Online Optimization with Uncertain Information*

Mohammad Mahdian
Yahoo! Research
mahdian@yahoo-inc.com

Hamid Nazerzadeh
Microsoft Research
hamidnz@microsoft.com

Amin Saberi
Stanford University
saberi@stanford.edu

December 28, 2009

Abstract

We introduce a new framework for designing online algorithms that can incorporate additional information about the input sequence, while maintaining a reasonable competitive ratio if the additional information is incorrect. Within this framework, we present online algorithms for several problems including allocation of online advertisement space, load balancing, and facility location.

1 Introduction

In many real-life optimization problems the input becomes available to the algorithm over time. The algorithm has to make a sequence of decisions in an "online" fashion, with zero or imperfect information about the future input. Online optimization is studied extensively [11, 1, 3]. In this literature, the prevalent way to evaluate the performance of an algorithm is *competitive analysis*. In competitive analysis, the performance of the algorithm is compared to the optimal offline solution, which knows the input sequence in advance. The worst-case ratio between the value obtained by the online algorithm and the optimum offline solution is called the competitive ratio of the algorithm.

In competitive analysis, one makes no assumptions about the input sequence and competes against an adversarial input. As a result, the algorithms with the best competitive ratio are often very conservative and not always useful in practice, see [16] for a discussion. One can try to make various assumptions about the distribution of the input or assume that the input follows a certain pattern. But learning the distribution or finding a pattern in real inputs is a challenging task. Even more importantly, in several applications, there are abrupt changes in the pattern of the input sequence. These changes may result in an extremely poor performance if the online algorithm does not adapt to the new input.

An illustrative example is the online optimization problem search engines such as Google, Yahoo! or Microsoft have to solve for allocating billions of dollars worth of of advertising space next to their search results [17]. If the frequencies of search queries are known in advance, one can solve the allocation problem optimally using linear programming. The main problem is that although the frequency estimates are often accurate, they might turn out to be completely wrong due to unexpected events. Examples include the sudden spike in the frequency of search for "gas mask" right after speculations about terrorist activities, or for drug "Vioxx" after concerns about increased

*An earlier version of this paper was presented in ACM Conference on Electronic Commerce [20].

risk of heart attack associated with its usage, or simply the surge in the volume of associated queries after the release of a new successful movie or record. Such spikes are usually very valuable for advertisers (and hence for the search engine), and at the same time they are essentially impossible to predict. As we will show later, an allocation algorithm that decides entirely based on the estimates might completely miss out on such revenue opportunities.

In this paper, we design online algorithms that can incorporate additional information about the input sequence and at the same time maintain a reasonable competitive ratio in unpredictable scenarios. We present these online algorithms for important optimization problems including load balancing, set cover, facility location, generalized Steiner tree, and allocation of online advertisement space.

Let us start with a more detailed description of the ad allocation problem motivated above. The objective is to find the optimal allocation of advertisement next to the search results for the search engine. Advertisers communicate their bids and total budgets to the search engine in advance. The search engine should decide how to allocate these advertisement spaces every time a user searches for a keyword.

Mehta et al. [21] first posed this problem as a generalization of the online matching problem. They gave an algorithm with the competitive ratio of $(1 - 1/e)$ and showed that this is the best possible ratio an online algorithm can achieve when there is no prior information about the future queries. Our goal is to use the critical information available to the search engine about the frequency of the keywords without a great sacrifice in the competitive ratio. We formalize this idea in the definition below. $V_A(I)$ denotes the value of the solution obtained by algorithm A, from instance I of the problem.

Definition 1 (Maximization problems)**.** *Consider two algorithms $\mathcal{P}$ and $\mathcal{O}$ for a maximization problem $\mathcal{I}$. We call an algorithm $\mathcal{H}$, (γ_p, γ_o)-balanced with respect to algorithms $\mathcal{P}$ and $\mathcal{O}$ if $V_{\mathcal{H}}(I) \geq \max\{\gamma_p \cdot V_{\mathcal{P}}(I), \gamma_o \cdot V_{\mathcal{O}}(I)\}$, for any instance I of the problem.*

Definition 2 (Minimization problems)**.** *Consider two algorithms $\mathcal{P}$ and $\mathcal{O}$ for a minimization problem $\mathcal{I}$. We call an algorithm $\mathcal{H}$, (γ_p, γ_o)-balanced with respect to algorithms $\mathcal{P}$ and $\mathcal{O}$ if $V_{\mathcal{H}}(I) \leq \min\{\gamma_p \cdot V_{\mathcal{P}}(I), \gamma_o \cdot V_{\mathcal{O}}(I)\}$, for any instance I of the problem.*

The above definitions can be applied with respect to any two algorithms $\mathcal{O}$ and $\mathcal{P}$, but we often think of $\mathcal{O}$ as an "optimistic" algorithm that assumes the input sequence arrives according to certain patterns, as predicted[1]. On the other hand, algorithm $\mathcal{P}$ is for the "pessimistic" scenario where we do not know anything about the future. It can be an online competitive algorithm (as in Sections 2 and 3), or, more directly, the optimum offline algorithm (as in Section 4).[2] This means that the algorithm $\mathcal{H}$ guarantees a value of $\gamma_o V_{\mathcal{O}}$ when the estimates of the input are accurate, while always obtaining a value of $\gamma_p V_{\mathcal{P}}$, even if the estimations turn out to be completely wrong.[3]

[1]Note that the algorithm $\mathcal{O}$ is assumed to work on *every* input. So, even if it is designed to behave optimally when the input sequence satisfies the predicted pattern, it should still respond to every input sequence. See, for example, Section 4.3 for a discussion of how unpredicted inputs are treated in the case of the query allocation problem.

[2]In the latter case, obviously, the algorithm $\mathcal{H}$ cannot see the output of $\mathcal{P}$, whereas in the former case where both $\mathcal{O}$ and $\mathcal{P}$ are online algorithms, we will design algorithms in Sections 2 and 3 that are balanced with respect to two algorithms $\mathcal{O}$ and $\mathcal{P}$ they have black-box access to.

[3]This means that when $\gamma_o V_{\mathcal{O}}$ is better than $V_{\mathcal{P}}$ (i.e., it is larger in the case of a maximization problem and smaller in the case of a minimization problem), $\mathcal{H}$ gives a better answer than $\mathcal{P}$. For most natural problems, when $\mathcal{O}$ is designed to be the optimal algorithm for a set of estimates, even when the estimates are *close* to the real input, the answer that $\mathcal{O}$ provides is considerably better than the answer provided by an algorithm $\mathcal{P}$. See, for example, the discussion in Section 4.3 on the query allocation problem.

In this paper, we design algorithms controlled by a parameter that give a tradeoff between the two factors γ_o and γ_p. We expect that as γ_o increases, γ_p decreases and vice versa.[4]

For the query allocation problem, for example, the online $(1 - 1/e)$-competitive algorithm of Mehta et al. can be used as $\mathcal{P}$, and $\mathcal{O}$ can be the linear program that computes the optimal allocation for given estimates of the frequencies of the queries, see Section 4.3. A simple algorithm for this problem, parameterized by a value $\gamma \in [0, 1]$ is as follows: Upon the arrival of a new query, allocate the query to the same advertiser as $\mathcal{O}$ if $\frac{1-\gamma}{\gamma}$ times the bid of this advertiser is greater than the bid of the advertiser chosen by $\mathcal{P}$. We analyze this algorithm in appendix B and prove that it is $(\gamma, 1-\gamma)$-balanced with respect to $\mathcal{P}$ and $\mathcal{O}$, i.e., it achieves a value at least $1-\gamma$ times the value of the optimal solution if the estimates are accurate, while guaranteeing a value of $\gamma \cdot (1 - 1/e)$ times the optimum offline solution on *any* input. We will also give a better algorithm in Section 4 that achieves a better factor with respect to the optimum offline solution directly. Our analysis for this algorithm is based on factor-revealing linear programs [14, 13].

Another problem we study is the online load balancing problem (see [6] for a survey). The problem is defined as follows. A sequence of n jobs arrives in an online fashion to be scheduled on m servers. The load of a job on each server is revealed as soon as the job arrives. The job should be allocated to one of the servers right away and it cannot be reassigned later. The goal is to minimize the maximum load of the servers.

If the load of each job on each server is available in advance, we can solve the problem using the algorithm of Lenstra et al. [19] for makespan minimization, which gives a 2-approximation of the optimal solution.[5] However, without prior information, no algorithm can achieve a competitive ratio better that $\Omega(\log n)$ [7]. Aspnes et al. [4] proposed a deterministic algorithm that obtains an $O(\log n)$-competitive ratio. We use these two algorithms as black boxes and obtain the following somewhat surprising result. It is possible to maintain a *constant* factor guarantee similar to Lenstra et al. when the estimates are accurate, while maintaining the $O(\log n)$ competitive ratio in the worst case.

We also apply our framework to the online version of a general class of resource allocation problems that includes set cover, facility location and generalized Steiner tree problem. We obtain guarantees similar to the load balancing problem, with a slightly different algorithm.

The rest of the paper is organized as follows. We discuss the load balancing problem in the next section. In section 3 we extend the results to a general class of resource allocation problems. Finally, the query allocation problem and our solution for it are presented in section 4.

2 Online Load Balancing

The online load balancing problem is defined as follows. A sequence of n jobs arrives in an online fashion to be scheduled on m servers. The job should be allocated to one of the servers immediately after its arrival and it cannot be reassigned later. The load of job j on each server i, denoted by l_{ij}, is revealed as soon as the job arrives. The load on a server is defined as the sum of the loads

[4] Another interpretation is to think of each of the algorithm $\mathcal{P}$ and $\mathcal{O}$ as an expert and the goal is to choose the best of the two. There exists an extensive literature on expert algorithms (e.g., see [15]). However, in the problems we consider the cost or reward functions may significantly change by the actions taken in the previous steps (for example due to budget constraints in the query allocation problem). Hence, we need to develop new techniques for these problems.

[5] It is easy to see that if the estimates of loads of the jobs is off by a constant factor, the algorithm is still constant competitive.

of all jobs that are assigned to that server. The goal is to minimize the maximum load of the servers. For instance, consider the problem of bandwidth allocation in communication channels. In this problem, communication requests (jobs) arrive online and should be immediately allocated to channels (servers). The load of a request is the percentage of the used bandwidth.

Suppose we have some estimates about the loads of the jobs. One naive approach is to compute the optimal allocation with respect to these estimates and then assign jobs according to this allocation. Not surprisingly, if the estimates are wrong, this approach can incur an arbitrary high cost compared to the optimal solution. This is shown in the following example.

Example The example consists of two servers and two jobs. The following estimates are available about the loads of the jobs: $l_{11} = 10$, $l_{21} = 1$, $l_{12} = 100$, and $l_{22} = 10$. Based on these estimates, the first job should be assigned to the first server and the second job to the second server. Suppose the load of the first job on each server matches the estimations. However, once the second job arrives, we observe that its load on each server is 1. The cost of the solution based on (inaccurate) estimates is at least 10. However, the optimal cost is 1.

The example above shows that the worst-case competitive ratio of the naive approach is zero. We use the term *worst-case competitive ratio* to emphasize that this is the minimum competitive ratio over *all* possible input sequences. In fact, it is easy to see from the example above that there is no algorithm that achieves the optimal solution when the estimates are accurate *and* has a bounded worst-case competitive ratio. A natural question is, if we allow a small increase in the cost in cases where our estimates are accurate, would we be able to achieve a bounded worst-case competitive ratio?

We give a positive answer to this question. Assume that we have access to two algorithms $\mathcal{P}$ and $\mathcal{O}$. Upon the arrival of every new job, each algorithm recommends a server to receive the job. For any $\gamma \geq 1$, we present an algorithm, denoted by $\mathcal{L}(\gamma)$, that is $(\gamma, \frac{\gamma}{\gamma-1})$-balanced with respect to $\mathcal{P}$ and $\mathcal{O}$.

As mentioned before, if the load of each job on each server is available in advance, we can use the algorithm of Lenstra et al. [19] for makespan minimization which is a factor 2-approximation algorithm. On the other hand, the algorithm of Aspnes et al. [4] is $\Theta(\log n)$-competitive in worst-case. Observe that using these algorithms as $\mathcal{O}$ and $\mathcal{P}$, our algorithm achieves a constant competitive ratio if the estimates are accurate, while the worst-case competitive ratio remains optimal (up to a constant factor).

The choice of γ is left to the user who can adjust it based on the reliability of the estimates. Algorithm $\mathcal{L}(1)$ gives the same assignment as $\mathcal{P}$. As γ increases, the algorithm follows the recommendations of $\mathcal{O}$ on more jobs. For example, for $\gamma = 2$, the cost of the solution found by $\mathcal{L}(2)$ is at most twice of the cost of the better algorithm between $\mathcal{P}$ and $\mathcal{O}$. Also, $\gamma = \infty$ means that $\mathcal{L}$ always follows $\mathcal{O}$.

