

# Online Advertisement, Optimization and Stochastic Networks

Bo (Rambo) Tan and R. Srikant

Department of Electrical and Computer Engineering  
University of Illinois at Urbana-Champaign  
Urbana, IL, USA

## Abstract

In this paper, we propose a stochastic model to describe how search service providers charge client companies based on users' queries for the keywords related to these companies' ads by using certain advertisement assignment strategies. We formulate an optimization problem to maximize the long-term average revenue for the service provider under each client's long-term average budget constraint, and design an online algorithm which captures the stochastic properties of users' queries and click-through behaviors. We solve the optimization problem by making connections to scheduling problems in wireless networks, queueing theory and stochastic networks. Unlike prior models, we do not assume that the number of query arrivals is known. Due to the stochastic nature of the arrival process considered here, either temporary "free" service, i.e., service above the specified budget (which we call "overdraft") or under-utilization of the budget (which we call "underdraft") is unavoidable. We prove that our online algorithm can achieve a revenue that is within  $O(\epsilon)$  of the optimal revenue while ensuring that the overdraft or underdraft is  $O(1/\epsilon)$ , where  $\epsilon$  can be arbitrarily small. With a view towards practice, we can show that one can always operate strictly under the budget. In addition, we extend our results to a click-through rate maximization model, and also show how our algorithm can be modified to handle non-stationary query arrival processes and clients with short-term contracts.

Our algorithm also allows us to quantify the effect of errors in click-through rate estimation on the achieved revenue. We show that we lose at most  $\frac{\Delta}{1+\Delta}$  fraction of the revenue if  $\Delta$  is the relative error in click-through rate estimation.

We also show that in the long run, an expected overdraft level of  $\Omega(\log(1/\epsilon))$  is unavoidable (a universal lower bound) under any stationary ad assignment algorithm which achieves a long-term average revenue within  $O(\epsilon)$  of the offline optimum.

## I. INTRODUCTION

Providing online advertising services has been the major source of revenue for search service providers such as Google, Yahoo and Microsoft. When an Internet user queries a keyword, alongside the search results, the search engine may also display advertisements from some companies which provide services or goods related to this keyword. These companies pay the search service providers for posting their ads with a specified amount of price for each ad on a pay-per-impression or pay-per-click basis. We call them "clients" in the following text.

Maximizing the revenue obtained from their clients is the key objective of search service providers. Research which targets this objective has followed two major directions. One is based on auction theory, in which the goal is to design mechanisms in favour of the service provider, and much of the research in this direction considers static bids (e.g. [13]; see [10] for a survey), while dynamic models such the one in [22] are still emerging. The other is from the perspective of online resource allocation without considering the impact of the service provider's mechanisms on the clients' bids, and the main focus of this kind of research is on designing an online algorithm which posts specific ads in response to each search query arriving online, in order to

achieve a high competitive ratio with respect to the offline optimal revenue. Our work follows the second direction.

Our model is as follows:

---

### Online Advertising Model:

Assume that queries for keyword  $q$  arrive to the search engine according to a stochastic process at rate  $\nu_q$  queries per time slot, where we have assumed that time is discrete and a “time slot” is our smallest discrete time unit. In response to each query arrival, the search engine may display ads from some clients on the webpage. There are  $L$  different places (e.g., top, bottom, left, right, etc.) on a webpage where ads could be displayed. We will call these places “webpage slots.” When client  $i$ ’s ad is displayed in webpage slot  $s$  when keyword  $q$  is queried, there is a probability with which the user who is viewing the page (the one who generated the query) will click on the ad. This probability, called the “click-through rate,” is denoted by  $c_{qis}$ .

A client specifies the amount of money (“bid”) that it is willing to pay to the search service provider when a user clicks on its ad related to a specific query. We use  $r_{qi}$  to denote this per-click payment from client  $i$  for its ad related to a query for keyword  $q$ . Additionally, client  $i$  also specifies an average budget  $b_i$  which is the maximum amount that it is willing to pay per “budgeting cycle” on average, where a budgeting cycle equals to  $N$  time slots (we have introduced the notion of a budgeting cycle since the time-scale over which queries arrive may be different than the time-scales over which budgets may be settled).

The problem faced by the search service provider is then to assign advertisements to webpage slots, in response to each query, so that its long-term average revenue is maximized.

---

Based on the above model, we design an online algorithm which achieves a long-term average revenue within  $O(\epsilon)$  of the offline optimal revenue, where  $\epsilon$  can be chosen arbitrarily small, indicating the near-optimality of our online algorithm. Before entering into the details, in the next two subsections we will first survey the related literature, highlight the main contributions of our work, and discuss the differences between our model and previous ones.

#### A. Related Work

We will only survey the online resource allocation models here, and not the auction models. The online ads model in prior literature mainly include two types, namely AdWords (AW) and Display Ads (DA), of which the difference lies in the constrained resource of each client. In the AW model, the resource is the client’s budget, while in the DA model, the resource is the maximum number of impressions agreed on by the client and the service provider. Correspondingly, after each resource allocation step, the resource of a client whose ad is posted, is reduced by the bid value<sup>1</sup> in the AW model, or 1 impression in the DA model. Both of them belong to a general class of packing linear programs formulated in [8]. Most of the prior online algorithms for solving the AW and DA model respect the hard constraint on the client’s resources. One exception is [9], where the authors argue that “free disposal” of resources makes the DA model more tractable (but not necessary for the AW model).

Mehta et al. [20] modeled the online ads problem as a generalization of an online matching problem [16] on a bipartite graph of queries and clients. Later in [5], Buchbinder et al. showed that matching clients to webpage slots (whether it is a single slot or multiple slots) can be solved

<sup>1</sup>This refers to the pay-per-impression scheme. With a pay-per-click scheme, the reduction only happens if the ad is clicked.

as a maximum-weighted matching problem. Following [5], a number of other online algorithms using the maximum-weighted bipartite matching idea have been proposed in [19], [9], [6] and [8]. The algorithms in [15] and [20], which were earlier than [5], can also be regarded as maximum-weighted matching solutions on this bipartite graph of clients and webpage slots.

In [15], the “b-matching” problem (related to the online ads context, bids are trivially 0 or 1 and budgets are all  $b$ ) is solved by an  $1 - 1/e$  competitive algorithm as  $b \rightarrow \infty$  and the weights are the remaining budgets of those clients interested in the newly arrived query (i.e., the bid equals 1). For the online ads problem in which bids and budgets can have general and different values, [20] (its longer version is [21]) uses the “discounted” bids as the weights corresponding to each client. The discount factor is calculated by a function  $\psi(x) = 1 - e^{x-1}$ , of which the input  $x$  is the fraction of a client’s budget that has been consumed. Their algorithm is also  $1 - 1/e$  competitive, under an assumption that bids are small compared to budgets. By taking advantage of estimated numbers of query arrivals for each keyword within a given period and modifying the discount factor in [20], Mahdian et al. [19] designed a class of algorithms which achieve a considerably better competitive ratio with accurate estimates while still guarantee a reasonably good competitive ratio with inaccurate estimates, also assuming small bids.

