$h_{q,q+1} \leq \left(q\Big(1 + \frac{1}{q(q+2)}\Big)\Big(1 + \frac{1}{q+2} + \frac{1}{2(q+2)^2}\Big) + \frac{2q+3}{(q+1)e}\Big(1 + \frac{1}{q+1}\Big)\right)\frac{1}{q+2} = 1 - P(q)/Q(q),$$

where $Q(q) = 2e(q+2)^4(q+1)^2 > 0$ and $P(q) = \sum_{i=0}^{5} \alpha_i q^i$ is a polynomial of degree 5 such that

$$\alpha_5 = 2e - 4, \alpha_4 = 17e - 38, \alpha_3 = 56e - 144,$$

all > 0, and

$$\alpha_2 = 88e - 272, \alpha_1 = 66e - 256, \alpha_0 = 19e - 96,$$

all < 0. Since $\alpha_5 > 0$ the polynomial P is eventually positive, implying that $h_{q,q+1} \leq 1$ whenever q is sufficiently large. Moreover, let us write

$$P(q) = q^2(q^3\alpha_5 + \alpha_2) + q(q^3\alpha_4 + \alpha_1) + (q^3\alpha_3 + \alpha_0),$$

Note that (with plenty of room to spare)

$$4^3\alpha_i > -\alpha_{i-3}, \quad \text{for} \quad i \in \{3,4,5\}.$$

and thus $q^3\alpha_i + \alpha_{i-3} > 0$ for all $q \geq 4$ and $i \in \{3,4,5\}$. Thus $P(q) > 0$ for $q \geq 4$ and the proof is completed. □