Our algorithm is based on the following simple rule: Assign the job to the server recommended by $\mathcal{O}$ if the total cost of following recommendations of $\mathcal{O}$ so far is bounded by a constant multiple of the cost of following recommendations of $\mathcal{P}$ so far. In this way, the algorithm balances the cost of $\mathcal{O}$ with respect to $\mathcal{P}$. To explain the algorithm formally, we need some definitions. Assume that the job arriving at time t is denoted by job t. Define $w_t(\mathcal{L}(\gamma))$ to be the maximum load of the servers up to time t. Also, define $w_t(\mathcal{P})$ to be the maximum load of the servers if one follows all the recommendations of $\mathcal{P}$ up to time t. Similarly, $w_t(\mathcal{O})$ is defined as the maximum load of the servers achieved by following all the recommendations of $\mathcal{O}$. The algorithm is presented in Figure 1.

Algorithm $\mathcal{L}(\gamma)$:

Upon the arrival of a new client j:

If $w_j(\mathcal{O}) \leq (\gamma - 1)w_j(\mathcal{P})$,

Assign j to the server recommended by $\mathcal{O}$.

else

Assign j to the server recommended by $\mathcal{P}$.

Figure 1: A $(\gamma, \frac{\gamma}{\gamma-1})$-balanced algorithm for online load balancing.

Lemma 1. *For every time t, we have $w_t(\mathcal{L}(\gamma)) \leq \gamma w_t(\mathcal{P})$.*

Proof. If the algorithm has followed all recommendations of $\mathcal{P}$ so far, then the claim trivially holds. Otherwise, let k be the last job that the algorithm allocated to the server recommended by $\mathcal{O}$ instead of $\mathcal{P}$. By the rule of the algorithm we have

$$w_k(\mathcal{O}) \leq (\gamma - 1)w_k(\mathcal{P}) \leq (\gamma - 1)w_t(\mathcal{P}) \tag{1}$$

The load on every server is due to the jobs that are recommended either by $\mathcal{P}$ or $\mathcal{O}$. Therefore, $w_t(\mathcal{L}(\gamma)) \leq w_k(\mathcal{O}) + w_t(\mathcal{P})$. Therefore, by plugging (1), we get:

$$w_t(\mathcal{L}(\gamma)) \leq w_k(\mathcal{O}) + w_t(\mathcal{P}) \leq (\gamma - 1)w_t(\mathcal{P}) + w_t(\mathcal{P}) = \gamma w_t(\mathcal{P})$$

□

Symmetrically, we have the following lemma.

Lemma 2. *For every time t, we have $w_t(\mathcal{L}(\gamma)) \leq \frac{\gamma}{\gamma-1} w_t(\mathcal{O})$.*

Proof. If the algorithm has followed all recommendations of $\mathcal{O}$ so far, then the claim trivially holds. Otherwise, let k be the last job that the algorithm ignores the recommendation of $\mathcal{O}$. We have,

$$(\gamma - 1)w_k(\mathcal{P}) < w_k(\mathcal{O}) \leq w_t(\mathcal{O})$$

Hence, for any server i, the total load of the jobs assigned to i that are not recommended by $\mathcal{O}$ is bounded by $\frac{1}{\gamma-1} w_t(\mathcal{O})$. Also, the total load of the jobs that are recommended by $\mathcal{O}$ is at most $w_t(\mathcal{O})$. Hence,

$$w_t(\mathcal{L}(\gamma)) \leq (1 + \frac{1}{\gamma - 1})w_t(\mathcal{O}) \leq \frac{\gamma}{\gamma - 1} w_t(\mathcal{O})$$

□

Now we are ready to prove the main result of this section.

Theorem 3. *For $\gamma \geq 1$, algorithm $\mathcal{L}(\gamma)$ is $(\gamma, \frac{\gamma}{\gamma-1})$-balanced with respect to $\mathcal{P}$ and $\mathcal{O}$. In addition, suppose algorithm $\mathcal{O}$ makes its recommendation based on an optimal allocation for a given estimate. There exists an algorithm $\mathcal{P}$ such that for any deterministic (γ, γ')-balanced algorithm with respect to $\mathcal{P}$ and $\mathcal{O}$, we have $\gamma' \geq \frac{\gamma}{\gamma-1}$.*

Proof. The lemmas above immediately show that the algorithm is $(\gamma, \frac{\gamma}{\gamma-1})$-balanced. We prove the second part of the theorem using the example below. The example consists of two jobs and two servers. The following estimates are available: $l_{11} = \gamma - 1$, $l_{21} = 1$, $l_{12} = \infty$ and $l_{22} = \gamma - 1$. Based on these estimates, $\mathcal{O}$ assigns the first job to the first server and the second job to the second server. When the first job arrives, its loads matches the estimates. However, $\mathcal{P}$ recommends the second server for the job. Consider the following cases for an algorithm $\mathcal{A}$.

1. $\mathcal{A}$ follows the recommendation of $\mathcal{O}$ and allocates the first job to the first server. However, when the second job arrives, we observe $l_{12} = 1$ and $l_{22} = \infty$. In this case, the cost of $\mathcal{A}$ is at least γ, while the cost of $\mathcal{P}$ is 1.

2. $\mathcal{A}$ ignores the recommendation of $\mathcal{O}$ and allocates the first job to the second server. Then, the second job arrives, and its loads matches the estimates. In this case, the cost of $\mathcal{A}$ is at least γ while the cost of the solution recommended by $\mathcal{O}$ is $\gamma - 1$.

Hence, the claim follows. □

3 Online Resource Allocation

In this section, we present algorithms for the following problems.

Online Set Cover A ground set $\mathcal{J}$ and a family of m of its subsets are given. A sequence of n elements of $\mathcal{J}$ arrives in an online fashion. Each new element, upon arrival, should be covered by one of the given subsets. The goal is to cover all elements appeared in the input sequence with the minimum number of subsets. For this problem, Alon et al. [2] gave an $O(\log n \log m)$-competitive algorithm, and showed that no deterministic algorithm can achieve a worst-case competitive ratio better than $\Omega(\frac{\log n \log m}{\log \log n + \log \log m})$. On the other hand, when accurate estimates of the input sequence is available, a simple greedy algorithm achieves an $O(\log n)$-approximation of the optimal solution, see [22].

Online Facility Location A set of m potential locations for building facilities, and a cost for opening a facility in each of these locations are given. A sequence of clients arrives in an online fashion. Once a new client arrives, the algorithm should take one of the following actions: i) Open a facility in one of the m locations and connect the client to it. ii) Connect the client to one of the open facilities. The goal is to find an assignment of clients to opened facilities that minimizes the cost of opening facilities plus the connection cost (given by the distances between the client and the facility). This problem generalizes the online set cover problem. However, when the connection costs are metric, better competitive ratios are achievable. Fotakis [12] showed that the worst-case competitive ratio of the online metric facility location problem is $\Theta(\frac{\log n}{\log \log n})$. On the flip side, if the sequence of clients is known in advance, one can use the algorithm of Byrka [10] which is a 1.5-approximation algorithm.

Online Generalized Steiner Tree We are given a graph with non-negative weights. A sequence of pairs of vertices arrives online. The goal is to construct a subgraph with minimum weight such that the two nodes of each pair are connected by a path in the subgraph. For this problem, Berman and Coulston [8] proposed a $O(\log n)$-competitive algorithm, which matches the lower bound of $\Omega(\log n)$ on the competitiveness of any online algorithm [5]. However, with accurate estimates of the input sequence, one can solve the problem within a constant factor of the optimal solution, see [22].

All the problems above are special cases of the online resource allocation problem defined below:

Definition 3 (Online Resource Allocation (ORA)). *A set $\mathcal{S}$ of servers and a ground set $\mathcal{J}$ of jobs are given. A sequence of jobs in $\mathcal{J}$ arrives in an online fashion. As a job j arrives, it should be immediately assigned to a* set *of servers. Let S_j denote the family of subsets of $\mathcal{S}$ that job j can be assigned to. The cost of assigning job j to a set $s \in S_j$ is given by function $d_j : S_j \to R^+$. Also, the first time that any job is assigned to a server $i \in \mathcal{S}$, an activation cost of $c_i \geq 0$ is incurred. The goal is to find an assignment of jobs to active servers that minimizes the cost of activating the servers plus the serving cost. We assume decisions taken by the algorithm are irrevocable, i.e., the algorithm cannot reassign a job to another set of servers or deactivate a server.*

In the set cover problem, elements in $\mathcal{J}$ correspond to jobs and subsets correspond to servers. An element (job) j can be assigned to any subset (server) i that contains j, i.e., $S_j = \{s : j \in s\}$. Activating cost c_i, $1 \leq i \leq m$ is equal to 1 for all servers and there is no serving cost. Also, observe that the facility location problem is a special case of ORA where each $S_j = \{\{1\}, \ldots, \{m\}\}$, i.e., each job (client) can be served by any single facility. In the generalized Steiner tree problem, each pair (a, b) of vertices corresponds to a job, and each edge corresponds to a server. Set $S_{(a,b)}$ is the set of all paths in graph that connect a to b. There is no serving cost.

To apply our framework we assume that we have access to two algorithms $\mathcal{P}$ and $\mathcal{O}$ which upon the arrival of every new job j, recommends a subset from S_j to serve j. Similar to the load balancing problem, we present a family of algorithms, denoted by $\mathcal{H}(\gamma)$, $\gamma \geq 1$, that are $(\gamma, \frac{\gamma}{\gamma-1})$-balanced.

As mentioned, for the special cases of the ORA problem, accurate estimates can improve the performance of algorithms by a logarithmic factor. Our algorithm can still enjoy this significant improvement, while at the same time maintaining an optimal (up to a constant) worst-case competitive ratio.

Now let us describe the algorithm. Let job t denote the job that arrives at time t. Let $w_t(\mathcal{H}(\gamma))$ be the total cost of algorithm $\mathcal{H}(\gamma)$ up to time t, i.e.,

$$w_t(\mathcal{H}(\gamma)) = \sum_{i:i \text{ is active}} c_i + \sum_{j=1}^{t} d_j(s_j)$$

where $s_j \in S_j$ denotes the set that the algorithm assigns to job t. Similarly, let $w_t(\mathcal{P})$ ($w_t(\mathcal{O})$, respectively) be the cost of the solution one obtains by following all the recommendations of $\mathcal{P}$ ($\mathcal{O}$, respectively). The algorithm is essentially the same as the one used in the previous section, and is based on the following simple rule: Assign the job to the set recommended by $\mathcal{O}$ if $w_t(\mathcal{O}) \leq (\gamma - 1)w_t(\mathcal{P})$. The algorithm is presented in Figure 2.

Theorem 4. *Algorithm $\mathcal{H}(\gamma)$, for $\gamma \geq 1$, is $(\gamma, \frac{\gamma}{\gamma-1})$-balanced with respect to algorithms $\mathcal{P}$ and $\mathcal{O}$. In addition, suppose algorithm $\mathcal{O}$ makes its recommendation based on an optimal allocation for a given estimate. There exists an algorithm $\mathcal{P}$ such that for any deterministic (γ, γ')-balanced algorithm with respect to $\mathcal{P}$ and $\mathcal{O}$, we have $\gamma' \geq \frac{\gamma}{\gamma-1}$.*

Algorithm $\mathcal{H}(\gamma)$:

Upon the arrival of a new job j:

If $w_j(\mathcal{O}) \leq (\gamma - 1)w_j(\mathcal{P})$

Let o be the set of servers recommended by $\mathcal{O}$ to serve j.

Activate all inactive servers in o.

Assign j to servers in o.

else

Let p be the set of servers recommended by $\mathcal{P}$ to serve j.

Activate all inactive servers in p.

Assign j to servers in p.

Figure 2: A $(\gamma, \frac{\gamma}{\gamma-1})$-balanced algorithm for the online resource allocation problem.

The proof of the theorem is similar to Theorem 3 and is given in appendix A.

4 The Query Allocation Problem

We start this section by defining the query allocation problem first posed by Mehta et al. [21] as a generalization of the online matching problem. In this model, the search engine receives the bids of advertisers for each keyword and their total budget for a certain period (e.g. a day). We make the following simplifying assumption. During the day, as a search query for a keyword arrives, the search engine assigns one advertiser to this query, and charges this advertiser an amount equal to their bid for the corresponding keyword.

Let $\mathcal{A}$ be the set of advertisers and $\mathcal{K}$ be the set of keywords. A sequence of queries of keywords in $\mathcal{K}$ are given to the algorithm in an online fashion. We denote the budget of advertiser i by B_i, and the bid of advertiser i for keyword j by b_{ij}. As in [21], we assume that bids are *small* compared to the budgets, which is justified in practice. The goal is to maximize the revenue, i.e., the sum of $b_{i(q),j(q)}$ over all queries q, where $i(q)$ denotes the advertiser assigned to this query, and $j(q)$ denotes the keyword corresponding to this query, subject to charging each advertiser at most its budget. Therefore, if we know the number of times n_j a query for keyword j occurs in the input sequence, the solution of this problem can be computed by solving the following maximization program with integrality constraints on x_{ij}'s:

$$
\begin{array}{lll}
\text{maximize} & \sum_{i\in\mathcal{A},j\in\mathcal{K}} b_{ij}x_{ij} & \\
\text{subject to} & \sum_{i\in\mathcal{A}} x_{ij} \leq n_j & \forall\ j \in \mathcal{K} \\
& \sum_{j\in\mathcal{K}} b_{ij}x_{ij} \leq B_i & \forall\ i \in \mathcal{A} \\
& x_{ij} \geq 0 & \forall\ i \in \mathcal{A}, \forall\ j \in \mathcal{K}
\end{array}
\tag{2}
$$

Note that the assumption that the bids are small compared to the budgets implies that relaxing the integrality constraints in the above program does not change the solution of the program by a significant amount.

