

Online Optimization with Uncertain Information*

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

Hamid Nazerzadeh
Stanford University
hamidnz@stanford.edu

Amin Saberi
Stanford University
saberi@stanford.edu

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 was 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 [10, 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 [14] 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 worth of billions of dollars of advertising space next to their search results [15]. 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 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

*An earlier version of this paper was presented in ACM Conference on Electronic Commerce [18].

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 reasonably high 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. [19] 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 Consider two online 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.

One interpretation of the above definition is the following. Algorithm \mathcal{O} is an “optimistic” algorithm that assumes the input sequence arrives as predicted. Algorithm \mathcal{P} is on the other hand “pessimistic” and it makes no assumption about the future. When the estimations of the input are accurate, algorithm \mathcal{H} obtains a value of at least $\gamma_o V_{\mathcal{O}}$. The pessimistic algorithm always obtains a value of $\gamma_p V_{\mathcal{P}}$ even if the estimations turn out to be wrong. Hence, the interesting range of values is $\gamma_o V_{\mathcal{O}} \geq \gamma_p V_{\mathcal{P}}$. We expect that $0 \leq \gamma_o, \gamma_p \leq 1$ and as γ_o increases, γ_p decreases and vice versa.

For the query allocation problem, for example, 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. The following simple algorithm, for $\gamma \in [0, 1]$, is $(\gamma, 1 - \gamma)$ -balanced: 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 (see appendix B). But our main contribution is to improve it using an intuitive interpretation of Mehta et al’s algorithm. Our analysis is based on factor-revealing linear programs [13, 12].

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 use the algorithm of Lenstra et al. [17] for makespan minimization which is a factor 2-approximation algorithm. However,

without prior information, no algorithm can achieve a competitive ratio better than $\Omega(\log n)$ [7]. Aspnes et al. [4] proposed a deterministic algorithm that obtains an $O(\log n)$ -competitive ratio. We use these two algorithms as a black box 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 similar guarantees as 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 first 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 is 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 can not 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 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. As the following example shows, this approach can incur an arbitrary high cost compared to the optimal solution.

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 non-zero 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 in the following. 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} .

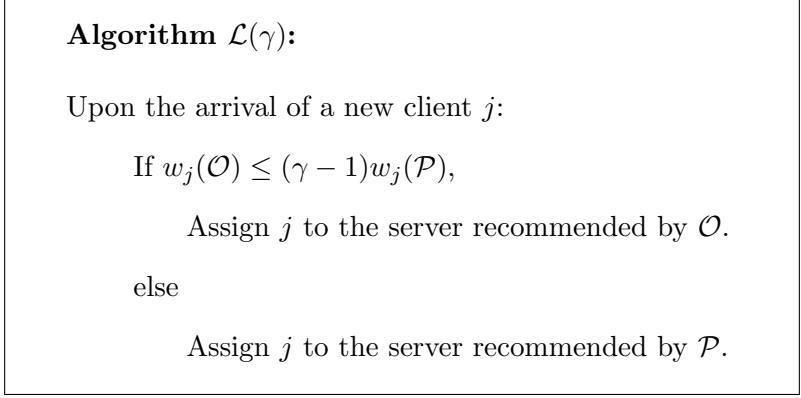


Figure 1: A $(\gamma, \frac{\gamma}{\gamma-1})$ -balanced algorithm for online load balancing.

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. [17] 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} .

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} is bounded by the cost of following recommendations of \mathcal{P} . 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. Without loss of generality, assume that jobs $1, \dots, t$ showed up to time 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.

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:

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

□

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} . Moreover, there exists no deterministic (γ, γ') -balanced algorithm such that $\gamma' > \frac{\gamma}{\gamma-1}$.

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 shows up, its loads matches the estimation. However, \mathcal{P} recommends the second server for the job. Suppose \mathcal{A} is an unreliability-balanced algorithm. Consider the following cases.

1. \mathcal{A} follows the recommendation of \mathcal{O} and allocates the first job to the first server. However, when the second job shows up, we observe $l_{12} = 1$ and $l_{22} = \infty$. 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 recommendation of \mathcal{O} and allocates the first job to the second server. Then, the second job shows up, and its loads matches the estimation. In this case, the cost of \mathcal{A} is at least γ while the cost of the solution recommended by \mathcal{O} is $\gamma - 1$.

Therefore, there exists no deterministic (γ, γ') -balanced algorithm such that $\gamma' > \frac{\gamma}{\gamma-1}$. □

3 Online Resource Allocation

In this section, we present a family of unreliability-balanced 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 [20].