The algorithms in [5], [9], [6], [8] and [1], all use a primal-dual framework to compute a maximum-weighted matching at each iteration, in which the dual variables (corresponding to each client) are used to determine the weights. The two  $1 - 1/e$  competitive algorithms in [5] and [9] update the dual variables dynamically in their primal-dual type algorithms every time a decision is made. Specifically, each dual variable in [5], which implicitly tracks the fraction of budget that has been spent by the corresponding client, grows during each iteration at a rate parameterized by the fraction of the bid for the incoming query in this client’s total budget, while [9] uses an “exponentially weighted average” of the up-to-date  $n(i)$  most valuable impressions<sup>2</sup> assigned to client  $i$  as a new dual variable with respect to this client. On the other hand, the three dual type learning-based algorithms in [6], [8] and [1] achieve a competitive ratio of  $1 - O(\epsilon)$  based on a random-order arrival model (rather than the adversarial model in most of the earlier work), assuming small bids and knowledge of the total number of queries. The main difference between them is that [6] and [8] use an initial  $\epsilon$  fraction of queries to learn the optimal dual variables (with respect to this training set), while the algorithm in [1] repeats the learning process over geometrically growing intervals. Additionally, the “small bids” condition in [1] is slightly weaker than the condition in [6] and [8].

## B. Our Contributions and Comparison to Prior Work

As in prior work (especially [5] and [9]), our solution relies on a primal-dual framework to solve a maximum-weighted matching problem on a bipartite graph of clients and webpage slots, with dynamically updated dual variables which contribute to the weights on the edges of the bipartite graph. However, unlike prior work, we are able to obtain a revenue which is  $O(\epsilon)$  close to the optimal revenue using a purely adaptive algorithm without the need for the knowledge of the number of query arrivals over a time period or the average arrival rates.

Our solution is related to scheduling problems in wireless networks. In particular, we use the optimization decomposition ideas in [11], the stochastic performance bounds in [18] and the modeling of delay-sensitive flows in [14]. Borrowing from that literature, we introduce the concept of an “*overdraft*” queue. The overdraft queue measures the amount by which the

<sup>2</sup>In the DA model in [9],  $n(i)$  is defined as the maximum number of impressions agreed for client  $i$ . After allowing free disposal, only the current  $n(i)$  most valuable impressions assigned to client  $i$  will be considered.

provided service temporarily exceeds the budget specified by a client. In making the connection to wireless networks, we define something called the “*per-client revenue region*,” which is related to the concept of capacity region in queueing networks (see [11], [18]). In our context, it characterizes the revenue extractable from each client as a function of all the clients’ budgets.

Our online algorithm exhibits a trade-off between the revenue obtained by the service provider and the level of overdrafts. We can further modify our online algorithm so that clients can always operate strictly under their budgets. Finally, our algorithm and analysis naturally allow us to assess the impact of click-through rate estimation on the service providers revenue.

We are able to show that our online algorithm achieves an overdraft level of  $O(1/\epsilon)$ . So a natural question is whether this bound is tight. We show that the overdraft for any algorithm must be  $\Omega(\log(1/\epsilon))$ . While there is a gap between the upper and lower bounds, together they imply that the overdraft must increase when  $\epsilon$  goes to zero. This work is related to [3], [25], [26], [24] and [12] in the context of communication networks. See Section IV for a detailed survey.

Besides the revenue maximization model, we also study another online ads model in which the objective is to maximize the average overall click-through rate, subject to a minimum impression requirement for each client. We also show that our results can be naturally extended to handle non-stationary query arrival processes and clients which have short-term contracts with the service provider. .

Like the algorithm in [1], our algorithm can also be generalized to a wider class of linear programs within different application contexts, where the coefficients in the objective function and constraints are not necessarily nonnegative.

There are two points of departure in our algorithm compared to existing models: the first one is that we assume a purely stochastic model in which the query arrival rates are unknown. Thus, there is no need to know the number of arrivals in a time period as in prior models, and this is even true for non-stationary query arrival processes. The other is that we assume an average budget rather a fixed budget over a time horizon. This allows us to better model permanent clients (e.g., big companies who do not stop advertising) and who do not provide a fixed time-horizon budget. Clients who advertise for a limited amount of time can also be handled well since the algorithm is naturally adaptive.

A minor difference with respect to prior models is that our model assumes that time is slotted. This can be easily modified to assume that query arrivals can occur at any time according to some continuous-time stochastic process. The only difference is that our analysis would then involve continuous-time Lyapunov drift instead of the discrete-time drift used in this paper. From a theoretical point of view, our analysis is different from prior work which uses competitive ratios: our model and solution is similar in spirit to stochastic approximation [4] where gradients (here the gradient of the dual objective) are known only with stochastic perturbations. This point of view is essential to model stochastic traffic with unknown statistics.

Instead of the  $1 - O(\epsilon)$  competitive ratio in prior work, we show that our algorithm achieves a revenue which is within  $O(\epsilon)$  of the optimal revenue. The  $O(\epsilon)$  penalty arises due to the stochastic nature of our model. However, we do not require assumptions such as knowledge of the total number of queries in a given period [19], [6], [8], [1], or information of keyword frequencies [19].<sup>3</sup>

<sup>3</sup>It should be mentioned that another common assumption “small bids” (or “large budgets”, “large offline optimal value”) used in [15], [20], [19], [9], [6] and [8] is not essentially different from our “long-term” assumption.

### C. Organization of the Paper

The rest of the paper is organized as follows: In Section II, we formulate an optimization problem involving long-term averages. In Section III, we start considering the stochastic version of our model and propose an online algorithm, which also introduces the concept of “overdraft queue.” Performance analysis of this online algorithm, which includes the near-optimality of the long-term revenue and an upper bound on the overdraft level, will also be done in Section III. The last two subsections of Section III present two extensions, namely the decisions based on estimated click-through rates and the “underdraft” mechanism. In Section IV, we derive a universal lower bound on the expected overdraft level under any stationary algorithms for online advertising. The second online ads model “click-through rate maximization problem” with its related extensions, algorithm design and analysis is given in Section V. Section VI concludes the whole paper.

Compared to an earlier version of this paper which appeared in [28], we give a more detailed literature survey in Subsection I-A, all the proofs for the lemmas, theorems and corollaries in Section III (we only stated these results without proofs in [28] due to page limits), and full discussions on the underdraft mechanism in Subsection III-F. Sections IV and V are completely new.

## II. AN OPTIMIZATION PROBLEM INVOLVING LONG-TERM AVERAGES

Based on the model described in Section I, we first pose the revenue maximization problem as an optimization problem involving long-term averages. For this purpose, we define an assignment of clients to webpage slots as a matrix  $M$  of which the  $(i, s)^{\text{th}}$  element is defined as follows:

$$M_{is} = \begin{cases} 1, & \text{if client } i \text{ is assigned to webpage slot } s \\ 0, & \text{else.} \end{cases}$$

The matrix  $M$  has to satisfy some practical constraints. First, a webpage slot can be assigned to only one client and vice versa. Furthermore, the assignment of clients to certain webpage slots may be prohibited for certain queries. For example, it may not make sense to advertise chocolates when someone is searching for information about treatments for diabetes. These constraints can be abstracted as follows: For the queried keyword  $q$ , the set of assignment matrices have to belong to some set  $\mathcal{M}_q$ . We also let  $p_{qM}$  be the probability of choosing matrix  $M$  when the queried keyword is  $q$ .