When frequencies n_j are unknown, Mehta et al. [21] give a $(1-1/e)$-competitive algorithm for this problem, and show that this is the best ratio an online algorithm can achieve.

Similar to the previous sections, in appendix B, we present a family of algorithms parameterized by $\gamma \in [0,1]$ that are $(\gamma, 1-\gamma)$-balanced with respect to two given online algorithms $\mathcal{P}$ and $\mathcal{O}$. This means that if $\mathcal{O}$ is taken to be the algorithm that assigns the queries optimally for given estimates of query volumes, and $\mathcal{P}$ is the algorithm of Mehta et al., then we have an algorithm that achieves at least a fraction $1-\gamma$ of the optimal revenue when estimates are accurate, and a fraction $\gamma\cdot(1-1/e)$ when the estimates are wrong. In this section, we use techniques from the competitive analysis of Mehta et al. to give an algorithm that achieves a better factor compared to the optimal solution when the estimates are wrong. We take $\mathcal{P}$ to be the optimal offline solution, i.e., we compare our algorithm directly against the optimal offline and not against an online competitive algorithm. Obviously, the algorithm does not see the output of $\mathcal{P}$ (in contrast to previous sections where we assumed access to $\mathcal{O}$ and $\mathcal{P}$). As before, we assume the algorithm has access to an algorithm $\mathcal{O}$ that recommends an advertiser for each query. $\mathcal{O}$ can be based on an algorithm that solves the linear program above for given estimates n_j of the frequencies (if such estimates are available), or even a more complex algorithm that learns the distribution of the queries and the corresponding optimal allocation over time.

We first briefly present the $(1-1/e)$-competitive algorithm of Mehta et al. At any point during the algorithm, the discounted bid of an advertiser is defined by multiplying the bid by a factor of $(1-e^{f-1})$ where f is the fraction of the budget of the advertiser that is spent. This means that the more an advertiser spends her budget, the more the algorithm discounts her bids. As a new query arrives, the algorithm assigns it to the advertiser with the maximum discounted bid.

Our algorithm is parameterized by a number $\alpha \geq 1$. This parameter controls the extent to which one would rely on $\mathcal{O}$. For $f \in [0,1]$, define $\Phi_\alpha(f) := 1 - e^{\alpha\cdot(f-1)}$. For ease of notation, when it is clear from the context, we denote Φ_α by Φ. Also, let f_i be the fraction of the budget of advertiser i that has been spent so far. As a new query for keyword j arrives, the algorithm finds advertiser p that maximizes $\Phi(f_i)b_{ij}$ over all $i \in \mathcal{A}$. Also, $\mathcal{O}$ recommends advertiser o to receive the query. The algorithm compares $\alpha\Phi(f_o)b_{oj}$ with $\Phi(f_p)b_{pj}$. If the former is bigger than the latter, then the query is allocated to o; otherwise, it is allocated to p. We denote the algorithm corresponding to a given α by $\mathcal{Q}(\alpha)$. The algorithm is presented in Figure 3. We assume the search engine charges advertisers the minimum of their bid and their remaining budget.

The intuition behind the algorithm follows from a primal-dual interpretation of the algorithm of Mehta et al. [21], due to Buchbinder et al. [9]. The following is the dual of the linear program

Algorithm $\mathcal{Q}(\alpha)$:

Upon the arrival of a new query for keyword j:

Let o be the advertiser recommended by $\mathcal{O}$ for receiving j.

Let p be the advertiser with maximum $\Phi_\alpha(f_i)b_{ij}$ among all $i \in \mathcal{A}$.

If $\alpha\Phi_\alpha(f_o)b_{oj} \geq \Phi_\alpha(f_p)b_{pj}$ then

Allocate j to o.

else

Allocate j to p.

Figure 3: An algorithm for the query allocation problem.

for the query allocation problem. Without loss of generality, we assume that all n_j's are equal to 1.

$$\begin{aligned} \text{minimize} \quad & \sum_{j\in\mathcal{K}} \lambda_j + \sum_{i\in\mathcal{A}} \beta_j B_i & & (3) \\ \text{subject to} \quad & \lambda_j \geq (1-\beta_i)b_{ij} & \forall\ j \in \mathcal{K}, \forall\ i \in \mathcal{A} \\ & \lambda_j, \beta_i \geq 0 & \forall\ j \in \mathcal{K}, \forall\ i \in \mathcal{A} \end{aligned}$$

The algorithm of Mehta et al., as interpreted by Buchbinder et al. [9], allocates query j to advertiser i that maximizes $(1-\beta_i)b_{ij}$. After the allocation, β_i is updated multiplicatively such that $1-\beta_i$ is proportional to $1-e^{f_i-1}$. In our algorithm, in order to incorporate the recommendations of $\mathcal{O}$, the feasible region of the dual linear program is extended by relaxing the dual constraint by a factor of α. Formally, upon arrival of a new query j, $\mathcal{Q}(\alpha)$ allocates j to advertiser o, recommended by $\mathcal{O}$, if $\alpha\Phi_\alpha(f_o)b_{oj} \geq \max_i\{\Phi_\alpha(f_i)b_{ij}\}$. After the allocation of the query to advertiser i, dual variable β_i is updated by letting: $\beta_i \leftarrow \beta_i(1+\alpha\frac{b_{ij}}{B_i}) + \frac{\alpha b_{ij}}{(e^\alpha-1)B_i}$, see also [9].

In the next subsection we show that the algorithm above is $\frac{1}{\alpha}(1-\frac{1}{e^\alpha})$-competitive with respect to the optimal offline solution, independent of the accuracy of the estimations. In subsection 4.2, we show that the algorithm would achieve a better competitive ratio if the estimations are accurate. The theorem below summarizes these results. Its proof follows from Lemmas 7, 13 and 15, presented in the next subsections.

Theorem 5. *Algorithm $\mathcal{Q}(\alpha)$ is $(\gamma_{\text{OPT}}, \gamma_{\mathcal{O}})$-balanced with respect to the optimal offline solution and $\mathcal{O}$, where $\gamma_{\text{OPT}} = \frac{1}{\alpha}(1-e^{-\alpha})$ and $\gamma_{\mathcal{O}}$ is defined as follows. Let $\alpha^* \approx 1.795$ be the root of the equation $(\alpha^2+\alpha+1)e^{-\alpha} = 1$. For $\alpha \geq \alpha^*$,*

$$\gamma_{\mathcal{O}} = \frac{\alpha(1-e^{-\alpha})}{(\alpha-\frac{1}{\alpha})(1-e^{-\alpha})+1}, \tag{4}$$

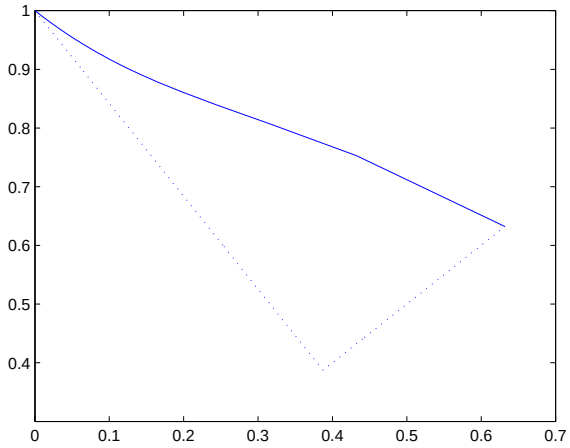

Figure 4: The trade-off between γ_{OPT} (x-axis) and $\gamma_{\mathcal{O}}$ (y-axis) for the query allocation problem. γ_{OPT} is the worst-case competitive ratio of algorithm $\mathcal{Q}(\alpha)$, while $\gamma_{\mathcal{O}}$ is the competitive ratio when the estimates are accurate. The dashed line depicts the competitive ratio for the algorithm $\mathcal{G}$ in Appendix B.

and for α *in* $(1, \alpha^*)$,

$$\gamma_{\mathcal{O}} = 1 - \frac{(1 - \frac{1}{\alpha}(1 - e^{-\alpha}))(1 - f^*)}{1 - \frac{1}{\alpha}(1 - e^{-\alpha}) + \alpha(1 - f^*)(1 - e^{-\alpha(f^*-1)})} \tag{5}$$

where $f^* \in [0, 1]$ *is the solution of the equation* $(\alpha(f-1))^2 e^{\alpha(f-1)} = 1 - \frac{1}{\alpha}(1 - e^{-\alpha})$. *Moreover, the analysis is tight.*

Similar to the previous sections, the above theorem gives a family of $(\gamma_{\mathrm{OPT}}, \gamma_{\mathcal{O}})$-balanced algorithms parameterized by α. Figure 4 depicts the value of $\gamma_{\mathcal{O}}$ against γ_{OPT}. Note that the maximum possible value for γ_{OPT} is $1 - \frac{1}{e}$. The dashed line depicts a lower bound on the competitive ratio for the algorithm $\mathcal{G}$ given in Appendix B with $\mathcal{P}$ taken as the algorithm of Mehta et al. This algorithm is $(\gamma, 1-\gamma)$-balanced with respect to $\mathcal{P}$ and $\mathcal{O}$. Since the algorithm of Mehta et al is $(1-\frac{1}{e})$-competitive with respect to the optimal offline solution (and hence $\mathcal{O}$), $\mathcal{G}$ is $(\gamma \cdot (1-1/e), \max\{\gamma \cdot (1-1/e), 1-\gamma\})$-balanced with respect to the offline optimal solution and $\mathcal{O}$.

4.1 Computing γ_{OPT}

In this section, we find γ_{OPT}, the competitive ratio of algorithm $\mathcal{Q}(\alpha)$ with respect to the optimal offline solution. The analysis is similar to [21] and we try to be consistent with their notation. Let k be a large number; we discretize the budgets of the advertisers into k equal parts called *slabs*, numbered 1 through k. For advertiser i, let slab(i) be equal to $\lceil k f_i \rceil$ if $f_i > 0$ and 1 if $f_i = 0$, i.e., slab(i) denotes the active slab of the budget of i. Also, let

$$\phi(s) \quad = \quad 1 - (1 - \frac{1}{k})^{\alpha(k-s)} \tag{6}$$

We define $\mathcal{Q}_k(\alpha)$ as a discrete version of algorithm $\mathcal{Q}(\alpha)$ in which the comparison is between $\alpha\phi_\alpha(\text{slab}(o))b_{oj}$ and $\phi_\alpha(\text{slab}(p))b_{pj}$ (instead of $\alpha\Phi_\alpha(f_o)b_{oj}$ and $\Phi_\alpha(f_p)b_{pj}$). Note that as k tends to infinity we can approximate the function $\phi(s)$ with $\Phi(\frac{s}{k})$ with arbitrary high precision. Therefore, as k tends to infinity, $\mathcal{Q}_k(\alpha)$ and $\mathcal{Q}(\alpha)$ have the same performance.

Now we analyze $\mathcal{Q}_k(\alpha)$. We call an advertiser of type t, $1 \le t \le k$, if *at the end of the algorithm* $\mathcal{Q}_k(\alpha)$ she has spent within $(\frac{t-1}{k}, \frac{t}{k}]$ fraction of her budget. The type of advertisers with no spent budget is defined to be 1. Let s_t be the total budget of the advertisers of type t. Also, fix an optimal offline solution OPT, and let r_t to be the total unspent budget of advertisers of type t in OPT. As in [21], for simplicity, we assume that advertisers of type t spend exactly a $\frac{t}{k}$ fraction of their budgets in $\mathcal{Q}_k(\alpha)$. Also, we assume that no query straddles a slab. If $\max\{\frac{b_{ij}}{B_i}\} \le \frac{1}{k^2}$, then total error due to these assumptions is $O(\frac{|\mathcal{A}|}{k})$ which is negligible as k tends to infinity. For $i = 0, 1, \ldots, k$, define w_i to be the total amount of money spent from slab i of the budget of the advertisers. Note that $w_i = \frac{1}{k}\sum_{j=i}^{k} s_j$. Also, observe that by the definition of s_t and r_t, the revenue of OPT is precisely $\sum_{i=1}^{k}(s_i - r_i)$, and the revenue of $\mathcal{Q}_k(\alpha)$ is equal to $\sum_{i=1}^{k}\frac{i}{k}s_i$. We relate these two quantities via the following lemmas.