Online Facility Location A set of facilities is given. A sequence of clients arrives in an online fashion. Once a new client shows up, the algorithm should take one of the following actions: i) Open a facility 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. This problem generalizes the online set cover problem. However, when the connection costs are metric, better competitive ratio is achievable. Fotakis [11] showed that the worst-case competitive of the online metric facility location problem is $\theta(\frac{\log n}{\log \log n})$. On the flip side, if the connection costs are known in advance, one can use the algorithm of Byrka [9] 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, Awerbuch et al. [5] proposed a $O(\log^2 n)$ -competitive algorithm, and gave a lower bound of $\Omega(\log n)$ on the competitiveness of any online algorithm. However, with accurate estimates of the input sequence, one can solve the problem within a constant factor of the optimal solution, see [20].

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

Definition 2 (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. Each job j can be served by a family of subsets of \mathcal{S} denoted by S_j . The cost of serving job j using the sets in S_j is given by function $d_j : S_j \rightarrow R^+$. Upon the arrival of a new job j , it should be assigned to a set in S_j . Also, there is another constraint that servers should be activated before starting serving any job. Activating server $i \in \mathcal{S}$ incurs a cost of $c_i \geq 0$. 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 the jobs to another set of servers or deactivate a server.*

In the set cover problem, elements correspond to jobs and subsets in \mathcal{S} correspond to servers. Activating cost c_i 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 is equal to the set of all facilities. 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 connects 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. The advantage of our algorithm is that one can still enjoy the significant improvements, while at the same time maintains an optimal (up to a constant) worst-case competitive ratio.

Now let us describe the algorithm. Without loss of generality, assume that jobs $1, \dots, t$ showed up to time t . Let $w_t(\mathcal{H}(\gamma))$ be the total cost of algorithm $\mathcal{H}(\gamma)$ up to time t , i.e.,

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

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.

where $s_j \in S_j$ denotes the set that the algorithm assigns to job t . Also, let $w_t(\mathcal{P})$ be the cost of the solution one obtains by following all the recommendations of \mathcal{P} . Similarly, $w_t(\mathcal{O})$ is defined as the cost of the solution provided by \mathcal{O} . The algorithm 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} . Moreover, there exists deterministic (γ, γ') -balanced algorithm such that $\gamma' > \frac{\gamma}{\gamma-1}$.*

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. [19] as a generalized online matching). 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 [19], we assume that bids are *arbitrary 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. 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:

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

Note that the assumption that bids are small compared to 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. [19] 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 parametrized by $\gamma \in [0, 1]$ that are $(\gamma, 1 - \gamma)$ -balanced with respect to given algorithms \mathcal{P} and \mathcal{O} . However, in this section, we take a slightly different approach. Using techniques from the competitive analysis of the problem, we design an algorithm with a better competitive ratio. 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. Our algorithm is balanced with respect to \mathcal{O} and the *optimal offline solution*, which knows the sequence of queries in advance.

We first briefly present the $(1 - 1/e)$ -competitive algorithm of Mehta et al. Their algorithm multiplies the bid of each advertiser by a factor $(1 - e^{f-1})$ where f is the fraction of the budget of an advertiser that is spent, i.e., the algorithm discounts the bid according to the spent budget. 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 $\Phi_\alpha(f) := 1 - e^{\alpha(f-1)}$. 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 [19, 8]. The following is the dual of the linear program for the query allocation problem. Without loss of generality, we assume that all n_j 's are equal to 1.

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 unreliability balanced algorithm for the query allocation problem.

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

The algorithm of Mehta et al. 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 multiplicatively: $\beta_i \leftarrow \beta_i(1 + \alpha \frac{b_{ij}}{B_i})$.

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 immediately from Lemmas 7 and 9 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 $\gamma_{\text{OPT}} = \frac{1}{\alpha}(1 - \frac{1}{e^\alpha})$ and $\gamma_{\mathcal{O}}$ is defined as follows. Let α^* be the root of equation $(\alpha^2 - 1/\alpha)e^{-\alpha} + 1/\alpha - 1 = 0$ in $[1, 2]$.¹ For $\alpha \geq \alpha^*$,*

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

¹ α^* is very close to 1.795.

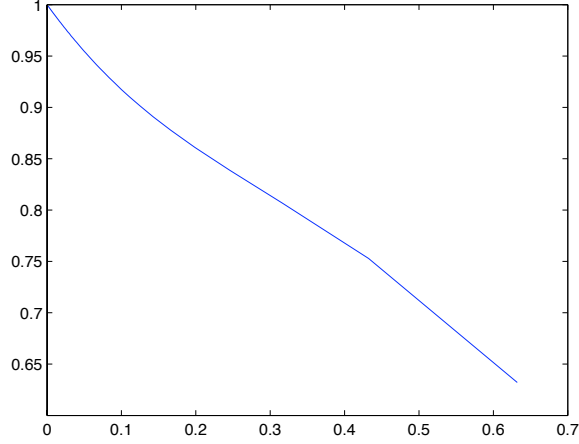


Figure 4: The trade off between γ_{OPT} (x -axis) and $\gamma_{\mathcal{O}}$ (y -axis) for the query allocation problem. γ_{OPT} is the worst-case competitive ratio of algorithm $\mathcal{Q}(\alpha)$, while $\gamma_{\mathcal{O}}$ is the competitive ratio when the estimates are accurate.