The optimization problem is then given by

$$\max_{\mathbf{p}} \bar{R}(\mathbf{p}) = \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_{i,s} M_{is} c_{qis} r_{qi} \quad (1)$$

subject to

$$N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_s M_{is} c_{qis} r_{qi} \leq b_i, \quad \forall i; \quad (2)$$

$$0 \leq p_{qM} \leq 1, \quad \forall q, M \in \mathcal{M}_q; \quad (3)$$

$$\sum_{M \in \mathcal{M}_q} p_{qM} \leq 1, \quad \forall q. \quad (4)$$

In the above formulation, the objective (1) is the average revenue per time slot and constraint (2) expresses the fact that the average payment over a budgeting cycle should not exceed the average budget. The optimization is a linear program and if all the problem parameters are known, in

principle, it can be solved offline, returning probabilities  $\{p_{qM}\}$  which can be used by a service provider to maximize its revenue. However, such an offline solution is not desirable for at least two reasons:

- Being a static approach, it does not use any feedback about the current state of the system. For example, the fact that the empirical average payment of a client has severely exceeded its average budget would have no impact on the subsequent assignment strategy. Since the formulation and hence, the solution, only cares about long-term budget constraint satisfaction, severe overdraft or underdraft of the budget can occur over long periods of time.
- The offline solution is a function of the query arrival rates  $\{\nu_q\}$ . Thus, a change in the arrival rates would require a recomputation of the solution.

In view of these limitations of the offline solution, we propose an online solution which adaptively assigns client advertisements to webpage slots to maximize the revenue. As we will see, the online solution does use feedback about the overdraft (or underdraft) level in future decisions, and does not require knowledge of  $\{\nu_q\}$ .

### III. ONLINE ALGORITHM AND PERFORMANCE ANALYSIS

#### A. A Dual Gradient Descent Solution

To get some insight into a possible adaptive solution to the problem, we first perform a dual decomposition which suggests a gradient solution. However, a direct gradient solution will not take into the account the stochastic nature of the problem and will also require knowledge of the query arrival rates  $\{\nu_q\}$ . We will address these issues in the following subsections, using techniques that, to the best of our knowledge, have not been used in prior literature on the online advertising problem.

We append the constraint (2) to the objective (1) using Lagrange multipliers  $\delta_i \geq 0$  to obtain a partial Lagrangian function

$$\begin{aligned} L(\mathbf{p}, \boldsymbol{\delta}) &= \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_{i,s} M_{is} c_{qis} r_{qi} - \sum_i \delta_i \cdot \left( \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_s M_{is} c_{qis} r_{qi} - \frac{b_i}{N} \right) \\ &= \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_{i,s} M_{is} c_{qis} r_{qi} (1 - \delta_i) + \sum_i \frac{\delta_i b_i}{N}, \end{aligned}$$

subject to constraints (3) and (4). The dual function is

$$D(\boldsymbol{\delta}) = \max_{\mathbf{p}} \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_{i,s} M_{is} c_{qis} r_{qi} (1 - \delta_i) + \sum_i \frac{\delta_i b_i}{N},$$

subject to constraints (3) and (4). Note that the maximization part in the dual function can be decomposed into independent maximization problems with regard to each queried keyword  $q$ , i.e., for all  $q$ ,

$$\max_{\{p_{qM}, M \in \mathcal{M}_q\}} \sum_{M \in \mathcal{M}_q} p_{qM} \sum_{i,s} M_{is} c_{qis} r_{qi} (1 - \delta_i) = \max_{M \in \mathcal{M}_q} \sum_{i,s} M_{is} c_{qis} r_{qi} (1 - \delta_i),$$

where it is easy to see that each maximization is solved by a deterministic solution. This suggests the following primal-dual algorithm to iteratively solve the original optimization problem (1): at

step  $k$ ,

$$\begin{aligned} \forall q, \quad \hat{M}^*(q, k) &\in \arg \max_{M \in \mathcal{M}_q} \sum_{i,s} M_{is} c_{qis} r_{qi} (1 - \delta_i(k)); \\ \forall i, \quad \delta_i(k+1) &= \left[ \delta_i(k) + \epsilon \left( N \sum_q \nu_q \sum_s [\hat{M}^*(q, k)]_{is} \cdot c_{qis} r_{qi} - b_i \right) \right]^+, \end{aligned}$$

where  $\epsilon > 0$  is a fixed step-size parameter, and  $[x]^+ = x$  if  $x \geq 0$  or  $[x]^+ = 0$  otherwise. Furthermore, defining  $\hat{Q}_i(k) \triangleq \delta_i(k)/\epsilon$ , the above iterative algorithm becomes

$$\begin{aligned} \forall q, \quad \hat{M}^*(q, k) &\in \arg \max_{M \in \mathcal{M}_q} \sum_{i,s} M_{is} c_{qis} r_{qi} \left( \frac{1}{\epsilon} - \hat{Q}_i(k) \right); \\ \forall i, \quad \hat{Q}_i(k+1) &= \left[ \hat{Q}_i(k) + \hat{\lambda}_i(k) - b_i \right]^+, \end{aligned}$$

where

$$\hat{\lambda}_i(k) \triangleq N \sum_q \nu_q \sum_s [\hat{M}^*(q, k)]_{is} c_{qis} r_{qi}. \quad (5)$$

Note that  $\hat{Q}_i(k)$  can be interpreted as a queue which has  $\hat{\lambda}_i(k)$  arrivals and  $b_i$  departures at step  $k$ . Although this algorithm already uses the feedback provided by  $\{\hat{Q}(k)\}$  (or  $\{\delta(k)\}$ ) about the state of the system, it is still using a priori information about the arrival rates of queries in  $\{\hat{\lambda}(k)\}$ , hence not really ‘‘online.’’ However, it motivates us to incorporate a queueing system with stochastic arrivals into the real online algorithm, which will be described in the next subsection.

### B. Stochastic Model, Online Algorithm, and ‘‘Overdraft Queue’’

In practice, a search service provider may not have a priori information about the query arrival rates  $\{\nu_q\}$ , and generally, query arrivals during each time slot are stochastic rather than constant. Let time slots be indexed by  $t \in \mathcal{Z}^+ \cup \{0\}$ . We specify our detailed statistical assumptions as follows:

- **Query arrivals:** Assume that a time slot is short enough so that query arrivals in each time slot can be modeled as a Bernoulli random variable with occurrence probability  $\nu$ . The probability that an arrived query is for keyword  $q$  is assumed to be  $\vartheta_q$  and  $\sum_q \vartheta_q = 1$ . Let  $\tilde{q}(t)$  represent the index of the keyword queried in time slot  $t$ , such that  $\tilde{q}(t) = q$  w.p.  $\nu_q = \nu \vartheta_q$  for all  $q$  (indexed by positive integers) and  $\tilde{q}(t) = 0$  w.p.  $1 - \nu$ , which accounts for the case that no query arrives.
- **Budget spending:** We limit the values of budget spent in each budgeting cycle to be integers. To match the average budget  $b_i$  (when it is not an integer), the budget of client  $i$  in budgeting cycle  $k$  is assumed to be a random variable  $\tilde{b}(k)$  which equals  $\lceil b_i \rceil$  w.p.  $\varrho_i$  and  $\lfloor b_i \rfloor$  otherwise, such that  $E[\tilde{b}(k)] = \varrho_i \lceil b_i \rceil + (1 - \varrho_i) \lfloor b_i \rfloor = b_i$ , i.e.,  $\varrho_i = \frac{b_i - \lfloor b_i \rfloor}{\lceil b_i \rceil - \lfloor b_i \rfloor} = b_i - \lfloor b_i \rfloor$ . For the trivial case that  $b_i$  is already an integer, we let  $\varrho_i = 1$ .
- **Click-through behaviors:** In time slot  $t$ , after a query for keyword  $q$  arrives, if the ad of client  $i$  is posted on webpage slot  $s$  in response to this query, then whether this ad will be clicked is modeled as a Bernoulli random variable  $\tilde{c}_{qis}(t)$  with occurrence probability  $c_{qis}$ .