Lemma 6. *We have:*

$$\sum_{i=1}^{k}\phi(i)(s_i - r_i) \le \sum_{i=1}^{k}\alpha\phi(i)w_i \tag{7}$$

Proof. Suppose OPT allocates query q to advertiser i and the algorithm allocates it to advertiser a. Let $g_i = \text{slab}(i)$ and $g_a = \text{slab}(a)$, at the time of allocation of q. By the rule of the algorithm we have

$$\phi(g_i)b_{iq} \le \alpha\phi(g_a)b_{aq}$$

Let g_i' be the type of advertiser i at the end of the algorithm. Because ϕ is decreasing, we get

$$\phi(g_i')b_{iq} \le \phi(g_i)b_{iq} \le \alpha\phi(g_a)b_{aq} \tag{8}$$

Note that the sum of $\phi(g_i')b_{iq}$ over all advertiser-query pairs (i, q) such that OPT allocated q to i is equal to $\sum_{i=1}^{k}\phi(i)(s_i - r_i)$. Similarly, the sum of $\phi(g_a)b_{aq}$ over all advertiser-query pairs (a, q) such that $\mathcal{Q}(\alpha)$ allocated q to a is equal to $\sum_{i=1}^{k}\phi(i)w_i$. By Inequality (8) we have

$$\sum_{i=1}^{k}\phi(i)(s_i - r_i) \le \sum_{i=1}^{k}\alpha\phi(i)w_i$$

□

Now, we are ready to prove the main result of this section.

Lemma 7. *As k tends to infinity, the competitive ratio of $\mathcal{Q}_k(\alpha)$ with respect to the optimum offline solution, γ_{OPT}, converges to $\frac{1}{\alpha}(1 - e^{-\alpha})$.*

Proof. Plugging $w_i = \frac{1}{k}\sum_{j=i}^{k} s_j$ into (7), we get

$$\sum_{i=1}^{k}\phi(i)(s_i - r_i) \le \alpha\sum_{j=1}^{k}\phi(j)(\frac{1}{k}\sum_{i=j}^{k}s_i) = \frac{\alpha}{k}\sum_{i=1}^{k}s_i\sum_{j=1}^{i}\phi(i) \tag{9}$$

In Lemma 23 in the appendix, we prove that as k tends to infinity we can approximate $\frac{1}{k}\sum_{t=1}^{i}\phi(t)$ by $\frac{i}{k}+\frac{1}{\alpha}(\phi(i)-\phi(0))+o(1)$. Plugging into (9) we have

$$\sum_{i=1}^{k}\phi(i)(s_i-r_i)\leq\sum_{i=1}^{k}(\alpha\frac{i}{k}+\phi(i)-\phi(0)+o(1))s_i$$

Since $\sum_{i=1}^{k}s_i$ is the total budget and does not increase with k, we have $o(1)\sum_{i=1}^{k}s_i=o(1)$. Rearranging the terms gives us

$$\sum_{i=1}^{k}\phi(0)s_i-\sum_{i=1}^{k}\phi(i)r_i\leq\alpha\sum_{i=1}^{k}\frac{i}{k}s_i+o(1).$$

Since ϕ is decreasing, we obtain

$$\phi(0)(\sum_{i=1}^{k}s_i-r_i)\leq\alpha\sum_{i=1}^{k}\frac{i}{k}s_i+o(1).$$

Note that $\sum_{i=1}^{k}(s_i-r_i)$ is the revenue of OPT and $\sum_{i=1}^{k}\frac{i}{k}s_i$ is the revenue of $\mathcal{Q}_k(\alpha)$. Hence, the lemma follows. □

Tight Example Consider a large integer k. Suppose there are k advertisers each with budget $\frac{\alpha}{k}$. A sequence of keywords shows up in k phases. In phase i, $\frac{1}{k\epsilon}$ keywords arrive for which advertiser $1,\ldots,i-1$ bid 0, advertiser i bids slightly less than $\alpha\epsilon$, and advertisers $i+1,\ldots,k$ bid ϵ. The algorithm, based on recommendations of $\mathcal{O}$, distributes these keywords among advertisers $i,\ldots,k$, such that the fraction of the spent budget of these advertisers remains equal. Define ℓ to be the phase in which the budget of advertiser k is exhausted. In phase $i\leq\ell$, the total value of the queries allocated to advertiser i is negligible. Therefore, we assume in phase i, an amount of $\frac{1}{k-i+1}\times\frac{1}{\alpha k}$ of the budget of every advertiser $j\geq i$ is spent. Hence, at the end of phase $i\leq\ell$, an amount of $\sum_{j=1}^{i}\frac{1}{k-j+1}\frac{1}{k}$ of the budget of each advertiser $j>i$ is spent. As $k\to\infty$, we can approximate $\sum_{j=1}^{i}\frac{1}{k-j+1}\frac{1}{k}$ with $\frac{1}{k}\int_0^{1-(i/k)}\frac{1}{1-x}dx=\frac{-1}{k}\ln(1-\frac{i}{k})$ with arbitrary high accuracy. Therefore, by definition of ℓ, we have $\frac{1}{k}\ln\frac{1}{1-(\ell/k)}=\frac{\alpha}{k}$ or $\frac{\ell}{k}=1-e^{-\alpha}$. Also, the revenue of algorithm $\mathcal{Q}$ is equal to $\ell\cdot\frac{1}{k}=1-e^{-\alpha}$ (observe that the revenue at each phase $i\leq\ell$ is $\frac{1}{k}$, and for $i>\ell$ is 0). The optimal solution spends the budget of all advertisers and obtains a revenue of α. Therefore, the approximation ratio is equal to $\frac{1}{\alpha}(1-e^{-\alpha})$.

4.2 Computing $\gamma_{\mathcal{O}}$

In this section, we compute $\gamma_{\mathcal{O}}$, the ratio of the revenue of our algorithm to the revenue that could be obtained by following all the recommendation of $\mathcal{O}$. We use the same notations and assumptions as the previous section except for r_i. Define r_i to be the total remaining budget of advertisers of type i in the allocation recommended by $\mathcal{O}$.

Let C be the set of all queries for which the algorithm does not follow the recommendation of $\mathcal{O}$. We find an upper bound on the value of the queries in C. For $j\leq i$, define t_{ij} to be the total amount of budget that advertisers of type i have spent from interval $(\frac{j-1}{k},\frac{j}{k}]$ fraction of their budget for the keywords in $\mathcal{Q}-C$, i.e., the set of keywords for which the algorithm followed the recommendation. For simplicity, we define $t_{ij}=0$ for $j>i$.

Lemma 8. *For algorithm $\mathcal{Q}_k(\alpha)$, when k tends to infinity, we have*

$$\sum_{i=1}^{k}\sum_{j=1}^{i}(\phi(j)-\alpha\phi(i))t_{ij} \le \sum_{i=1}^{k}(\frac{i}{k}-\frac{1}{\alpha}\phi(0)+(\frac{1}{\alpha}-\alpha)\phi(i))s_i+\alpha\sum_{i=1}^{k}\phi(i)r_i+o(1).$$

Proof. The proof of this lemma is similar to Lemma 6. Suppose $\mathcal{O}$ allocates query $q \in C$ to advertiser i and the algorithm allocates it to advertiser a. Let $g_i = \text{slab}(i)$ and $g_a = \text{slab}(a)$, at the time of allocation of q. Recall that for each keyword q by the rule of the algorithm we have

$$\phi(g_i)b_{iq} \le \frac{1}{\alpha}\phi(g_a)b_{aq}$$

Let g_i' be the type of advertiser i at the end of the algorithm. Because ϕ is decreasing, we get

$$\phi(g_i')b_{iq} \le \phi(g_i)b_{iq} \le \frac{1}{\alpha}\phi(g_a)b_{aq} \tag{10}$$

Also note that the sum of $\phi(g_i')b_{iq}$ over all advertiser-query pairs (i,q), such that $\mathcal{O}$ allocated $q \in C$ to i, is equal to $\sum_{i=1}^{k}\phi(i)(s_i - r_i - \sum_{j=1}^{i}t_{ij})$. Recall that $\sum_{j=1}^{i}t_{ij}$ is the amount of budget that advertisers of type i spend on the keywords in $\mathcal{Q}-C$. Similarly, the sum of $\phi(g_a)b_{aq}$ over all advertiser-query pairs (a,q), $q \in C$, in the solution of our algorithm is equal to $\sum_{i=1}^{k}\phi(i)(w_i - -\sum_{j=i}^{k}t_{ji})$. Therefore, by Inequality (10) we have

$$\sum_{i=1}^{k}\phi(i)(s_i - r_i - \sum_{j=1}^{i}t_{ij}) \quad < \quad \frac{1}{\alpha}\sum_{i=1}^{k}\phi(i)(w_i - \sum_{j=i}^{k}t_{ji})$$

Plugging $w_i = \frac{1}{k}\sum_{j=i}^{k}s_j$ and rearranging the terms we get:

$$\frac{1}{\alpha}\sum_{i=1}^{k}\phi(i)\sum_{j=i}^{k}t_{ji} - \sum_{i=1}^{k}\phi(i)\sum_{j=1}^{i}t_{ij} \quad \le \quad \frac{1}{\alpha}\sum_{i=1}^{k}\phi(i)(\frac{1}{k}\sum_{j=i}^{k}s_j) - \sum_{i=1}^{k}\phi(i)(s_i - r_i) \tag{11}$$

For the l.h.s. and r.h.s. of (11) we have:

$$l.h.s. = \frac{1}{\alpha}\sum_{i=1}^{k}\phi(i)\sum_{j=i}^{k}t_{ji} - \sum_{i=1}^{k}\phi(i)\sum_{j=1}^{i}t_{ij} = \frac{1}{\alpha}(\sum_{i=1}^{k}\sum_{j=1}^{i}(\phi(j)-\alpha\phi(i))t_{ij}) \tag{12}$$

$$\begin{aligned} r.h.s. &= \frac{1}{\alpha k}\sum_{i=1}^{k}\phi(i)\sum_{j=i}^{k}s_j - \sum_{i=1}^{k}\phi(i)s_i + \sum_{i=1}^{k}\phi(i)r_i \\ &= \frac{1}{\alpha}\sum_{i=1}^{k}(\frac{1}{k}\sum_{j=1}^{i}\phi(j) - \alpha\phi(i))s_i + \sum_{i=1}^{k}\phi(i)r_i \\ &= \frac{1}{\alpha}\sum_{i=1}^{k}(\frac{i}{k} + \frac{1}{\alpha}\phi(i) - \frac{1}{\alpha}\phi(0) + o(1) - \alpha\phi(i))s_i + \sum_{i=1}^{k}\phi(i)r_i \end{aligned} \tag{13}$$

In the last expression, we substitute $\frac{1}{k}\sum_{t=1}^{i}\phi(t)$ with $\frac{i}{k} + \frac{1}{\alpha}(\phi(i)-\phi(0)) + o(1)$, see Lemma 23. From (12) and (13) the lemma follows. □

To compute $\gamma_{\mathcal{O}}$ we use the factor revealing LP technique [14, 13]. The idea is to show that certain quantities in the algorithm (in this case, the s_i's, r_i's, and t_{ij}'s) satisfy several inequalities (in this case, the inequality given by the above lemma and several trivial inequalities), and then bound the approximation factor of the algorithm by the solution of an optimization program that consists of variables representing these quantities and the established inequalities. In our case, consider the following linear program:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{k} \frac{i}{k} s_i && && (14)\\
\text{subject to} \quad & \sum_{i=1}^{k} s_i - \sum_{i=1}^{k} r_i = V && \\
& \sum_{i=1}^{k} \left(\frac{i}{k} - \frac{1}{\alpha}\phi(0) + (\frac{1}{\alpha} - \alpha)\phi(i) \right) s_i + \alpha \sum_{i=1}^{k} \phi(i) r_i \;\geq \sum_{i=1}^{k} \sum_{j=1}^{i} (\phi(j) - \alpha\phi(i)) t_{ij} && \\
& \frac{1}{k} s_i \geq t_{ij} && \forall 1 \leq j \leq i \leq k \\
& s_i \geq r_i && \forall 1 \leq i \leq k \\
& s_i, r_i, t_{ij} \geq 0 && \forall 1 \leq j \leq i \leq k
\end{aligned}
$$

The value V is treated as a constant (which represents the value achieved by the algorithm $\mathcal{O}$) in the above program. The objective function of this linear program represents the revenue of the algorithm and the linear constraints are satisfied by the quantities s_i, r_i, and t_{ij} coming from the algorithm, except for the missing $o(1)$ term in the inequality of Lemma 8, which as we will argue has a negligible effect. This means that the quantities s_i, r_i, and t_{ij} from the algorithm constitute an almost feasible solution for the above linear program. Therefore, if we prove that for every V, the value of *every* feasible solution of the above program is at least some factor ψ times V, it follows that the approximation ratio of the algorithm $\mathcal{Q}(\alpha)$ with respect to $\mathcal{O}$ is at least ψ. The rest of this section is mainly devoted to computing the solution of the above linear program.