Also, let $f^* \in [0, 1]$ be the solution of the equation $(\alpha(f-1))^2 e^{\alpha(f-1)} = 1 - \frac{1}{\alpha}(1 - e^{-\alpha})$. For α in $(1, \alpha^*)$, we have

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

Moreover, the analysis is tight.

Similar to the previous sections, one can interpret the theorem above as a family of balanced algorithms that are parametrized by $\gamma \in [0, 1]$ such that $\gamma = \gamma_{\text{OPT}}$. Figure 4 depicts the value of $\gamma_{\mathcal{O}}$ with respect to γ_{OPT} . Note that the maximum possible value for γ_{OPT} is $1 - \frac{1}{e}$.

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 [19] 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 $\text{slab}(i)$ be equal to $\lfloor \frac{f_i}{k} \rfloor + 1$, i.e., $\text{slab}(i)$ denotes the active slab of the budget of i . Also, let

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

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 , $1 \leq t \leq k$, if 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 at the end of the algorithm. Also, let r_t to be the total remaining budget of these advertisers with respect to an optimal solution OPT . As in [19], for simplicity, we assume that advertisers of type t have spent exactly $\frac{t}{k}$ fraction of their budgets. Also, we assume that no query straddle a slab. If $\max\{\frac{b_{ij}}{B_i}\} \leq \frac{1}{k^2}$, then total error due to this assumption is $O(\frac{|A|}{k})$ which is negligible. For $j = 0, 1, \dots, k$, define w_j to be the total amount of money spent from slab j of the budget of the advertisers. Note that $w_i = \frac{1}{k} \sum_{j=i}^k s_j$. Also, observe that the revenue of OPT is $\sum_{i=1}^k (s_i - r_i)$. 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 lemmas below.

Lemma 6 *At the end of the algorithm, we have:*

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

Proof : Suppose OPT allocates query j 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 j . By the rule of the algorithm we have,

$$\phi(g_i) b_{ij} \leq \alpha \phi(g_a) b_{aj}$$

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

$$\phi(t_i) b_{ij} \leq \phi(g_i) b_{ij} \leq \alpha \phi(g_a) b_{aj}$$

Summing up this inequality over all queries, we get

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

Observe that $\phi(t_i) b_{ij}$ contributes to the l.h.s and $\phi(g_a) b_{aj}$ contributes to the r.h.s of the inequality above. \square

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 $\frac{1}{\alpha}(1 - \frac{1}{e^\alpha})$.*

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

$$\sum_{i=1}^k \phi(i)(s_i - r_i) \leq \alpha \sum_{j=1}^k \phi(j) \left(\frac{1}{k} \sum_{i=j}^k s_i \right) = \frac{\alpha}{k} \sum_{i=1}^k s_i \sum_{j=1}^i \phi(j) \quad (8)$$

In Lemma 17 in the appendix, we prove the following inequality.

$$\lim_{k \rightarrow \infty} \frac{1}{k} \sum_{t=j}^i \phi(t) = \frac{i-j+1}{k} + \frac{1}{\alpha} (\phi(i) - \phi(j-1)) \quad (9)$$

Plugging to the inequality above we get,

$$\sum_{i=1}^k \phi(i)(s_i - r_i) \leq \sum_{i=1}^k \left(\alpha \frac{i}{k} + \phi(i) - \phi(0) \right) s_i \quad (10)$$