We now want to implement the above iterative algorithm online based on this stochastic model. According to definition (5),  $\hat{\lambda}_i$  includes average query arrivals and click-through choices within

$N$  time slots (i.e., one budgeting cycle). Thus, each iteration step in the online algorithm should correspond to a budgeting cycle. For convenience, we define

$$\mathbf{u}(k) \triangleq \{\tilde{q}(t), \tilde{c}(t) \text{ for } kN \leq t \leq kN + N - 1\}$$

as a collection of random variables describing user behaviors (including stochastic query arrivals and click-through choices) in budgeting cycle  $k$ . The online algorithm is then described as follows:

---

**Online Algorithm:** (in each budgeting cycle  $k \geq 0$ )

In each time slot  $t \in [kN, kN + N - 1]$ , if  $\tilde{q}(t) > 0$ , choose the assignment matrix

$$\tilde{M}^*(t, \tilde{q}(t), \mathbf{Q}(k)) \in \arg \max_{M \in \mathcal{M}_{\tilde{q}(t)}} \sum_{i,s} M_{is} c_{\tilde{q}(t)is} r_{\tilde{q}(t)i} \left( \frac{1}{\epsilon} - Q_i(k) \right). \quad (6)$$

At the end of budgeting cycle  $k$ , for each client  $i$ , update

$$Q_i(k+1) = \left[ Q_i(k) + A_i(k, \mathbf{Q}(k), \mathbf{u}(k)) - \tilde{b}_i(k) \right]^+, \quad (7)$$

where

$$A_i(k, \mathbf{Q}(k), \mathbf{u}(k)) \triangleq \sum_{t=kN}^{kN+N-1} \sum_s [\tilde{M}^*(t, \tilde{q}(t), \mathbf{Q}(k))]_{is} \cdot \tilde{c}_{\tilde{q}(t)is}(t) \cdot r_{\tilde{q}(t)i}. \quad (8)$$

---

Here,  $A_i(k, \mathbf{Q}(k), \mathbf{u}(k))$  represents the revenue obtained by the service provider from client  $i$  during budgeting cycle  $k$ , and recall that  $\tilde{b}_i(k)$  is a random variable which takes integer values whose mean is equal to the average budget per budgeting cycle.

In this algorithm, client  $i$  is associated with a virtual queue  $Q_i$  (maintained at the search service provider). During budgeting cycle  $k$ , the amount of money client  $i$  is charged by the search service provider  $A_i(k, \mathbf{Q}(k), \mathbf{u}(k))$  is the arrival to this queue, and the average budget per budgeting cycle  $b_i$  is the departure from this queue. Note that if this queue is positive, it means that the total value of the real service already provided to the client has temporarily exceeded the client's budget, i.e., "free" service has been provided temporarily. Hence, we call this queue the "overdraft queue."

There are two different time scales here. The faster one is a time slot, the smallest time unit used to capture user behaviors (including stochastic query arrivals and click-through choices) and execute ad-posting strategies. The slower one is a budgeting cycle (equal to  $N$  time slots), at the end of which the overdraft queues are updated based on the revenue obtained over the whole budgeting cycle.

We make the following assumptions on the above stochastic model:  $\{\tilde{q}(t)\}$  are i.i.d. across time slots  $t$ ;  $\{\tilde{c}_{qis}(t)\}$  are independent across  $q, i, s$ , and  $t$ ; each variable in  $\{\tilde{q}(t)\}$  and each variable in  $\{\tilde{c}_{qis}(t)\}$  are mutually independent. In fact, the model can be generalized to allow for query arrivals correlated over time and across keywords, and other similar correlations inside the click-through choices or between these two stochastic processes. Such models would only make the stochastic analysis more cumbersome, but the main results will continue to hold under these more general models.

In order to guarantee that the Markov chain which we will define later is both irreducible and aperiodic, we further assume that the probability of whether there is an arrival in a time slot  $\nu \in (0, 1)$ . We also assume that  $r_{qi}$  for all  $q$  and  $i$  can only take integer values. Together with the fact that  $\tilde{b}(k)$  takes integer values,  $\{\mathbf{Q}(k)\}$  becomes a discrete-time integer-valued queue. Note that assuming integer values is only for ease of analysis, but not necessary.

### C. An Upper Bound on the Overdraft

According to the ad assignment step (6), if at the beginning of budgeting cycle  $k$ ,  $Q_i(k) > 1/\epsilon$ , then for this budgeting cycle, the  $i^{\text{th}}$  row of  $\tilde{M}^*(t, q, \mathbf{Q}(k))$  is always a zero vector, i.e., the service provider will not post the ads of client  $i$  until  $Q_i(k)$  falls below  $1/\epsilon$ . Since by assumption the number of query arrivals per time slot is upper bounded, for any budgeting cycle  $k$ , one can bound the transient length of each overdraft queue as below:

$$Q_i(k) \leq \frac{1}{\epsilon} + N \cdot \arg \max_{q,s} \{r_{qi}c_{qis}\} - \lfloor b_i \rfloor, \quad \forall i.$$

Therefore,  $Q_i(k) \sim O(1/\epsilon)$  for all  $i$ , and stability is not an issue for these ‘‘upper bounded’’ queues. It further implies that this online algorithm satisfies the budget constraints in the long run, i.e., for all client  $i$ ,

$$\lim_{K \rightarrow \infty} E \left[ \frac{1}{K} \sum_{k=0}^{K-1} A_i(k, \mathbf{Q}(k), \mathbf{u}(k)) \right] \leq b_i \quad (9)$$

must hold.

It should be mentioned that in [12], through using the LIFO queueing discipline, the authors show an  $O((\log(1/\epsilon))^2)$  bound on the averaged waiting time encountered by most of the packets, which is tighter than the bound  $O(1/\epsilon)$  under the FIFO queueing discipline (see e.g. [11]; our above result also fits this bound). While the length of a FIFO queue is proportional to the arrival rate according to Little’s law [2], the length of a LIFO queue in [12] is still  $O(1/\epsilon)$ , even if it is occupied by very ‘‘old’’ packets which only accounts for a negligible fraction  $O(\epsilon^{\log(1/\epsilon)})$  of all the packets that have arrived. Unlike in a communication network where waiting time is usually the main concern and dropping a small fraction of old packets does almost no hurt to many online applications, what clients of online advertising service care about is how much they have paid beyond their budgets, which is measured by the overdraft queue in our model.