By LP duality, to give a lower bound on the solution of the linear program (14) (and hence on the approximation ratio of our algorithm with respect to $\mathcal{O}$), we need to give a feasible solution to the dual of this program. Therefore, we start by writing the dual of the above linear program. Variables x, y, z, and u correspond respectively to the first, second, third, and forth set of primal constraints.

$$
\begin{aligned}
\text{maximize} \quad & Vx && && (15)\\
\text{subject to} \quad & x + \left(\frac{i}{k} - \frac{1}{\alpha}\phi(0) + (\frac{1}{\alpha} - \alpha)\phi(i) \right) y + \frac{1}{k} \sum_{j=1}^{i} z_{ij} + u_i \leq \frac{i}{k} && \forall 1 \leq i \leq k \\
& y\alpha\phi(i) - x - u_i \leq 0 && \forall 1 \leq i \leq k \\
& (\alpha\phi(i) - \phi(j))y - z_{ij} \leq 0 && \forall 1 \leq j \leq i \leq k \\
& y, z_{ij}, u_i \geq 0 && \forall 1 \leq j \leq i \leq k
\end{aligned}
$$

The following lemma simplifies this dual program.

Lemma 9. *The maximization program (15) is equivalent to the following:*

$$\begin{aligned}
&\textit{maximize} \quad Vx && (16)\\
&\textit{subject to} \quad x \le \frac{i}{k} - \left(\frac{i}{k} - \frac{1}{\alpha}\phi(0) + (\frac{1}{\alpha} - \alpha)\phi(i) + \frac{1}{k}\sum_{j=1}^{i}\max(0, \alpha\phi(i) - \phi(j))\right) y && \forall 1 \le i \le k\\
&\qquad\qquad y\alpha\phi(i) \le \frac{i}{k} - \left(\frac{i}{k} - \frac{1}{\alpha}\phi(0) + (\frac{1}{\alpha} - \alpha)\phi(i) + \frac{1}{k}\sum_{j=1}^{i}\max(0, \alpha\phi(i) - \phi(j))\right) y && \forall 1 \le i \le k\\
&\qquad\qquad y \ge 0
\end{aligned}$$

Proof. We obtain (16) from (15) by eliminating the variables z_{ij} and u_i. First, we eliminate the variable z_{ij}. The only constraints bounding z_{ij} from below are $z_{ij} \ge 0$ and $z_{ij} \ge (\alpha\phi(i) - \phi(j))y$. Therefore, without loss of generality, we may set $z_{ij} = \max(0, (\alpha\phi(i) - \phi(j))y)$. Given that $y \ge 0$, this can be written as $z_{ij} = \max(0, \alpha\phi(i) - \phi(j))y$. Next, we eliminate u_i. The only constraint in (15) bounding u_i from above is the first inequality. Therefore, in any feasible solution of the program we can set $u_i = \frac{i}{k} - \left(x + \left(\frac{i}{k} - \frac{1}{\alpha}\phi(0) + (\frac{1}{\alpha} - \alpha)\phi(i)\right) y + \frac{1}{k}\sum_{j=1}^{i}\max(0, \alpha\phi(i) - \phi(j))y\right)$ without violating feasibility or changing the value of the optimal solution. Replacing this value into the constraints $u_i \ge 0$ and $y\alpha\phi(i) - x - u_i \le 0$ (which are the only other constraints in the program that involve u_i) yields the first and the second constraints of the program (16). □

All that remains is to "guess" a solution for the above program where x is greater than or equal to the ratio stated in Theorem 5, and show that it is feasible. To guess a solution for the above program, we approximate it at the limit (as k tends to infinity) with a "continuous" version, and use standard tools of calculus to solve the resulting optimization program. We will be intentionally sloppy in deriving the continuous program from the linear program (for example, we liberally replace discrete terms by their limit as $k \to \infty$ without worrying about the existence of the limit or the error terms), since our goal in this step is to merely guess a solution, and once this solution is derived, the formal proof will be complete by verifying the feasibility of a small perturbation of this solution in the linear program (16).

We start by approximating the limit of the maximization program (16) as k tends to infinity as follows: Let $f \in [0, 1]$ represent i/k. The value $\phi(i)$ can be approximated by $\Phi(f)$, and the summation $\frac{1}{k}\sum_{j=1}^{i}\max(0, \alpha\phi(i) - \phi(j))$ can be approximated by

$$\int_0^f \max(0, \alpha\Phi(f) - \Phi(x))dx. \tag{17}$$

We define:

$$\ell(f) = \min_{x \ge 0}\{\Phi(x) \le \alpha\Phi(f)\}. \tag{18}$$

Since Φ is a decreasing function, the value of ℓ can be written as follows:

$$\ell(f) = \begin{cases} 0 & f \le \ell^* \\ 1 + \frac{1}{\alpha}\ln(1 - \alpha\Phi(f)) & f \ge \ell^*, \end{cases} \tag{19}$$

where

$$\ell^* = \max_f\{\Phi(0) \le \alpha\Phi(f)\} = 1 + \frac{1}{\alpha}\ln\left(1 - \frac{1}{\alpha}\Phi(0)\right). \tag{20}$$

Therefore, the integral (17) can be written as

$$\begin{aligned}\int_0^f \max(0, \alpha\Phi(f) - \Phi(x))dx &= \int_{\ell(f)}^f (\alpha\Phi(f) - \Phi(x))dx \\ &= \alpha\Phi(f)(f - \ell(f)) - \int_{\ell(f)}^f (1 - e^{\alpha(x-1)})dx \\ &= (f - \ell(f))(\alpha\Phi(f) - 1) + \left(\frac{1}{\alpha}e^{\alpha(f-1)} - \frac{1}{\alpha}e^{\alpha(\ell(f)-1)}\right) \\ &= (f - \ell(f))(\alpha\Phi(f) - 1) + \frac{1}{\alpha}(\Phi(\ell(f)) - \Phi(f))\end{aligned}$$

Given this, we estimate the dual linear program (16) by the following continuous optimization problem. Note that we have omitted V in the objective, as it does not change the optimal solution.

$$\begin{array}{lll} \text{maximize} & x & \\ \text{subject to} & x \le f - h(f)y & \forall f \in [0,1] \\ & (\alpha\Phi(f) + h(f))\, y \le f & \forall f \in [0,1] \\ & y \ge 0, & \end{array} \tag{21}$$

where

$$h(f) = f - \frac{1}{\alpha}\Phi(0) + (\frac{1}{\alpha} - \alpha)\Phi(f) + (f - \ell(f))(\alpha\Phi(f) - 1) + \frac{1}{\alpha}(\Phi(\ell(f)) - \Phi(f))$$

For $f \le \ell^*$, we have $\ell(f) = 0$, and therefore

$$\begin{aligned} h(f) &= f - \frac{1}{\alpha}\Phi(0) + (\frac{1}{\alpha} - \alpha)\Phi(f) + (\alpha\Phi(f) - 1)f + \frac{1}{\alpha}(\Phi(0) - \Phi(f)) \\ &= \alpha(f-1)\Phi(f). \end{aligned}$$

For $f \ge \ell^*$, we have $\Phi(\ell(f)) = \alpha\Phi(f)$, and therefore

$$\begin{aligned} h(f) &= f - \frac{1}{\alpha}\Phi(0) + (\frac{1}{\alpha} - \alpha)\Phi(f) + (f - \ell(f))(\alpha\Phi(f) - 1) + \frac{\alpha - 1}{\alpha}\Phi(f) \\ &= \ell(f) - \frac{1}{\alpha}\Phi(0) + (1 - \alpha + \alpha f - \alpha\ell(f))\Phi(f). \end{aligned}$$

To summarize, the optimum value of x in the optimization problem (21) can be written as follows:

$$\begin{array}{lll} \text{maximize} & \min\limits_{f\in[0,1]}\{f - h(f)y\} & \\ \text{subject to} & (\alpha\Phi(f) + h(f))\, y \le f & \forall f \in [0,1] \\ & y \ge 0, & \end{array} \tag{22}$$

where

$$h(f) = \begin{cases} \alpha(f-1)\Phi(f) & f \le \ell^* \\ \ell(f) - \frac{1}{\alpha}\Phi(0) + (1 - \alpha + \alpha f - \alpha\ell(f))\Phi(f) & f \ge \ell^*. \end{cases} \tag{23}$$

The following lemma simplifies the constraint of the program. The proof of this lemma is given in Appendix C.

Lemma 10. *For every $f \in [0,1]$, $\alpha\Phi(f) + h(f) \leq \alpha\Phi(0)f$.*

By the above lemma, the constraint of the optimization problem (22) at any $f \in [0,1]$ is dominated by the inequality $\alpha\Phi(0)fy \leq f$, or $y \leq (\alpha\Phi(0))^{-1}$. Also, taking f arbitrarily close to 0, we obtain that any feasible solution of (22) should satisfy $y \leq (\alpha\Phi(0))^{-1}$. Therefore the problem is equivalent to computing

$$\max_{y \in [0, \frac{1}{\alpha\Phi(0)}]} \min_{f \in [0,1]} \{f - h(f)y\} \tag{24}$$

Next, we turn our attention to the function $f - h(f)y$, and show that the minimum of this function can only occur in one of four possible values of f. The following lemma, whose proof is given in Appendix C, takes care of the range of $f \geq \ell^*$.

Lemma 11. *For any value of $y \geq 0$, the function $f - h(f)y$ is a concave function of f in the range $f \in [\ell^*, 1]$. In particular, we have*

$$\min_{f \in [\ell^*, 1]} \{f - h(f)y\} = \min\{\ell^* - h(\ell^*)y, 1 - h(1)y\}.$$

Next, we consider the function $f - h(f)y$ in the range $f \leq \ell^*$. Recall that by (23), in this range this function can be written as $f - h(f)y = f - \alpha y(f-1)\Phi(f)$. The proof of the following lemma is given in Appendix C.

Lemma 12. *For any value of $y \geq 0$, the function $f - h(f)y$ is a concave of f on $[0, \max(0, 1-2/\alpha)]$ and a convex function on $[\max(0, 1-2/\alpha), \ell^*]$. The minimum of this function over $f \in [0, \ell^*]$ occurs in one of the three points 0, ℓ^*, or $\mu(y)$, where $\mu(y)$ is the root of the equation $1 - e^{\alpha(x-1)} + \alpha(1-x)e^{\alpha(x-1)} = 1/(\alpha y)$ in the interval $[\max(0, 1-2/\alpha), \ell^*]$, if such a root exists.*

By Lemmas 11 and 12, the minimum in the optimization problem (24) occurs in one of four possible points, 0, 1, ℓ^*, or $\mu(y)$. Given this and guided by numerical calculations, we are now ready to guess a solution for (24) and therefore for the dual program (16). This is done in the following two lemmas.

Lemma 13. *Let $\alpha^* > 1$ denote the root of the equation $(x^2 + x + 1)e^{-x} = 1$. For $\alpha \geq \alpha^*$, the optimization problem (24) has a solution of value at least*

$$x^* := \frac{\alpha\Phi(0)}{(\alpha - \frac{1}{\alpha})\Phi(0) + 1}. \tag{25}$$

Proof. Consider the following value for y. This is the value that makes the value of the function $f - h(f)y$ at 0 equal to its value at 1.

$$y^* = \frac{1}{(\alpha - \frac{1}{\alpha})\Phi(0) + 1} \tag{26}$$

First, we observe that

$$\begin{aligned}
\alpha y^* &= \frac{\alpha^2}{(\alpha^2-1)(1-e^{-\alpha})+\alpha} \\
&= \frac{\alpha^2}{\alpha^2+(\alpha^2-1)(1-e^{-\alpha})+\alpha-\alpha^2} \\
&= \frac{\alpha^2}{\alpha^2+(\alpha-1)\left((\alpha+1)(1-e^{-\alpha})-\alpha\right)} \\
&= \frac{\alpha^2}{\alpha^2+(\alpha-1)e^{-\alpha}\left(e^{\alpha}-(\alpha+1)\right)} \\
&\leq 1 \qquad (27)
\end{aligned}$$

where the last inequality follows from $e^{\alpha} > 1+\alpha$. Therefore, $y^* \leq \frac{1}{\alpha} \leq \frac{1}{\alpha\Phi(0)}$.