Rearranging the terms gives us

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

ϕ is decreasing. Therefore,

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

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. \square

Tight Example Consider k advertisers each with budget $\frac{1}{k}$. A sequence of keywords shows up in k phases. In phase i , $\frac{1}{k\alpha\epsilon}$ keywords arrive for which advertiser i bids slightly less than $\alpha\epsilon$, and other advertisers bid ϵ . The algorithm, based on recommendations of \mathcal{O} , distributes these keywords among advertisers i, \dots, k , such that the fraction of the spent budget of these advertisers remains equal. Let $k \rightarrow \infty$. Define ℓ to be the phase in which the budget of advertiser k is exhausted. At 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 $\frac{1}{k-i} \times \frac{1}{\alpha k}$ of the budget of every advertiser $j > i$ is spent. We can approximate $\sum_{j=1}^i \frac{1}{k-j} \frac{1}{\alpha k}$ with $\frac{1}{\alpha k} \int_{1-(i/k)}^1 \frac{1}{x} = \frac{1}{\alpha k} \ln \frac{1}{1-(i/k)}$ with arbitrary high accuracy. Hence, at the end of phase i , an amount of $\frac{1}{\alpha k} \ln \frac{1}{1-(i/k)}$ of the budget of each advertiser $j > i$ is spent. Therefore, by definition of ℓ we have $\frac{\ell}{k} = 1 - e^{-\alpha}$. Also, the revenue of algorithm \mathcal{Q} is equal to $\ell \cdot \frac{1}{k} = 1 - e^{-\alpha}$. 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 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 . 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.

We now analyze $\mathcal{Q}_k(\alpha)$. Recall that each keyword j is allocated to the advertiser recommended by \mathcal{O} , denoted by o , if $\alpha\phi_k(f_o)b_{oj} \geq \phi_k(f_p)b_{pj}$, where $p \in \operatorname{argmax}_{i \in \mathcal{A}} \{\phi_k(f_i)b_{ij}\}$. Observe that for every keyword $j \in C$:

$$\phi_k(f_o)b_{oj} < \frac{1}{\alpha} \phi_k(f_p)b_{pj}$$

Summing up over all queries in C , similar to Lemma 6, we get:

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

With algebraic manipulation we get the inequality below, the proof is appeared in the appendix.

Lemma 8 *For algorithm $\mathcal{Q}_k(\alpha)$, at the end of the algorithm we have,*

$$\sum_{i=1}^k \sum_{j=1}^i (\Phi(j) - \alpha\Phi(i))t_{ij} < \sum_{i=1}^k \left(\frac{i}{k} - \frac{1}{\alpha}\Phi(0) + \left(\frac{1}{\alpha} - \alpha\right)\Phi(i)\right)s_i + \alpha \sum_{i=1}^k \Phi(i)r_i$$

To compute $\gamma_{\mathcal{O}}$ we use the factor revealing LP technique [13, 12]. Consider the LP below. The objective function is the revenue of the algorithm and the linear constraints are imposed by the algorithm and the structure of the problem. Therefore, any feasible solution of this LP gives an upper bound on the competitive ratio of the algorithm.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^k \frac{i}{k} s_i \\ \text{subject to:} \quad & \sum_{i=1}^k s_i = 1 \\ & \sum_{i=1}^k \left(\frac{i}{k} - \frac{1}{\alpha}\Phi(0) + \left(\frac{1}{\alpha} - \alpha\right)\Phi(i)\right)s_i + \alpha \sum_{i=1}^k \Phi(i)r_i \geq \sum_{i=1}^k \sum_{j=1}^i (\Phi(j) - \alpha\Phi(i))t_{ij} \quad (12) \\ & \frac{1}{k}s_i \geq t_{ij} \quad 1 \leq j \leq i \leq k \\ & s_i \geq r_i \quad 1 \leq i \leq k \\ & s_i, r_i, t_{ij} \geq 0 \quad 1 \leq j \leq i \leq k \end{aligned}$$

Here is the dual of this linear program. Variables x , y , z , and u correspond respectively to the first, second, third and fourth set of primal constraints.

$$\begin{aligned} \text{maximize} \quad & x \\ \text{subject to:} \quad & x + \left(\frac{i}{k} - \frac{1}{\alpha}\phi(0) + \left(\frac{1}{\alpha} - \alpha\right)\phi(i)\right)y + \frac{1}{k} \sum_{j=1}^i z_{ij} + u_i \leq \frac{i}{k} \quad 1 \leq i \leq k \\ & y\alpha\phi(i) \leq u_i \quad 1 \leq i \leq k \quad (13) \\ & (\alpha\phi(i) - \phi(j))y \leq z_{ij} \quad 1 \leq j \leq i \leq k \\ & y, z_{ij}, u_i \geq 0, \end{aligned}$$

In the next lemma, we find a feasible solution for the dual LP.

Lemma 9 Let α^* be the root of $(\alpha^2 - 1/\alpha)e^{-\alpha} + 1/\alpha - 1 = 0$ in $[1, 2]$. For $\alpha \geq \alpha^*$, let

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

For α in $(1, \alpha^*)$, let

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

where $f^* \in [0, 1]$ is the solution for the equation $(\alpha(f - 1))^2 e^{\alpha(f-1)} = 1 - \frac{1}{\alpha}(1 - e^{-\alpha})$. Then on any input, the revenue of algorithm $\mathcal{Q}(\alpha)$ is at least a $\gamma_{\mathcal{O}}$ fraction of the revenue of the solution provided by \mathcal{O} .