### D. Near-Optimality of the Online Algorithm

We now show that, in the long term, the proposed online algorithm achieves a revenue that is close to the optimal revenue  $\bar{R}(\mathbf{p}^*)$  (where  $\mathbf{p}^*$  is the solution to the optimization problem (1)). We start with the following lemma:

*Lemma 1:* Consider the Lyapunov function  $V(\mathbf{Q}) = \frac{1}{2} \sum_i Q_i^2$ . For any  $\epsilon > 0$ , and each time period  $k$ ,

$$E[V(\mathbf{Q}(k+1)) | \mathbf{Q}(k) = \mathbf{Q}] - V(\mathbf{Q}) \leq -\frac{N}{\epsilon} (\bar{R}(\mathbf{p}^*) - \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q}))) + B_1 - B_2 \sum_i Q_i.$$

Here,

$$B_1 \triangleq \frac{1}{2} \left( (N(N-1)L^2 + NL) (\arg \max_{q,i,s} \{c_{qis}r_{qi}\})^2 + \sum_i \lfloor b_i \rfloor^2 (b_i - \lfloor b_i \rfloor) + \lfloor b_i \rfloor^2 (1 - b_i + \lfloor b_i \rfloor) \right), \quad (10)$$

where  $L$  is the number of webpage slots;

$$B_2 \triangleq \min_i \{b_i - N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM}^* \sum_s M_{is} c_{qis} r_{qi}\}; \quad (11)$$

and  $\tilde{\mathbf{p}}^*(k, \mathbf{Q}) \triangleq \{\tilde{p}_{qM}^*(k, \mathbf{Q}), \forall q, M \in \mathcal{M}_q\}$  where  $\tilde{p}_{qM}^*(k, \mathbf{Q})$  equals 1 if  $M = \tilde{M}^*(t, q, \mathbf{Q})$  for  $kN \leq t \leq kN + N - 1$  (i.e., the optimal matrix in the maximization step (6)) and 0 otherwise.

◇

The proof is given in Appendix A.

Now we are ready to present one of the major theorems in this paper, indicating that the long-term average revenue achieved by our online algorithm is within  $O(\epsilon)$  of the maximum revenue obtained by the offline optimal solution. The proof is given in Appendix B.

*Theorem 1:* For any  $\epsilon > 0$ ,

$$0 \leq \lim_{K \rightarrow \infty} E \left[ \bar{R}(\mathbf{p}^*) - \frac{1}{KN} \sum_{k=0}^{K-1} R(k) \right] \leq \frac{B_1 \epsilon}{N}$$

for some constant  $B_1 > 0$  (defined in (10) in Lemma 1), where  $R(k) \triangleq \sum_i A_i(k, \mathbf{Q}(k), \mathbf{u}(k))$ . is defined as the revenue obtained during budgeting cycle  $k$ . ◇

*Remark 1:* If we choose a very small  $\epsilon$ , the matching in (6) behaves like a greedy solution until the queue lengths grows comparably large. This indicates a tradeoff between how close to the long-term optimal revenue the algorithm can achieve and the actual convergence time.

Additionally, supposing that  $\{r_{qi}\}$  and  $\{b_i\}$  are both measured in another scale with a factor  $\alpha$ , e.g., using cents instead of dollars ( $\alpha = 100$ ), and assuming that  $\alpha$  is unknown, it can be shown that the  $O(\epsilon)$  convergence bound will also be scaled by  $\alpha$  if we measure the revenue in the original scale. To change the algorithm into a “scale-free” version,  $\{r_{qi}\}$  and  $\{b_i\}$  should be divided by a common benchmark value, e.g., the largest budget specified by all the initially existing clients. Since the benchmark value is also implicitly multiplied by  $\alpha$  if measured in another scale, the scaling factor will be canceled in the normalized  $\{r_{qi}\}$  and  $\{b_i\}$  and no longer affect the convergence bound. ◇

### E. Impact of Click-Through Rate Estimation

In our online algorithm, the decision of picking an optimal ad assignment matrix in (6) in response to each query is based on the true click-through rates  $\mathbf{c}$ . In reality, an estimate  $\hat{\mathbf{c}}$  based on historical click-through behaviors is used, i.e., in response to each query for keyword  $q$ , which arrives in time slot  $t \in [kN, kN + N - 1]$ , we choose the assignment matrix

$$\tilde{M}^*(t, \tilde{q}(t), \mathbf{Q}(k)) \in \arg \max_{M \in \mathcal{M}_{\tilde{q}(t)}} \sum_{i,s} M_{is} \hat{c}_{\tilde{q}(t)is} r_{\tilde{q}(t)i} \left( \frac{1}{\epsilon} - Q_i(k) \right). \quad (12)$$

We then have the following corollary in addition to Theorem 1 in Subsection III-D:

*Corollary 1:* Assume that the estimated click-through rates  $\hat{\mathbf{c}} \in [\mathbf{c}(1 - \Delta), \mathbf{c}(1 + \Delta)]$  with some  $\Delta \in (0, 1)$ . Under our online algorithm with estimated click-through rates,  $\mathbf{Q}(k)$  is still positive recurrent. Then, for any  $\epsilon > 0$ ,

$$\lim_{K \rightarrow \infty} E \left[ \frac{1}{KN} \sum_{k=0}^{K-1} R(k) \right] \geq \left( \frac{1 - \Delta}{1 + \Delta} \right) \cdot \bar{R}(\mathbf{p}^*) - \frac{B_1 \epsilon}{N},$$

for some constant  $B_1 > 0$  (defined in equation (10) in Lemma 1). ◇

Proving this needs some minor changes to the proof of Lemma 1 and Theorem 1, which will be shown in Appendix C.

*Remark 2:* Corollary 1 tells us that for small  $\epsilon$ , the long-term average revenue achieved by our online algorithm with estimated click-through rates will be at least  $(\frac{1-\Delta}{1+\Delta})$  of the offline optimal revenue.  $\diamond$

#### F. Underdraft: Staying under the Budget

In the previous sections, we allowed the provision of temporary free service to clients, which we call overdraft. If this is not desirable for some reason, the algorithm can be modified to have non-positive overdraft. We do this by allowing the queue lengths to become negative, but not positive. The practical meaning of negative queue lengths is to allow each client to accumulate a certain volume of “credits” if the current budget is under-utilized and use these credits to offset future possible overdrafts. We call this negative queue length “*underdraft*.” Corresponding to this mechanism, we modify our online algorithm as follows: in response to each query for keyword  $q$ , which arrives in time slot  $t \in [kN, kN + N - 1]$ , choose the assignment matrix

$$\tilde{M}^*(t, \tilde{q}(t), \mathbf{Q}(k)) \in \arg \max_{M \in \mathcal{M}_{\tilde{q}(t)}} \sum_{i,s} M_{is} c_{\tilde{q}(t)is} r_{\tilde{q}(t)i} (\Gamma_i - Q_i(k)),$$

and at the end of budgeting cycle  $k$ , for each client  $i$ , update

$$Q_i(k+1) = \max\{Q_i(k) + A_i(k, \mathbf{Q}(k), \mathbf{u}(k)) - \tilde{b}_i(k), -C_i\},$$

where  $\Gamma_i$  denotes a customized “throttling threshold” (not necessarily  $1/\epsilon$ ) and  $C_i$  denotes the maximum allowable credit volume for client  $i$ . Recall that  $A_i(k, \mathbf{Q}(k), \mathbf{u}(k))$  is defined in equation (8).