Next, we prove that at $y = y^*$, the function $f - h(f)y$ is at least x^* for every $f \in [0,1]$. By Lemmas 11 and 12, it is enough to verify this at 0, 1, ℓ^*, and $\mu(y^*)$ (if exists). By the choice of y^*, this inequality holds with equality at 0 and 1. We now verify this inequality for at $\mu(y^*)$, if it exists. We calculate the derivative of the function $f - h(f)y$ with respect to f at $f = \max(0, 1-2/\alpha)$ (assuming this value is less than ℓ^*). Recall that by Equation (23), $h(f) = \alpha(f-1)\Phi(f)$ in $[0, \ell^*]$, and hence

$$\begin{aligned}
\left.\frac{\partial(f-h(f)y^*)}{\partial f}\right|_{f=0} &= \left.(1-\alpha y^*(1-e^{\alpha(f-1)}-\alpha(f-1)e^{\alpha(f-1)}))\right|_{f=0} \\
&= 1-\alpha y^*(1-e^{-\alpha}+\alpha e^{-\alpha}) \\
&= y^*\left((\alpha-\frac{1}{\alpha})(1-e^{-\alpha})+1-\alpha(1-e^{-\alpha}+\alpha e^{-\alpha})\right) \\
&= \frac{y^*}{\alpha}\left(-1+e^{-\alpha}+\alpha-\alpha^3 e^{-\alpha}\right) \\
&= \frac{y^*}{\alpha}(\alpha-1)\left(1-(\alpha^2+\alpha+1)e^{-\alpha}\right) \qquad (28)
\end{aligned}$$

and

$$\begin{aligned}
\left.\frac{\partial(f-h(f)y^*)}{\partial f}\right|_{f=1-2/\alpha} &= \left.(1-\alpha y^*(1-e^{\alpha(f-1)}-\alpha(f-1)e^{\alpha(f-1)}))\right|_{f=1-2/\alpha} \\
&= 1-\alpha y^*(1+e^{-2}) \\
&= \frac{y^*}{\alpha}\left((\alpha^2-1)(1-e^{-\alpha})+\alpha-(1+e^{-2})\alpha^2\right).
\end{aligned}$$

The derivative at 0 is zero for $\alpha = \alpha^*$, and is positive afterwards. Also, it is easy to see that the derivative at $1-2/\alpha$ is positive for $2 \leq \alpha \leq 3$. Hence, the derivative of the function $f - h(f)y^*$ with respect to f at $f = \max(0, 1-2/\alpha)$ is non-negative for $\alpha \in [\alpha^*, 3]$. This, together with the fact that this function is convex on $[\max(0, 1-2/\alpha), \ell^*]$, implies that for this range of values of α, no root $\mu(y^*)$ of the derivative of $f - h(f)y^*$ in the interval $[\max(0, 1-2/\alpha), \ell^*]$ exists. For $\alpha > 3$, we argue as follows: since $\mu(y^*)$ is a root of the equation $1 - e^{\alpha(x-1)} + \alpha(1-x)e^{\alpha(x-1)} = 1/(\alpha y^*)$, we have $\alpha y^* \Phi(\mu(y^*)) = 1 - \alpha^2(1-\mu(y^*))e^{\alpha(\mu(y^*)-1)}y^*$. Hence

$$f - h(f)y^*|_{f=\mu(y^*)} = \mu(y^*) + \alpha y^*\Phi(\mu(y^*))(1-\mu(y^*)) = 1 - \alpha^2(1-\mu(y^*))^2 e^{\alpha(\mu(y^*)-1)}y^*. \qquad (29)$$

Now, we note that the function $1-\alpha^2(1-x)^2e^{\alpha(x-1)}$ is an increasing function of x for $x \geq 1-2/\alpha$. This is because the derivative of this function with respect to x is

$$-\alpha^2(-2(1-x)+\alpha(1-x)^2)e^{\alpha(x-1)} = \alpha^3(1-x)(x-1+2/\alpha)e^{\alpha(x-1)},$$

which is positive for $x > 1-2/\alpha$. Therefore, since $\mu(y^*) \geq 1-2/\alpha$, we have

$$f - h(f)y^*|_{f=\mu(y^*)} \geq 1-\alpha^2(1-(1-2/\alpha))^2e^{\alpha((1-2/\alpha)-1)}y^* = 1-4e^{-2}y^*. \tag{30}$$

On the other hand, the derivative of the function $1-\frac{\Phi(0)}{\alpha}$ with respect to α is

$$-\frac{\alpha e^{-\alpha}+1-e^{-\alpha}}{\alpha^2} = \frac{e^{\alpha}}{\alpha^2}(e^{\alpha}-1-\alpha) \geq 0,$$

and therefore $1-\frac{\Phi(0)}{\alpha}$ is an increasing function of α. Hence, for every $\alpha \geq 3$, we have

$$1-\frac{\Phi(0)}{\alpha} \geq 1-\frac{1-e^{-3}}{3} > 4e^{-2}.$$

Thus,

$$x^* = 1-(1-\frac{\Phi(0)}{\alpha})y^* < 1-4e^{-2}y^* \tag{31}$$

Equations (30) and (31) together show that the value of the function $f-h(f)y$ at $f=\mu(y^*)$ is at least x^*.

Finally, we prove that the value of $f-h(f)y^*$ at $f=\ell^*$ is at least x^*. As we observed earlier, $1-\frac{\Phi(0)}{\alpha}$ is an increasing function of α. Therefore, for every $\alpha \geq 1$, $1-\frac{\Phi(0)}{\alpha} \geq e^{-1}$, and hence

$$\ell^* = 1+\frac{1}{\alpha}\ln\left(1-\frac{1}{\alpha}\Phi(0)\right) \geq 1-\frac{1}{\alpha}.$$

This implies that $1-1/\alpha$ always belongs to the interval $[\max(0,1-2/\alpha),\ell^*]$. Recall that by Lemma 12, $f-h(f)y^*$ is convex in this interval. Furthermore, the derivative of this function at $f=1-1/\alpha$ is equal to $1-\alpha y^*$, which by (27) is non-negative. Therefore, the value of $f-h(f)y^*$ over $[0,\ell^*]$ cannot be minimized at ℓ^*. □

Lemma 14. *For every $\alpha \leq \alpha^*$, the equation $(\alpha(f-1))^2e^{\alpha(f-1)} = 1-\frac{\Phi(0)}{\alpha}$ has a unique solution f^* in $[0,1]$.*

Proof. As $\alpha \leq \alpha^* < 2$, the value $\alpha(f-1)$ ranges in $[-2,0]$. In this range, $(\alpha(f-1))^2e^{\alpha(f-1)}$ is a decreasing function of f. At $f=1$, the value of this function is 0, and at $f=0$, its value is

$$\alpha^2e^{-\alpha} = 1-\frac{\Phi(0)}{\alpha}+\frac{\alpha^3e^{-\alpha}+1-e^{-\alpha}}{\alpha}-1 = 1-\frac{\Phi(0)}{\alpha}+\frac{\alpha-1}{\alpha}((\alpha^2+\alpha+1)e^{-\alpha}-1) \geq 1-\frac{\Phi(0)}{\alpha},$$

where the last inequality follows from the fact that by the definition of α^*, for every $\alpha \leq \alpha^*$, $(\alpha^2+\alpha+1)e^{-\alpha} \geq 1$. Therefore, there must be a point f^* in $[0,1]$ where the value of $(\alpha(f-1))^2e^{\alpha(f-1)}$ becomes equal to $1-\Phi(0)/\alpha$. Furthermore, since this function is strictly decreasing in this interval, this point is unique. □

Lemma 15. *For $\alpha \leq \alpha^*$, the optimization problem (24) has a solution of value at least*

$$x^* := 1 - (1 - \frac{\Phi(0)}{\alpha}) \frac{1 - f^*}{1 - \frac{\Phi(0)}{\alpha} + \alpha(1 - f^*)\Phi(f^*)} \tag{32}$$

times V, where f^* is as defined in Lemma 14.

Proof. Consider the following value for y. This is the value that makes the value of the function $f - h(f)y$ equal at $f = 1$ and at $f = \mu(y)$.

$$y^* = \frac{1 - f^*}{1 - \frac{\Phi(0)}{\alpha} + \alpha(1 - f^*)\Phi(f^*)} \tag{33}$$

First, we show that $y^* \leq \frac{1}{\alpha\Phi(0)}$. Since $x^2 e^{-x}$ is an increasing function of x for $0 < x < 2$, if $\alpha(1 - f^*) < 1$, we have $(\alpha(1 - f^*))^2 e^{\alpha(f^*-1)} < e^{-1}$. By the definition of f^*, this means that $1 - \frac{\Phi(0)}{\alpha} < e^{-1}$, which contradicts the fact that $1 - \frac{\Phi(0)}{\alpha}$ is an increasing function of α and $\alpha \geq 1$. Therefore, we must have

$$\alpha(1 - f^*) \geq 1 \tag{34}$$

Thus, $\alpha(1 - f^*)e^{\alpha(f^*-1)} \leq (\alpha(1 - f^*))^2 e^{\alpha(f^*-1)} = 1 - \frac{\Phi(0)}{\alpha}$. Hence, $1 - \frac{\Phi(0)}{\alpha} + \alpha(1 - f^*)\Phi(f^*) \geq \alpha(1 - f^*)$, implying that

$$y^* \leq \frac{1}{\alpha} \leq \frac{1}{\alpha\Phi(0)}. \tag{35}$$

Next, we prove that at $y = y^*$, for every $f \in [0, 1]$, the function $f - h(f)y^*$ is at least x^*. By Lemmas 11 and 12, we only need to verify this at four values of f: 0, $\mu(y^*)$, ℓ^*, and 1. It is clear that the inequality holds with equality at $f = 1$. It also holds with equality at $f = \mu(y^*)$ by the following argument: first, observe that f^* is a root of the equation $1 - e^{\alpha(x-1)} + \alpha(1 - x)e^{\alpha(x-1)} = 1/(\alpha y^*)$, and hence $\mu(y^*) = f^*$. Given this, we obtain:

$$\begin{aligned}
\mu(y^*) - \alpha y^*(\mu(y^*) - 1)\Phi(\mu(y^*)) &= f^* - \frac{\alpha(1 - f^*)}{1 - \frac{\Phi(0)}{\alpha} + \alpha(1 - f^*)\Phi(f^*)}(f^* - 1)\Phi(f^*) \\
&= 1 - (1 - f^*)\left(1 - \frac{\alpha(1 - f^*)\Phi(f^*)}{1 - \frac{\Phi(0)}{\alpha} + \alpha(1 - f^*)\Phi(f^*)}\right) \\
&= 1 - (1 - f^*)\frac{1 - \frac{\Phi(0)}{\alpha}}{1 - \frac{\Phi(0)}{\alpha} + \alpha(1 - f^*)\Phi(f^*)} = x^*
\end{aligned}$$

Next, we prove that the value of $f - h(f)y^*$ at $f = \ell^*$ is at least x^*. This is done by an argument similar to the one in the proof of Lemma 15: as proved there, $1 - 1/\alpha$ always belongs to the interval $[\max(0, 1 - 2/\alpha), \ell^*]$, and $f - h(f)y^*$ is convex in this interval. Furthermore, the derivative of this function at $f = 1 - 1/\alpha$ is equal to $1 - \alpha y^*$, which by (35) is non-negative. Therefore, the value of $f - h(f)y^*$ over $[0, \ell^*]$ cannot be minimized at ℓ^*.

Finally, we need to show that $f - h(f)y^*$ at $f = 0$ is at least x^*. Consider the expression

$$\frac{1 - x}{1 - \frac{\Phi(0)}{\alpha} + \alpha(1 - x)\Phi(x)} \tag{36}$$

as a function of x. The derivative of this function with respect to x can be written as

$$\frac{-(1-\frac{\Phi(0)}{\alpha}+\alpha(1-x)\Phi(x))-(1-x)(-\alpha\Phi(x)-\alpha^2(1-x)e^{\alpha(x-1)})}{(1-\frac{\Phi(0)}{\alpha}+\alpha(1-x)\Phi(x))^2}$$

The denominator of this expression is always positive, and the numerator is equal to $\alpha^2(1-x)^2e^{\alpha(x-1)}-(1-\frac{\Phi(0)}{\alpha})$. This value is zero at $x=f^*$, positive for $0<x<f^*$, and negative for $f^*<x<1$. Therefore, $x=f^*$ maximizes the function (36) over $[0,1]$. In particular, the value of this function at $x=f^*$ is at least its value at $x=0$. In other words,

$$y^* \geq \frac{1}{1-\frac{\Phi(0)}{\alpha}+\alpha\Phi(0)} \tag{37}$$

The derivative of the function $f-h(f)y^*$ with respect to f at $f=0$ is $1-\alpha y^*(1-e^{-\alpha}+\alpha e^{-\alpha})$. By inequality (37) this value is not greater than

$$1-\frac{\alpha(1-e^{-\alpha}+\alpha e^{-\alpha})}{1-\frac{1-e^{-\alpha}}{\alpha}+\alpha(1-e^{-\alpha})}=\frac{\frac{-(\alpha-1)}{\alpha}((\alpha^2+\alpha+1)e^{-\alpha}-1)}{1-\frac{1-e^{-\alpha}}{\alpha}+\alpha(1-e^{-\alpha})}$$

By the definition of α^*, this value is negative for $\alpha<\alpha^*$. Therefore, the derivative of $f-h(f)y^*$ is non-positive at $f=0$. This means that $f=0$ cannot be the minimum of $f-h(f)y^*$ over $[0,1]$, thereby completing the proof. □

We are now ready to prove Theorem 5.