Proof : We construct a feasible solution for the dual linear program. Let $u_i = y\alpha\phi(i)$, and $z_{ij} = \max\{(\alpha\phi(i) - \phi(j))y, 0\}$, for all $1 \leq j \leq i \leq k$. Now, the only remaining constraints are

$$x + \left(\frac{i}{k} - \frac{1}{\alpha}\phi(0) + \left(\frac{1}{\alpha} - \alpha\right)\phi(i)\right)y + \frac{1}{k} \sum_{j=1}^i z_{ij} + u_i \leq \frac{i}{k} \quad (14)$$

For i , $1 \leq i \leq k$, let $j^* = \min_{j \geq 0}\{\alpha\phi(i) - \phi(j) < 0\}$. Therefore, by (9), we have:

$$\frac{1}{k} \sum_{j=1}^i z_{ij} \geq \frac{1}{k} \sum_{j=j^*+1}^i (\alpha\phi(i) - \phi(j))y = \left(\alpha \frac{i - j^*}{k}\right)\phi(i) - \frac{i - j^*}{k} - \frac{1}{\alpha}(\phi(i) - \phi(j^*))y \quad (15)$$

Let k tend to infinity and $f = \frac{i}{k}$. Recall that $\Phi(f) = 1 - e^{\alpha(f-1)}$. If $\alpha\Phi(f) \geq \Phi(0)$, then $j^* = 0$. Otherwise, we have $\Phi(\frac{j^*}{k}) = \alpha\Phi(f)$, i.e., $\frac{j^*}{k} = \frac{1}{\alpha} \ln(1 - \alpha(1 - e^{\alpha(f-1)})) + 1$. Also, let $\ell = \ln(1 - \frac{\Phi(0)}{\alpha}) + 1$, i.e., $\alpha\Phi(\ell) = \Phi(0)$. We have,

$$\frac{1}{k} \sum_{j=1}^i z_{ij} \geq \begin{cases} (\alpha f \Phi(f) - f - \frac{1}{\alpha}(\Phi(i) - \Phi(0)))y & f \geq \ell \\ ((\alpha f - \ln(1 - \alpha(1 - e^{\alpha(f-1)})) + \alpha)\Phi(f) - & \text{o.w.} \\ f + \frac{1}{\alpha} \ln(1 - \alpha(1 - e^{\alpha(f-1)})) + 1 - \frac{1}{\alpha}\Phi(f) + \Phi(f))y & \end{cases} \quad (16)$$

Plugging into (14), we get inequalities below:

$$x \leq \begin{cases} g_1(f, y) & f \geq \ell \\ g_2(f, y) & \text{o.w.} \end{cases} \quad (17)$$

where,

$$\begin{aligned} g_1(f, y) &= f - \alpha(f - 1)\Phi(f)y \\ g_2(f, y) &= f - \left((\alpha(f - 1) + 1)\Phi(f) - \frac{1}{\alpha}\Phi(0) + \left(\frac{1}{\alpha} \ln(1 - \alpha\Phi(f)) + 1\right)(1 - \alpha\Phi(f)) \right) y \end{aligned}$$

Now the dual linear program is converted to a continuous optimization problem. We compute the optimal solution by taking the derivatives of the r.h.s of (17) and applying the first and second

order conditions. We first prove the theorem for $\alpha \geq \alpha^*$. Let y^* be the solution of $g_1(0, y) = g_2(1, y)$, i.e.,

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

We show that g_1 takes its minimum in interval $[0, \ell]$ at $f = 0$, and the minimum of g_2 between ℓ and 1 is at $f = 1$. Therefore, we have

$$x = g_2(1, y^*) = g_1(0, y^*) = \alpha\Phi(0)y^* = \frac{\alpha(e^\alpha - 1)}{(\alpha - \frac{1}{\alpha})(e^\alpha - 1) + e^\alpha}$$

To prove the claim, first observe that

$$\begin{aligned} \frac{\partial g_1}{\partial f} &= 1 - \alpha y + \alpha(\alpha(f - 1) + 1)e^{\alpha(f-1)}y \\ \frac{\partial g_1}{\partial^2 f} &= \alpha^2(\alpha f - \alpha + 2)ye^{\alpha(f-1)} \end{aligned}$$

For y^* and $\alpha \geq \alpha^*$, function g_1 is increasing at $f = 0$. Also, g_1 is strictly concave before $1 - \frac{2}{\alpha}$ and strictly convex after that. Therefore, g_1 is strictly increasing in the interval $[0, \ell]$. Also, as proved in Lemma 19 in the appendix, function g_2 is strictly concave in the interval $[\ell, 1]$. Because $g_2(1, y^*) = g_1(0, y^*) \leq g_1(\ell, y^*) = g_2(\ell, y^*)$, g_2 is strictly decreasing in the interval $[\ell, 1]$. Therefore, we have $x = g_1(0, y^*) = g_2(1, y^*)$.