We can bound each overdraft queue as below:

$$Q_i(k) \leq \Gamma_i + N \cdot \arg \max_{q,s} \{r_{qi} c_{qis}\} - \lfloor b_i \rfloor, \quad \forall i, k.$$

Thus, if our objective is to eliminate overdrafts (i.e.,  $Q_i(k) \leq 0$  for all  $k$ ), we can set

$$\Gamma_i := \left[ \lfloor b_i \rfloor - N \cdot \arg \max_{q,s} \{r_{qi} c_{qis}\} \right]^{-}, \quad \forall i, \quad (13)$$

where in contrary to  $[x]^+$ ,  $[x]^-$  takes the non-positive part of  $x$ , i.e.,  $[x]^- = x$  if  $x \leq 0$  or  $[x]^- = 0$  otherwise. We further let

$$C_i := \frac{1}{\epsilon} - \Gamma_i, \quad \forall i,$$

so that after converting  $Q_i(k)$  to be nonnegative by using  $\tilde{Q}_i(k) = Q_i(k) + C_i$  for all  $i$ , everything is transformed back to the original online algorithm except that each  $Q_i(k)$  is replaced by  $\tilde{Q}_i(k)$ , hence we can still show that the revenue achieved by this modified version of online algorithm is within  $O(\epsilon)$  of the optimal revenue.

It might seem counter-intuitive that by letting  $\epsilon$  go to zero, we can incur potentially large underdrafts (under-utilization of the budget) and yet are able to achieve maximum revenue. This is not a contradiction: for each fixed  $\epsilon$ , in the long term, the average service provided to each client is close to the average budget. The  $O(1/\epsilon)$  is a fixed amount by which the total budget

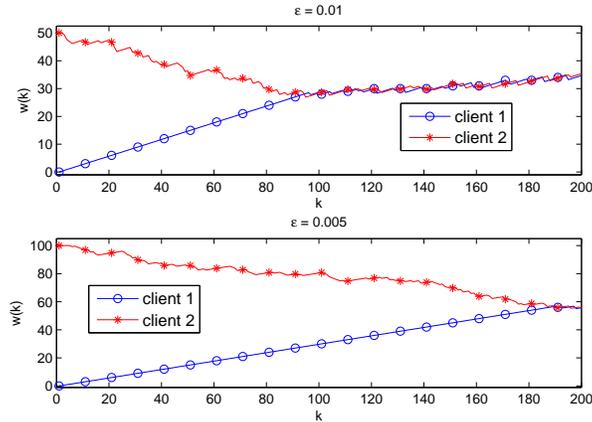


Fig. 1: Temporary unfairness in service

up to any time  $T$  is under-utilized, and, after divided by  $T$ , it goes to zero when  $T$  approaches infinity.

We note that while an underdraft does not seem to significantly hurt either the client, who actually benefits from an underdraft, or the service provider, whose long-run average revenue is still diminished only by  $O(\epsilon)$ , large values of the underdraft may result in temporary unfairness in the system.<sup>4</sup> If, for example, a client accumulates a large underdraft compared to the other clients, then it may receive priority over other clients for large periods of time. To illustrate this, we consider an example with two clients and one queried keyword. Assume that  $\Gamma_i < 0$  for  $i = 1, 2$ , and at time slot  $k_0$ ,  $Q_1(k_0) = \Gamma_1$  and  $Q_2(k_0) = -C_2$  (this occurs with a positive probability due to the ergodicity of the Markov chain  $\{\mathbf{Q}(k)\}$  proved before). We simulate the sample paths of the weights in the maximization step (32) with the following setting: budgets  $b_1 = b_2 = 0.6$ , click-through rates  $c_1 = c_2 = 0.5$ , revenue-per-click  $r_1 = r_2 = 1$ ; the number of query arrivals per time slot equals 2 w.p. 0.5 and 0 otherwise; a budgeting cycle equals to one time slot ( $N = 1$ ) for simplicity. The results for both  $\epsilon = 0.01$  and  $\epsilon = 0.005$  ( $k = 0$  corresponds to  $k_0$  here) are shown in Figure 1. Client 2 keeps getting services until the weights of both clients reaches the same level, and the smaller  $\epsilon$  is, the longer the “unfair serving” period lasts.

It should be mentioned that this underdraft idea can be used under any upper-bounded query arrival model, not restricted in the Bernoulli arrival model considered in this paper.

#### IV. A UNIVERSAL LOWER BOUND ON THE EXPECTED OVERDRAFT LEVEL

We want to show that in the long run, an expected overdraft level of  $\Omega(\log(1/\epsilon))$  is unavoidable under any stationary ad assignment algorithm which achieves a long-term average revenue within  $O(\epsilon)$  of the offline optimum, when the queue length is only allowed to be nonnegative. An ad assignment algorithm  $\varpi$  is defined as a strategy which uses matrix  $M^\varpi(t, q) \in \mathcal{M}_q$  for ad

<sup>4</sup>Note that this temporary unfairness is not an artifact of the underdraft mechanism. In fact, it occurs once a sample path enters a state where some clients have huge differences from others in their corresponding queue lengths, which can also happen under the original algorithm. We are just using the underdraft scheme to illustrate this phenomenon.

assignment when a query for keyword  $q$  arrives at each time slot  $t$ . During each budgeting cycle  $k$ , the revenue obtained from client  $i$  under algorithm  $\varpi$  is defined as

$$A_i^\varpi(k) \triangleq \sum_{t=kN}^{kN+N-1} \sum_s [M^\varpi(t, \tilde{q}(t))]_{is} \cdot \tilde{c}_{\tilde{q}(t)is}(t) \cdot r_{\tilde{q}(t)i}. \quad (14)$$

We then define average revenue obtained from client  $i$  per budgeting cycle as  $\lambda_i^\varpi \triangleq E[A_i^\varpi(k)]$  in the steady state. The long-term average revenue (per time slot) is thus  $\bar{R}^\varpi = \sum_i \lambda_i^\varpi / N$ , and the overdraft level of client  $i$  evolves as

$$Q_i^\varpi(k+1) = \left[ Q_i^\varpi(k) + A_i^\varpi(k) - \tilde{b}_i(k) \right]^+. \quad (15)$$

Note that our online algorithm is one particular  $\varpi$ , which makes the decision based on the current overdraft levels of all clients.

To seek a universal lower bound on expected overdraft level in the long run (here, equivalent to steady state), we only have to consider those algorithms  $\varpi$  such that  $Q_i^\varpi \triangleq E[Q_i^\varpi(k)] < \infty$  for all  $i$ . To categorize these “stable” algorithms, we define “per-client revenue region,” similar to the concept of “capacity region” in the context of queueing networks:

*Definition 1 (“Per-Client Revenue Region”):*

$$\mathcal{C} \triangleq \left\{ \boldsymbol{\lambda}^\varpi = \{\lambda_i^\varpi\} \geq \mathbf{0} : \exists \varpi \text{ s.t. } \lambda_i^\varpi \triangleq E[A_i^\varpi(k)] \leq b_i, \forall i \right\},$$

given fixed parameters  $\{r_{qi}\}$ ,  $\{b_i\}$ ,  $\{c_{qis}\}$ ,  $N$  and statistical properties of  $\tilde{q}(t)$  and  $\{\tilde{c}_{qis}(t)\}$ .  $\diamond$   
The offline optimal average revenue is then equal to  $\max_{\boldsymbol{\lambda} \in \mathcal{C}} \sum_i \lambda_i / N$ , which is denoted as  $\bar{R}^*$ .