Proof of Theorem 5. Lemma 7 establishes that the competitive ratio of $\mathcal{Q}_k(\alpha)$ with respect to the optimum offline solution is at least γ_{OPT}. Also, lemmas 13 and 15 give solutions y^* to the optimization program (24). This can be easily turned into a solution for the dual linear program (16) as follows: we simply subtract a small value δ from y^*. Since the coefficients of the linear program (16) tend to the coefficients in the continuous program (24) as k tends to infinity, for any fixed $\delta>0$, if k is large enough, subtracting δ from y^* yields a solution to the linear program (16) that satisfies the second inequality. Furthermore, since the coefficient of y in the first inequality in (16) is bounded and tends to $h(i/k)$ as k tends to infinity, the value x obtained for this solution is only a small additive factor ϵ away from y, where ϵ tends to zero as k tends to infinity. Therefore, for any $\epsilon>0$, if k is large enough, there is a feasible solution for the dual program (16) of value at least $(x^*-\epsilon)V$, where x^* is the factor obtained in Lemmas 13 and 15. By weak LP duality, this implies that when k is large, any solution of the linear program (14) has a value at least $(x^*-\epsilon)V$. This means that the competitive ratio of $\mathcal{Q}(\alpha)$ with respect to $\mathcal{O}$ is at least $\gamma_{\mathcal{O}}(\alpha)$. □

Tight Example We first give a tight example for this analysis for $\alpha\geq\alpha^*$. Suppose there are two advertisers with budgets

$$B_1=\frac{1}{\alpha\Phi(0)}\int_0^1\Phi(f)df=\frac{1}{\alpha\Phi(0)}(1-\frac{1}{\alpha}(1-e^{-\alpha}))=\frac{1}{\alpha\Phi(0)}(1-\frac{1}{\alpha}\Phi(0))$$

and $B_2=1$. The sequence of keywords arrives in two phases. Consider an arbitrary small $\epsilon>0$. In the phase one, a sequence of keywords arrives for which advertiser 1 bids $\frac{\Phi(f_2)}{\alpha\Phi(0)}\epsilon$ and advertiser

2 bids slightly more than ϵ. $\mathcal{O}$ recommends advertiser 1 for the query. However, $\mathcal{Q}(\alpha)$ allocates all these queries to the second advertiser. This continues up to the time advertiser 2 runs out of budget. After that, the second phase starts and a set of keywords shows up that only 2 is interested in but she has no budget left. Under this scenario, the total revenue of $\mathcal{O}$ is equal to $\frac{1}{\alpha\Phi(0)}\int_0^1 \Phi(f)df + 1 = B_1 + B_2$. However, the total revenue of the algorithm is equal to 1. Therefore, we have

$$\gamma_{\text{OPT}} \leq \frac{B_2}{B_1 + B_2} = \frac{\alpha(1 - e^{-\alpha})}{(\alpha - \frac{1}{\alpha})(1 - e^{-\alpha}) + 1}$$

Also, this example corresponds to a feasible primal solution with $s_0 = B_1$, $s_1 = B_2$, and all other variables equal to zero. For the case $\alpha < \alpha^*$, the optimal primal solution has a similar structure except that $s_{f^*} = B_1$ and $t_{1j} = \frac{1}{k}B_1$ for $j \leq kf^*$. The corresponding scenario is as follows: At the beginning, a set of queries are allocated to the first advertiser, until she spent a fractionf^* of her budget. After that, the second sequence of of queries shows up for which $b_{1j} = \frac{\Phi(f_2)}{\alpha\Phi(f^*)}\epsilon$ and b_{2j} is slightly more than ϵ. Hence, the algorithm allocates all of the queries to the second advertisers. At the end, the third sequence of queries shows up in which only advertiser 2 is interested. The rest of the example and its analysis is similar to the previous case.

4.3 Almost accurate estimates

In this section, as we see below, we can get a guarantee with respect to the optimal solution. In this section we describe an algorithm $\mathcal{O}$ is based on the offline problem solved on the estimates of query frequencies. As queries in $\mathcal{Q}$ arrive online, $\mathcal{O}$ assigns them to advertisers as suggested by the offline solution. Here we have to be careful about two points: first, if we see a frequency more than what was predicted, the expert simply discards the query (e.g., allocates the query to an advertiser with bid 0 for all keywords). Second, when a query of a certain type arrives, allocate the query to the highest bid among the advertisers who are assigned this query in the offline solution. The above definition guarantees that if our estimate of the frequency of each type of query is within a $(1 \pm \epsilon)$ factor of the actual frequency of that type of query, then the value of the solution that the expert proposes is at least a $(1 - 2\epsilon)$ fraction of the optimal offline solution. Therefor algorithm achieves a $(1 - 2\epsilon)\gamma_a$ fraction of the optimal revenue.

5 Conclusion

We studied a new approach toward online optimization in the presence of uncertain estimates. Our approach goes beyond competitive analysis by using the additional information available about the input while still maintains a bounded competitive ratio in the worst-case. The power of our framework was demonstrated by applying it to a wide range of online problems. We believe that our framework can be useful for many other practically and theoretically important problems, and extends the applicability of the online optimization techniques in practice.

References

[1] Susanne Albers. Online algorithms: a survey. *Mathematical Programming*, 2003.

[2] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. In *STOC*, pages 100–105. ACM, 2003.

[3] N. Ascheuer and J. Rambau M. Grotschel, SO. Krumke. Combinatorial online optimization. *Operations Research Proceedings*, 1998.

[4] James Aspnes, Yossi Azar, Amos Fiat, Serge A. Plotkin, and Orli Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *STOC*, pages 623–631, 1993.

[5] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. *Theor. Comput. Sci.*, 324(2-3):313–324, 2004.

[6] Yossi Azar. On-line load balancing. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms*, volume 1442 of *Lecture Notes in Computer Science*, pages 178–195. Springer, 1996.

[7] Yossi Azar, Andrei Z. Broder, and Anna R. Karlin. On-line load balancing. *Theor. Comput. Sci.*, 130(1):73–84, 1994.

[8] Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems. *Proceedings of the twenty-ninth Annual ACM Symposium on Theory of Computing*, 1997.

[9] Niv Buchbinder, Kamal Jain, and Seffi Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th Annual European Symposium on Algorithms*, 2007.

[10] Jaroslaw Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In *APPROX-RANDOM*, pages 29–43, 2007.

[11] Amos Fiat and Gerhard Woeginger. *Online algorithms: The state of the art*. Springer-Verlag, 1998.

[12] Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

[13] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *J. ACM*, 50(6):795–824, 2003.

[14] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *STOC*, pages 731–740, 2002.

[15] Adam Kalai and Santoush Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71:291307, 2005.

[16] Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. In *FOCS*, pages 394–400. IEEE, 1994.

[17] Sébastien Lahaie, David M. Pennock, Amin Saberi, and Rakesh Vohra. Sponsored search auctions. In N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 28. Cambridge University Press, 2007.

[18] Benny Lehmann Daniel Lehmann and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 52(2), 2006.

[19] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.

[20] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *ACM Conference on Electronic Commerce*, pages 288–294, 2007.

[21] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5), 2007.

[22] Vijay Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2001.

A Proof of Theorem 4

The proof follows immediately from lemmas below.

Lemma 16. *For any time t, $w_t(\mathcal{H}(\gamma)) \leq \gamma w_t(\mathcal{P})$.*

Proof. Let l be the last time the algorithm follows the recommendation of $\mathcal{O}$ instead of $\mathcal{P}$. The activating cost plus the serving cost of the jobs that the algorithm followed $\mathcal{O}$, up to time t, is bounded by $w_l(\mathcal{O}) \leq (\gamma - 1)w_l(\mathcal{P}) \leq (\gamma - 1)w_t(\mathcal{P})$. Also, note that the activating cost plus the serving cost of the jobs that are allocated according to $\mathcal{P}$ is bounded by $w_t(\mathcal{P})$. Therefore, $w_t(\mathcal{H}(\gamma)) \leq \gamma w_t(\mathcal{P})$. □

Lemma 17. *For any time t, $w_t(\mathcal{H}(\gamma)) \leq \frac{\gamma}{\gamma-1} w_t(\mathcal{O})$.*

Proof. Let l be the last time the algorithm ignores the recommendation of $\mathcal{O}$. The activating cost plus the serving cost of the jobs that the algorithm followed $\mathcal{P}$ up to time t is bounded by $w_l(\mathcal{P}) < \frac{1}{\gamma-1} w_l(\mathcal{O}) \leq \frac{1}{\gamma-1} w_t(\mathcal{O})$. Therefore, $w_t(\mathcal{H}(\gamma)) \leq (1 + \frac{1}{\gamma-1}) w_t(\mathcal{O}) \leq \frac{\gamma}{\gamma-1} w_t(\mathcal{O})$. □

Lemma 18. *Suppose algorithm $\mathcal{O}$ makes its recommendation based on an optimal allocation for a given estimate. There exists an algorithm $\mathcal{P}$ such that for any deterministic (γ, γ')-balanced algorithm with respect to $\mathcal{P}$ and $\mathcal{O}$, we have $\gamma' \geq \frac{\gamma}{\gamma-1}$.*

Proof. We prove the claim for a special case of the problem, the online weighted set cover problem. In weighted set cover each subset is associated with a weight and the goal is to cover all elements in the input with subsets of minimum total weight. Consider the following example. There are 3 elements and two subsets $S_1 = \{1, 2\}$ with weight $\gamma - 1$ and $S_2 = \{1, 3\}$ with weight 1. The estimated input sequence is $1, 2$. Based on these estimates, $\mathcal{O}$ chooses the first subset. The first element in the input is 1. However, $\mathcal{P}$ recommends the second subset. Consider the following cases for an algorithm $\mathcal{A}$.

1. $\mathcal{A}$ follows the recommendations of $\mathcal{O}$ and chooses the first subset. However, the second element turns out be 3. In this case, the cost of $\mathcal{A}$ is at least γ, while it could achieve a cost of 1 by following $\mathcal{P}$.

2. $\mathcal{A}$ ignores the recommendations of $\mathcal{O}$ and chooses the second subset. The second element in the input turns out to be 2. In this case, the cost of $\mathcal{A}$ is t γ while the cost of the solution recommended by $\mathcal{O}$ is $\gamma - 1$.

Hence, the lemma follows. □

Algorithm $\mathcal{G}(\gamma)$:

Let $\alpha = \frac{1}{\gamma} - 1$.

Upon the arrival of a new query j:

Let o be the advertiser recommended by $\mathcal{O}$ to receive the query.

Let p be the advertiser recommended by $\mathcal{P}$ to receive the query.

If $\alpha b_{oj} \geq b_{pj}$ and the budget of o is not exhausted,

Assign j to o.

else

Assign j to p.

Figure 5: A $(\gamma, 1-\gamma)$-balanced algorithm for the query allocation problem.

B A $(\gamma, 1-\gamma)$-balanced Algorithm for Query Allocation

In this section we present a family of algorithms parameterized by $\gamma \in [0,1]$ that are essentially $(\gamma, 1-\gamma)$-balanced with respect to given algorithms $\mathcal{P}$ and $\mathcal{O}$. Similar to the load balancing and resource allocation problem, we assume the algorithm has access to two algorithms $\mathcal{P}$ and $\mathcal{O}$ that upon the arrival of every new query, each recommends an advertiser to receive it. The algorithm corresponding to parameter γ is denoted by $\mathcal{G}(\gamma)$. The algorithm is presented in Figure 5. We assume that the search engine charges the advertiser the minimum of his bid and the remaining budget.

The analysis of the algorithm is similar to the online greedy algorithm for query allocation problem, see [18]. Let ε be the maximum ratio of a bid to the budget of an advertiser.

Lemma 19. $V_{\mathcal{G}(\gamma)} \geq \frac{1}{(1+\varepsilon)(1+\alpha)} V_{\mathcal{P}}$

Proof. Let C be the set of queries that are allocated to the advertiser recommended by $\mathcal{P}$. First observe that $\sum_{j \in C} b_{pj} \leq (1+\varepsilon) V_{\mathcal{G}(\gamma)}$. Similarly, $\sum_{j \notin C} b_{oj} \leq (1+\varepsilon) V_{\mathcal{G}(\gamma)}$. Also, by the rule of the algorithm, the sum of the bids of all advertisers that received the queries despite of the recommendation of $\mathcal{P}$ (i.e., $\sum_{j \notin C} b_{oj}$), is at least $\frac{1}{\alpha} \sum_{j \notin C} b_{pj}$. Therefore, we have $V_{\mathcal{P}} \leq (1+\varepsilon)(1+\alpha) V_{\mathcal{G}(\gamma)}$. □

Lemma 20. $V_{\mathcal{G}(\gamma)} \geq \frac{\alpha}{1+\alpha} V_{\mathcal{O}}$

Proof. Let $\mathcal{A}_1$ denote the set of advertisers who exhaust their budget by the end of the algorithm $\mathcal{G}(\gamma)$, and $\mathcal{A}_2$ denote the set of other advertisers. Also, let $\mathcal{Q}_1$ denote the set of queries that $\mathcal{O}$ assigns to an advertiser in $\mathcal{A}_1$, $\mathcal{Q}_2$ denote the set of queries that $\mathcal{O}$ assigns to an advertiser in $\mathcal{A}_2$ and $\mathcal{G}(\gamma)$ follows $\mathcal{O}$'s recommendation, and $\mathcal{Q}_3$ denote the set of queries that $\mathcal{O}$ assigns to an advertiser in $\mathcal{A}_2$, but $\mathcal{G}(\gamma)$ assigns to the advertiser recommended by $\mathcal{P}$.