For $\alpha < \alpha^*$, function g_1 is decreasing at $f = 0$. In this case, g_1 takes its the minimum at f^* which makes the derivative of g_1 zero, i.e.,

$$1 - \alpha(1 - (\alpha(f^* - 1) + 1)e^{\alpha(f^*-1)})y = 0 \quad (18)$$

Similar to the previous case, g_2 takes its minimum in interval $[\ell, 1]$ at $f = 1$. Namely,

$$x = f^* - \alpha(f^* - 1)\Phi(f^*)y = 1 - (1 - \frac{1}{\alpha}\Phi(0))y \quad (19)$$

The value of f^* is computed by solving the equation above. Plugging the value of y from (18) we have

$$x = 1 - \frac{1 - \frac{1}{\alpha}(1 - e^{-\alpha})}{\alpha(1 - (1 + \alpha(f^* - 1))e^{\alpha(f^*-1)})}$$

□

Figure 5 depicts the r.h.s of inequality (17) for different values of α .

Tight Example We first give a tight example for this analysis for $\alpha \geq \alpha^*$. Suppose there are two advertisers with budgets $B_1 = \frac{1}{\alpha\phi(0)} \int_0^1 \Phi(f)df = \frac{1}{\alpha\Phi(0)}(1 - \frac{1}{\alpha}(1 - e^\Phi(0)))$ and $B_2 = 1$. First, 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 . Although \mathcal{O} recommends advertiser 1 for the query, the algorithm allocates all these queries to the second advertiser. This continues up to the time advertiser 2 runs out of budget. After that, a set of keywords shows up that only 2 is interested in but she has no budget left. The total revenue of \mathcal{O} is equal to $\frac{1}{\alpha\phi(0)} \int_0^1 \phi(f)df + 1 = B_1 + B_2$. The total revenue of the algorithm 1. Therefore, we have

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

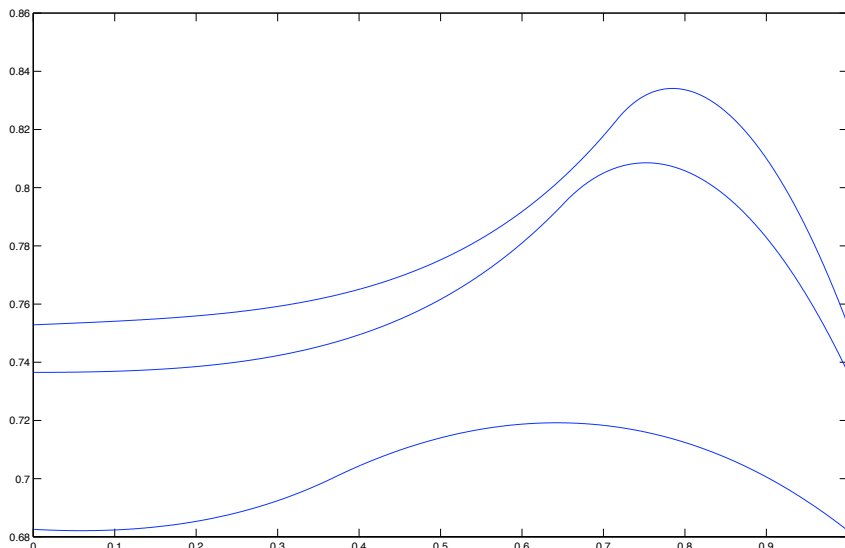


Figure 5: The r.h.s of inequality (17). For α equal to 2 (top), α^* (middle) and 1.3 (bottom). The value of γ is chosen as suggested in the proof of the lemma.

Also, this example corresponds to a feasible primal solution with $s_0 = B_1$, $s_1 = B_2$, and all other variables are 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 fraction f^* of her budget. After that, the second series 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. The rest of the example and its analysis is similar to the previous case.

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] 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.
- [9] Jaroslaw Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In *APPROX-RANDOM*, pages 29–43, 2007.
- [10] Amos Fiat and Gerhard Woeginger. *Online algorithms: The state of the art*. Springer-Verlag, 1998.
- [11] Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.
- [12] 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.
- [13] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *STOC*, pages 731–740, 2002.
- [14] Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. In *FOCS*, pages 394–400. IEEE, 1994.
- [15] 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.
- [16] Benny Lehmann Daniel Lehmann and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 52(2), 2006.
- [17] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.
- [18] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *ACM Conference on Electronic Commerce*, pages 288–294, 2007.