Note that if the query arrival rates per budgeting cycle are too low, the average revenue drawn from some client will never hit its specified budget, no matter which algorithm  $\varpi$  s.t.  $\boldsymbol{\lambda}^\varpi \in \mathcal{C}$  you pick (i.e.,  $\exists i$  s.t. no feasible solution  $\mathbf{p}$  can make constraint (2) for this  $i$  tight). The system resources (here, budgets) are underutilized and it is not so important to consider the tradeoff between revenue and overdraft. To avoid this, we can assume a relatively large  $N$  (i.e., the number of time slots in one budgeting cycle) such that

$$N \geq \max_i \left\{ \frac{b_i}{\sum_q \nu_q r_{qi} \cdot \max_{M \in \mathcal{M}_q^i} c_{qis(i,M)}} \right\}, \quad (16)$$

where  $\mathcal{M}_q^i \subseteq \mathcal{M}_q$  is defined as a set of ad assignment matrices, of which the  $i^{\text{th}}$  row has a “1”, and  $s(i, M)$  in  $c_{qis(i,M)}$  refers to the column in  $M$  where that “1” stays. This guarantees that for each  $i$ , there exists an algorithm  $\varpi_i$  such that  $\boldsymbol{\lambda}^{\varpi_i} \in \mathcal{C}$  and  $\lambda_i^{\varpi_i} = b_i$ . The reason is that

In the following text, we will assume the above condition for  $N$ .

#### A. One Keyword, One Client and One Webpage Slot

We start with the simplest model: one keyword, one client and one webpage slot (hence we omit all the subscripts in the corresponding notations). Under condition (16), the offline maximum average revenue is trivially  $b/N$ .

*Theorem 2:* Given a small  $\epsilon > 0$ , if an algorithm  $\varpi$  leads to  $E[A^{\varpi}(k)] \geq b - \epsilon$  in the steady state, then

$$\bar{Q}^{\varpi} \geq \frac{\log(1/\epsilon)}{2(1 - \log(\varphi P_+))} - 1,$$

where we assume that

$$\varphi \triangleq \Pr(\text{no query arrival in a budgeting cycle}) > 0,$$

and  $P_+ \triangleq \Pr(\tilde{b}(k) > 0) > 0$ .  $\diamond$

Note that this result works for any query arrival and budget spending model satisfying the above two stated assumptions, and not only restricted to the model we described in Subsection III-B. In the proof below, we generally write  $\tilde{b}(k)$  as a random variable which can possibly take all nonnegative integer values.

*Proof:* We ignore the superscript  $\varpi$  for brevity. The dynamics of the queue is rewritten as  $Q(k+1) = Q(k) + A(k) - \hat{b}(k)$ , where the actual departure process is defined as

$$\hat{b}(k) \triangleq \begin{cases} \tilde{b}(k) & \text{if } Q(k) + A(k) - \tilde{b}(k) \geq 0; \\ Q(k) + A(k) & \text{otherwise.} \end{cases} \quad (17)$$

Let  $p_i \triangleq \Pr(\hat{b}(k) = i)$  and  $q_i \triangleq \Pr(\tilde{b}(k) = i)$  in the steady state. Note that

$$\begin{aligned} b - \epsilon &\leq E[A(k)] = E[\hat{b}(k)] = \sum_{i=1}^{\infty} \Pr(\hat{b}(k) \geq i) = \Pr(\hat{b}(k) \geq 1) + \sum_{i=2}^{\infty} \Pr(\hat{b}(k) \geq i) \\ &\stackrel{(a)}{\leq} (1 - p_0) + \sum_{i=2}^{\infty} \Pr(\tilde{b}(k) \geq i) = 1 - p_0 + (b - \Pr(\tilde{b}(k) \geq 1)) = 1 - p_0 + b - (1 - q_0) \\ &= q_0 - p_0 + b, \end{aligned}$$

where (a) holds because  $\Pr(\hat{b}(k) \geq i) \leq \Pr(\tilde{b}(k) \geq i)$  for all  $i \geq 0$ . Thus,  $p_0 \leq q_0 + \epsilon$ . Since

$$\Pr(\hat{b}(k) = 0) = \Pr(\tilde{b}(k) = 0) + \Pr(\hat{b}(k) = 0, \tilde{b}(k) \geq 1),$$

we have  $p_0 = q_0 + \tilde{p}_0$ , where  $\tilde{p}_0 \triangleq \Pr(\hat{b}(k) = 0, \tilde{b}(k) \geq 1)$ . Therefore,

$$\tilde{p}_0 \leq \epsilon. \quad (18)$$

Next, we are looking for a lower bound on  $\tilde{p}_0$  in relation to  $\bar{Q}$ . Letting  $P_+ \triangleq \Pr(\tilde{b}(k) > 0)$  (which is surely a positive constant since  $b > 0$ ), we then have

$$\begin{aligned} n\tilde{p}_0 &= \sum_{k=0}^{n-1} \Pr(\hat{b}(k) = 0, \tilde{b}(k) > 0) \stackrel{(a)}{\geq} \Pr\left(\bigcup_{k=0}^{n-1} \{\hat{b}(k) = 0, \tilde{b}(k) > 0\}\right) \\ &\stackrel{(b)}{\geq} \Pr(Q(0) \leq n-1; A(k) = 0, \tilde{b}(k) > 0, \forall 0 \leq k \leq n-1) \\ &= \Pr(Q(0) \leq n-1) \cdot \prod_{k=0}^{n-1} \Pr(A(k) = 0) \cdot \Pr(\tilde{b}(k) > 0) \\ &= (\varphi P_+)^n \cdot \Pr(Q(0) \leq n-1) \stackrel{(c)}{\geq} (\varphi P_+)^n (1 - \bar{Q}/n), \end{aligned} \quad (19)$$

where (a) holds according to the union bound, (b) holds since the event on the RHS implies the one on the LHS, and (c) holds due to the Markov inequality. If we pick  $n := \lceil 2\bar{Q} \rceil \in [2\bar{Q}, 2(\bar{Q} + 1)]$ , inequality (19) further implies that

$$\tilde{p}_0 \geq \frac{(\varphi P_+)^n}{2n} \stackrel{(e)}{\geq} e^{-n(1-\log(\varphi P_+))} \geq e^{-2(\bar{Q}+1)(1-\log(\varphi P_+))}, \quad (20)$$

where (e) holds because  $\frac{1}{2x} \geq e^{-x}$  for all  $x > 0$ . Combining inequalities (18) and (20) then completes the proof. ■

In the related literature, [3] comes up with an  $\Omega(1/\sqrt{\epsilon})$  bound for a set of algorithms under some admissibility conditions, while [25] provides an  $\Omega(\log(1/\epsilon))$  bound for more general algorithms.

Our proof uses the following ideas inspired by [25]: if the throughput is lower bounded by a number close to the average potential departure rate, then the probability of zero actual departures given nonzero potential departures must be upper bounded by a small number; further, if the average queue length is given, then the probability of hitting zero must be upper bounded because otherwise, the queue length would become small. However, we cannot directly use the expression for the lower bound in [25] since it imposes certain strict convexity assumptions which do not apply to our model where the objective is linear. So we have provided a very simple derivation of the lower bound on the queue length for our specific model.