The total revenue that $\mathcal{O}$ derives from queries in $\mathcal{Q}_1$ is at most the total budget of the advertisers in $\mathcal{A}_1$, which is the amount of revenue that $\mathcal{G}(\gamma)$ derives from these advertisers. Also, the revenue derived by $\mathcal{O}$ from queries in $\mathcal{Q}_2$ is the same as the revenue $\mathcal{G}(\gamma)$ derives on these queries, which is at most the total revenue $\mathcal{G}(\gamma)$ derives from advertisers in $\mathcal{A}_2$. Therefore, the total revenue of $\mathcal{O}$ on queries in $\mathcal{Q}_1 \cup \mathcal{Q}_2$ is at most $V_{\mathcal{G}(\gamma)}$.

Consider a query $j \in \mathcal{Q}_3$ that $\mathcal{O}$ assigns to an advertiser $o \in \mathcal{A}_2$, but $\mathcal{P}$ and $\mathcal{G}(\gamma)$ assign to p. Since the budget of o is not exhausted even until the end of the algorithm, we must have

$$b_{pj} > \alpha b_{oj} \Rightarrow b_{oj} < \frac{1}{\alpha} b_{pj}.$$

Summing this inequality for all $j \in \mathcal{Q}_3$, we obtain that the total revenue of $\mathcal{O}$ on queries in $\mathcal{Q}_3$ is less than $\frac{1}{\alpha}$ of the total bid on such queries by the advertiser $\mathcal{G}(\gamma)$ assigns them to, which is at most $\frac{1}{\alpha} V_{\mathcal{G}(\gamma)}$.

Therefore, the total revenue of $\mathcal{O}$ is at most $(1 + \frac{1}{\alpha}) V_{\mathcal{G}(\gamma)}$. □

Lemmas above immediately lead to the following theorem.

Theorem 21. *For $0 \leq \gamma \leq 1$, and $\alpha = \frac{1}{\gamma} - 1$ we have*

$$V_{\mathcal{G}(\gamma)} \geq \max\{\frac{1}{1 + \alpha(1+\varepsilon)} V(\mathcal{P}), \frac{\alpha}{1+\alpha} V(\mathcal{O})\}$$

Corollary 22. *By the assumption that bids are small compared to the budget, algorithm $\mathcal{G}(\gamma)$ is essentially a $(\gamma, 1-\gamma)$-balanced algorithm with respect to $\mathcal{P}$ and $\mathcal{O}$.*

C Skipped Proofs from Section 4

Lemma 23. *For every i, j, as $k \to \infty$, we have*[6]

$$\frac{1}{k} \sum_{t=j}^{i} \phi(t) = \frac{i-j+1}{k} + \frac{1}{\alpha}(\phi(i) - \phi(j-1)) + o(1).$$

Proof. Plugging (6) we get,

$$\sum_{t=j}^{i} \phi(t) = \sum_{t=j}^{i} \left(1 - (1 - \frac{1}{k})^{\alpha(k-t)}\right) = i - j + 1 - (1 - \frac{1}{k})^{\alpha(k-i)} \sum_{t=0}^{i-j} (1 - \frac{1}{k})^{\alpha(t)}$$

Substituting the geometric sum,

$$\sum_{t=j}^{i} \phi(t) = i - j + 1 - (1 - \frac{1}{k})^{\alpha(k-i)} \frac{1 - (1 - \frac{1}{k})^{\alpha(i-j+1)}}{1 - (1 - \frac{1}{k})^{\alpha}} = i - j + 1 + \frac{\phi(i) - \phi(j-1)}{1 - (1 - \frac{1}{k})^{\alpha}}$$

Now note that $\lim_{k \to \infty} k(1 - (1 - \frac{1}{k})^{\alpha}) = \alpha$. Therefore, we have

$$\frac{1}{k} \sum_{t=j}^{i} \phi(t) = \frac{i-j+1}{k} + \frac{\phi(i) - \phi(j-1)}{\alpha + o(1)} = \frac{i-j+1}{k} + \frac{1}{\alpha}(\phi(i) - \phi(j-1)) + o(1).$$

□

[6] Note that in this lemma we are not assuming that i and j are constants.

Lemma 10. *For every $f \in [0,1]$, $\alpha\Phi(f) + h(f) \leq \alpha\Phi(0)f$.*

Proof. Let $W = \alpha\Phi(f) + h(f) - \alpha\Phi(0)f$. We have

$$W = \begin{cases} \alpha(\Phi(f) - \Phi(0))f & f \leq \ell^* \\ \ell(f) - \frac{1}{\alpha}\Phi(0) + (1 + \alpha f - \alpha\ell(f))\Phi(f) - \alpha\Phi(0)f & f \geq \ell^*. \end{cases}$$

For $f \leq \ell^*$, the lemma follows from the fact that $\Phi(.)$ is decreasing. For $f \geq \ell^*$, let $Z := \Phi(f) \in [0, \frac{\Phi(0)}{\alpha}]$. We have $f = 1 + \frac{1}{\alpha}\ln(1 - Z)$ and $\ell(f) = 1 + \frac{1}{\alpha}\ln(1 - \alpha Z)$, and therefore

$$\begin{aligned} W &= 1 + \frac{1}{\alpha}\ln(1 - \alpha Z) - \frac{\Phi(0)}{\alpha} + (1 + \alpha + \ln(1 - Z) - \alpha - \ln(1 - \alpha Z))\, Z \\ &\quad - \alpha\Phi(0)\left(1 + \frac{1}{\alpha}\ln(1 - Z)\right) \end{aligned}$$

Taking the derivative of the above expression with respect to Z, we obtain

$$\begin{aligned} \frac{\partial W}{\partial Z} &= -\frac{1}{1 - \alpha Z} + (1 + \ln(1 - Z) - \ln(1 - \alpha Z)) + \left(\frac{-1}{1 - Z} - \frac{-\alpha}{1 - \alpha Z}\right) Z + \frac{\Phi(0)}{1 - Z} \\ &= 1 - \ln\left(\frac{1 - \alpha Z}{1 - Z}\right) + \frac{\Phi(0) - 1}{1 - Z} \\ &\geq 2 - \frac{1 - \alpha Z}{1 - Z} - \frac{e^{-\alpha}}{1 - Z} \\ &= \frac{(\alpha - 2)Z + 1 - e^{-\alpha}}{1 - Z}, \end{aligned} \tag{38}$$

where the inequality follows from $\ln(x) \leq x - 1$. The value (38) is a monotone function of Z. Therefore, for $Z \in [0, \frac{\Phi(0)}{\alpha}]$, we have

$$\frac{\partial W}{\partial Z} \geq \min\{1 - e^{-\alpha}, \frac{(\alpha - 2)\Phi(0)/\alpha + 1 - e^{-\alpha}}{1 - \Phi(0)/\alpha}\} = \min\{1 - e^{-\alpha}, \frac{(\alpha - 2 + \alpha)\Phi(0)/\alpha}{1 - \Phi(0)/\alpha}\} \geq 0$$

This shows that W is an increasing function of Z. Thus, it takes its maximum at $Z = \Phi(0)/\alpha$. At this point, we have $\ln(1 - \alpha Z) = -\alpha$ and therefore

$$\begin{aligned} W &= -\frac{\Phi(0)}{\alpha} + \left(1 + \ln(1 - \frac{\Phi(0)}{\alpha}) + \alpha\right)\frac{\Phi(0)}{\alpha} - \alpha\Phi(0)\left(1 + \frac{1}{\alpha}\ln(1 - \frac{\Phi(0)}{\alpha})\right) \\ &= \frac{\Phi(0)}{\alpha}\left(\ln(1 - \frac{\Phi(0)}{\alpha}) + \alpha - \alpha^2 - \alpha\ln(1 - \frac{\Phi(0)}{\alpha})\right) \\ &= \Phi(0)(1 - \alpha)\ell^* \\ &\leq 0 \end{aligned}$$

This completes the proof of the lemma. □

Lemma 11. *For any value of $y \geq 0$, the function $f - h(f)y$ is a concave function of f in the range $f \in [\ell^*, 1]$. In particular, we have*

$$\min_{f \in [\ell^*, 1]} \{f - h(f)y\} = \min\{\ell^* - h(\ell^*)y, 1 - h(1)y\}.$$

Proof. All the derivatives in this proof are with respect to f. First observe that

$$\begin{aligned}\Phi'(f) &= -\alpha e^{\alpha(f-1)} \\ \Phi''(f) &= -\alpha^2 e^{\alpha(f-1)} = \alpha\Phi'(f)\end{aligned}$$

We also have

$$\begin{aligned}\ell(f) &= \frac{1}{\alpha}\ln(1-\alpha\Phi(f))+1 \\ \ell'(f) &= \frac{-\Phi'(f)}{1-\alpha\Phi(f)} \\ \ell''(f) &= \frac{-\Phi''(f)}{1-\alpha\Phi(f)} - \frac{\alpha(\Phi'(f))^2}{(1-\alpha\Phi(f))^2} = \alpha\ell'(f) - \alpha(\ell'(f))^2 \end{aligned} \tag{39}$$

And for the second derivative of $f - h(f)y$ we have

$$\begin{aligned}\frac{\partial^2}{\partial f^2}(f-h(f)y) &= -y\left(\ell''(f) + (\alpha(f-\ell(f)-1)+1)\Phi''(f) + 2\alpha(1-\ell'(f))\Phi'(f) - \alpha\ell''(f)\Phi(f)\right) \\ &= -y\left((1-\alpha\Phi(f))\ell''(f) + (\alpha(f-\ell(f)-1)+3-2\ell'(f))\Phi''(f)\right) \end{aligned} \tag{40}$$

On the other hand

$$(1-\alpha\Phi(f))\ell''(f) = -\Phi''(f) - \frac{\alpha(\Phi'(f))^2}{1-\alpha\Phi(f)} = -\Phi''(f) + \frac{-\Phi'(f)}{1-\alpha\Phi(f)}(\alpha\Phi'(f)) = \Phi''(\alpha)(-1+\ell'(f))$$

Plugging into (40), we get

$$\frac{\partial^2}{\partial f^2}(f-h(f)y) = -y\left((\alpha(f-\ell(f)-1)+2-\ell'(f))\Phi''(f)\right)$$

Note that $\Phi''(f)$ is negative. Thus, in order to complete the proof we need to show that

$$T := \alpha(f-\ell(f)-1)+2-\ell'(f) < 0 \tag{41}$$

Taking the first derivative, by (39), we have

$$\begin{aligned}T' &= \alpha(1-\ell'(f)) - \ell''(f) \\ &= \alpha(1-\ell'(f)) - (\alpha\ell'(f) - \alpha(\ell'(f))^2) \\ &= \alpha(1-2\ell'(f) + (\ell'(f))^2) \\ &= \alpha(1-\ell'(f))^2 \\ &> 0\end{aligned}$$

Therefore, T is increasing, and hence its value on $f \in [\ell^*, 1]$ is bounded by its value at 1.

$$T(f=1) = \alpha(1-\ell(1)-1)+2-\ell'(1) = \alpha(1-1-1)+2-\alpha e^{\alpha(1-1)} = 2(1-\alpha) < 0$$

Therefore T is negative for $f \le 1$, which completes the proof. □

Lemma 12. *For any value of $y \geq 0$, the function $f - h(f)y$ is a concave of f on $[0, \max(0, 1-2/\alpha)]$ and a convex function on $[\max(0, 1-2/\alpha), \ell^*]$. The minimum of this function over $f \in [0, \ell^*]$ occurs in one of the three points 0, ℓ^*, or $\mu(y)$, where $\mu(y)$ is the root of the equation $1 - e^{\alpha(x-1)} + \alpha(1-x)e^{\alpha(x-1)} = 1/(\alpha y)$ in the interval $[\max(0, 1-2/\alpha), \ell^*]$, if such a root exists.*

Proof. The second derivative of the function $f - h(f)y$ in $[0, \ell^*]$ can be written as

$$\begin{aligned}
\frac{\partial}{\partial f}(f - h(f)y) &= 1 - \alpha y \left(\Phi(f) + (f-1)\Phi'(f)\right) \\
\frac{\partial^2}{\partial f^2}(f - h(f)y) &= -\alpha y \left(2\Phi'(f) + (f-1)\Phi''(f)\right) \\
&= -\alpha y \left(-2\alpha e^{\alpha(f-1)} - (f-1)\alpha^2 e^{\alpha(f-1)}\right) \\
&= \alpha^3 y e^{\alpha(f-1)} \left(\frac{2}{\alpha} + f - 1\right)
\end{aligned}$$

Clearly, this expression is negative (i.e., the function is concave) for $f \leq 1 - 2/\alpha$ and is positive (i.e., the function is convex) for $f \geq 1 - 2/\alpha$. To obtain the minimum of the function in the interval where it is convex, we set the derivative to zero:

$$\frac{\partial}{\partial f}(f - h(f)y) = 0 \Rightarrow \Phi(f) + (f-1)\Phi'(f) = 1/(\alpha y) \Rightarrow 1 - e^{\alpha(f-1)} - (f-1)\alpha e^{\alpha(f-1)} = 1/(\alpha y)$$

Therefore, f is the root of the equation $1 - e^{\alpha(x-1)} + \alpha(1-x)e^{\alpha(x-1)} = 1/(\alpha y)$. □