- [19] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5), 2007.
- [20] Vijay Vazirani. *Approximation algorithms*. Springer-Verlag, Berlin, 2001.

A Proof of Theorem 4

The proof follows immediately from lemmas below.

Lemma 10 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, $w_t(\mathcal{H}(\gamma)) \leq \gamma w_t(\mathcal{P})$. \square

Lemma 11 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, $w_t(\mathcal{H}(\gamma)) \leq (1 + \frac{1}{\gamma-1}) w_t(\mathcal{O}) \leq \frac{\gamma}{\gamma-1} w_t(\mathcal{O})$. \square

Lemma 12 For the ORA problem, there is no deterministic (γ, γ') -balanced algorithm such that $\gamma' > \frac{\gamma}{\gamma-1}$.

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. Suppose \mathcal{A} is an unreliability-balanced algorithm. Consider the following cases.

1. \mathcal{A} follows the recommendations of \mathcal{O} and chooses the first subset. However, the second element turns out to 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 γ while the cost of the solution recommended by \mathcal{O} is $\gamma - 1$.

Therefore, there is no deterministic (γ, γ') -balanced algorithm such that $\gamma' > \frac{\gamma}{\gamma-1}$. \square

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 6: 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 6. 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 [16]. Let ε be the maximum ratio of a bid to the budget of an advertiser.

Lemma 13 $V_{\mathcal{G}(\gamma)} \geq \frac{1}{1 + \alpha(1 + \varepsilon)} V_{\mathcal{P}}$

Proof : For each query j we consider the following two cases:

1. The algorithm allocates j to the advertiser recommended by \mathcal{P} . Observe that the total contribution of all such queries to $V_{\mathcal{P}}$ is at most $V_{\mathcal{G}(\gamma)}$, because we cannot charge an advertiser more than her budget.
2. Otherwise, by the rule of the algorithm, the sum of the bids of all advertisers that received the queries despite to the recommendation of \mathcal{P} is at least $\frac{1}{\alpha} V_{\mathcal{P}}$. On the other hand, the sum of the bids of these advertisers is at most $(1 + \varepsilon) V_{\mathcal{G}(\gamma)}$. Therefore, we have $\frac{1}{\alpha} V_{\mathcal{P}} \leq (1 + \varepsilon) V_{\mathcal{G}(\gamma)}$.

Summing up the inequalities above we have $V_{\mathcal{P}} \leq (1 + \alpha(1 + \varepsilon)) V_{\mathcal{G}(\gamma)}$ □

Lemma 14 $V_{\mathcal{G}(\gamma)} \geq \frac{\alpha}{1 + \alpha} V_{\mathcal{O}}$

Proof : Let s be the total value of the queries that $\mathcal{G}(\gamma)$ does not allocate to the advertiser recommended by \mathcal{O} . The revenue could be obtained from this set of queries by following the recommendation is at most $\frac{1}{\alpha}s$. Therefore, the increase in the revenue by following the recommendations of \mathcal{O} is at most $\frac{1}{\alpha}V_{\mathcal{G}(\gamma)}$. \square

Lemma aboves immediately lead to the following theorem.

Theorem 15 For $0 \leq \gamma \leq 1$, and $\alpha = \frac{1}{\gamma} - 1$ we have

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

Corollary 16 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 17 $\lim_{k \rightarrow \infty} \frac{1}{k} \sum_{t=j}^i \phi(t) = \frac{i-j+1}{k} + \frac{1}{\alpha}(\phi(i) - \phi(j-1))$

Proof : Plugging (6) we get,

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

Substituting the geometric sum,

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

Now note that $\lim_{k \leftarrow \infty} k(1 - \left(1 - \frac{1}{k}\right)^{\alpha}) = \alpha$. Therefore, dividing both sides of the inequality above with $\frac{1}{k}$, we have,

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

\square

Lemma 18 For algorithm $\mathcal{Q}_k(\alpha)$, at the end of the algorithm we have,

$$\sum_{i=1}^k \sum_{j=1}^i (\Phi(j) - \alpha\Phi(i))t_{ij} < \sum_{i=1}^k \left(\frac{i}{k} - \frac{1}{\alpha}\Phi(0) + \left(\frac{1}{\alpha} - \alpha\right)\Phi(i)\right)s_i + \alpha \sum_{i=1}^k \Phi(i)r_i$$

Proof : From (11) we have:

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

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

$$\frac{1}{\alpha} \sum_{i=1}^k \Phi(i) \sum_{j=i}^k t_{ji} - \sum_{i=1}^k \Phi(i) \sum_{j=1}^i t_{ij} \leq \frac{1}{\alpha} \sum_{i=1}^k \Phi(i) \left(\frac{1}{k} \sum_{j=i}^k s_j \right) - \sum_{i=1}^k \Phi(i) (s_i - r_i) \quad (20)$$

For the l.h.s of (20) we have:

$$l.h.s = \frac{1}{\alpha} \sum_{i=1}^k \Phi(i) \sum_{j=i}^k t_{ji} - \sum_{i=1}^k \Phi(i) \sum_{j=1}^i t_{ij} = \frac{1}{\alpha} \left(\sum_{i=1}^k \sum_{j=1}^i (\Phi(j) - \alpha \Phi(i)) t_{ij} \right) \quad (21)$$

For the l.h.s of (20) we have:

$$\begin{aligned} r.h.s &= \frac{1}{\alpha k} \sum_{i=1}^k \Phi(i) \sum_{j=i}^k s_j - \sum_{i=1}^k \Phi(i) s_i + \sum_{i=1}^k \Phi(i) r_i \\ &= \frac{1}{\alpha} \sum_{i=1}^k \left(\frac{1}{k} \sum_{j=1}^i \Phi(j) - \alpha \Phi(i) \right) s_i + \sum_{i=1}^k \Phi(i) r_i \\ &= \left(\frac{1}{\alpha} \sum_{i=1}^k \left(\frac{i}{k} - \frac{1}{\alpha} \phi(i) - \frac{1}{\alpha} \phi(0) \right) - \alpha \phi(i) \right) s_i + \sum_{i=1}^k \Phi(i) r_i \end{aligned} \quad (22)$$

From (21) and (22) the lemma follows. \square

Lemma 19 *Function g_2 is strictly concave in the interval $[\ell, 1]$.*

Proof : We start by taking the first derivative with respect to f .

$$\begin{aligned} \frac{\partial g_2}{\partial f} &= 1 - (\alpha \Phi(f) + (\alpha(f-1) + 1) \Phi'(f) - \Phi'(f) - \left(\frac{1}{\alpha} \ln(1 - \alpha \Phi(f)) + 1 \right) \alpha \Phi'(f)) y \\ &= 1 - (\alpha \Phi(f) + (\alpha f - \ln(1 - \alpha \Phi(f)) - 2\alpha) \Phi'(f)) y \end{aligned}$$

Therefore, for the second derivative we have:

$$\begin{aligned} \frac{\partial g_2}{\partial f} &= -y (\alpha \Phi'(f) + \left(\alpha + \frac{\alpha \Phi'(f)}{1 - \alpha \Phi(f)} \right) \Phi'(f) + (\alpha f - \ln(1 - \alpha \Phi(f)) - 2\alpha) \Phi''(f)) \\ &= y \alpha^2 e^{\alpha(f-1)} \left(1 + \left(1 - \frac{\alpha e^{\alpha(f-1)}}{1 - \alpha + \alpha e^{\alpha(f-1)}} \right) + (\alpha f - \ln(1 - \alpha \Phi(f)) - 2\alpha) \right) \\ &= y \alpha^2 e^{\alpha(f-1)} \left(1 - 2\alpha + \alpha f + \frac{1 - \alpha}{1 - \alpha \Phi(f)} - \ln(1 - \alpha \Phi(f)) \right) \end{aligned} \quad (23)$$

Now consider function $1 - 2\alpha + \alpha f + \frac{1 - \alpha}{1 - \alpha \Phi(f)} - \ln(1 - \alpha \Phi(f))$. We show that this function is increasing and hence take its maximum at $f = 1$.

$$\begin{aligned}
(1 - 2\alpha + \alpha f + \frac{1 - \alpha}{1 - \alpha\Phi(f)} - \ln(1 - \alpha\Phi(f)))' &= \alpha + (\frac{1 - \alpha}{1 - \alpha\Phi(f)} - 1) \frac{\alpha^2 e^{\alpha(f-1)}}{1 - \alpha\Phi(f)} \\
&= \alpha + \frac{\alpha(\Phi(f) - 1)}{1 - \alpha\Phi(f)} \frac{\alpha^2 e^{\alpha(f-1)}}{1 - \alpha\Phi(f)} \\
&= \alpha(1 - \left(\frac{\alpha e^{\alpha(f-1)}}{1 - \alpha\Phi(f)}\right)^2) \\
&= \alpha(1 - \left(1 - \frac{1 - \alpha}{1 - \alpha\Phi(f)}\right)^2) \\
&\geq 0
\end{aligned}$$

For $f = 1$ the value of (23) is negative for $\alpha > 1$. Therefore, the lemma follows. \square