Additionally, our  $\Omega(\log(1/\epsilon))$  bound based on a linear objective function can be extended to the multi-queue case (in Subsection IV-B). The  $\Omega(1/\sqrt{\epsilon})$  bound in [3] has been extended to the multi-queue case in [24] but still under strict convexity assumption and for a restrictive class of algorithms. Whether the  $\Omega(\log(1/\epsilon))$  bound in [25] can be easily extended to multiple queues still remains a question.

### B. Multiple Keywords, Multiple Clients and Multiple Webpage Slots

We now extend this lower bound to the original general model, which can have multiple keywords, multiple clients and multiple webpage slots. It is easy to see that the “per-client revenue region”  $\mathcal{C}$  in Definition 1 is a polytope, which can then be rewritten as

$$\mathcal{C} = \left\{ \boldsymbol{\lambda} \geq \mathbf{0} : \sum_i h_i^{(n)} \lambda_i \leq d^{(n)}, \forall 1 \leq n \leq L \right\}, \quad (21)$$

where  $h_i^{(n)} \geq 0$  and  $d^{(n)} > 0$  for all  $i$  and  $n$ . The outer boundary of the polytope  $\mathcal{C}$  consists of the  $L$  hyperplanes, i.e.,  $\sum_i h_i^{(n)} \lambda_i = d^{(n)}$  for all  $n \in [1, L]$ .

Under condition (16),  $L$  is at least equal to the number of clients (i.e., number of budget constraints), so (21) gives a more precise description of the stability condition for this “multi-queue system,” compared to the original definition of  $\mathcal{C}$ . Thus, corresponding to the normal vector of each hyperplane, we convert the original multi-queue system into a new one with  $L$  queues: For each  $n \in [1, L]$ , we first scale the  $i^{\text{th}}$  queue described in (15) by  $h_i^{(n)}$ , so that it has a queue length equal to  $h_i^{(n)} Q_i(k)$ , with  $h_i^{(n)} A_i(k)$  arrivals and  $h_i^{(n)} \tilde{b}_i(k)$  potential departures in time slot  $k$ , for all  $i$ . Next, we treat  $\sum_i h_i^{(n)} Q_i(k)$  as the  $n^{\text{th}}$  queue, and since any  $\boldsymbol{\lambda} \in \mathcal{C}$  satisfies  $\sum_i h_i^{(n)} \lambda_i \leq d^{(n)}$ , its maximum achievable average departure rate equals  $d^{(n)}$ , where  $d^{(n)} \leq \sum_i h_i^{(n)} b_i$ , because the potential departure rate of each individual scaled queue may not be fully achieved when all of them are coupled together.

We then come up with the formal definition of the class of algorithms which achieves a “near-optimal” average revenue.

*Definition 2 (“ $\epsilon$ -Neighbourhood” of the maximum):* Let  $\lambda^*$  be one optimal point in  $\mathcal{C}$  such that  $\sum_i \lambda_i^* = \bar{R}^*$ . The  $\epsilon$ -neighbourhood of  $\lambda^*$  is defined as

$$\mathcal{N}_\epsilon \triangleq \{\lambda^\varpi \in \mathcal{C} \setminus \partial\mathcal{C} : 0 < N \cdot (\bar{R}^* - \bar{R}^\varpi) \leq \epsilon\}, \quad (22)$$

where  $\partial\mathcal{C}$  represents the outer boundary of  $\mathcal{C}$ , and it should be noted that the average revenue is evaluated per time slot while  $\lambda$  is evaluated per  $N$  time slots.  $\diamond$

Note that in the above definition, since  $\lambda^\varpi \in \mathcal{N}_\epsilon$  is not on any boundary,  $\bar{R}^*$  is strictly larger than  $\bar{R}^\varpi$ , which is easy to see from some basic principles of linear programming.

The following theorem shows the universal lower bound  $\Omega(\log(1/\epsilon))$  for the general case.

*Theorem 3:* For any algorithm  $\varpi$  s.t.  $\lambda^\varpi \in \mathcal{N}_\epsilon$ , we have

$$\sum_{i=1}^M \bar{Q}_i^\varpi \geq \frac{\log(1/\epsilon) - C_2}{C_1} - 1,$$

where  $\varphi \triangleq \Pr(\text{no query arrival in a budgeting cycle}) = (1 - \nu)^N > 0$ ,  $P_+ \triangleq \Pr(\tilde{b}_i(k) > 0, \forall i) > 0$ , and

$$\begin{aligned} C_1 &\triangleq 2(1 - \log(\varphi P_+)) \cdot \max_{i,n} h_i^{(n)} \in (0, \infty), \\ C_2 &\triangleq \max\{\log(\max_{i,n} h_i^{(n)}), 0\} \in [0, \infty). \end{aligned} \quad (23)$$

$\diamond$

*Proof:* We ignore the superscript  $\varpi$  for brevity. According to some basic principles of linear programming, an optimal point  $\lambda^*$  is at a corner of  $\mathcal{C}$ . If there are several optimal points, any convex combination of them is also optimal. Denote this optimal point sets as  $\Lambda^*$  and  $\forall \lambda^* \in \Lambda^*$ ,  $\exists n^* \in [1, L]$ , s.t.  $\sum_i h_i^{(n^*)} \lambda_i^* = d^{(n^*)}$ .

Given a  $\lambda \in \mathcal{N}_\epsilon$ ,  $\exists \theta$  s.t.  $\sum_i \theta_i = \sum_i \lambda_i^*$  and  $\theta_i \geq \lambda_i$  for all  $i$  (but at least one inequality is strict). Besides, for this  $\theta$ ,  $\exists \tilde{n} \in [1, L]$ , s.t.  $\sum_i h_i^{(\tilde{n})} \theta_i \geq d^{(\tilde{n})}$  (otherwise,  $\theta \in \mathcal{C} \setminus \partial\mathcal{C}$  will hold and hence  $\sum_i \theta_i < \sum_i \lambda_i^*$ , which leads to a contradiction). Therefore,

$$\begin{aligned} d^{(\tilde{n})} - \sum_i h_i^{(\tilde{n})} \lambda_i &\leq \sum_i h_i^{(\tilde{n})} (\theta_i - \lambda_i) \stackrel{(a)}{\leq} h_{max}^{(\tilde{n})} \sum_i (\theta_i - \lambda_i) \\ &= h_{max}^{(\tilde{n})} \sum_i (\lambda_i^* - \lambda_i) \leq h_{max}^{(\tilde{n})} \epsilon, \end{aligned} \quad (24)$$

where  $h_{max}^{(\tilde{n})} \triangleq \max_i h_i^{(\tilde{n})} > 0$  and inequality (a) holds because  $\theta_i \geq \lambda_i$  for all  $i$ . Letting  $P'_+ \triangleq \Pr(\sum_i h_i^{(\tilde{n})} \tilde{b}_i(k) > 0)$ , it is easy to see that  $P'_+ \geq \Pr(\tilde{b}_i(k) > 0, \forall i) = P_+ > 0$ . Together with Theorem 2, we can conclude that

$$\sum_i h_i^{(\tilde{n})} \bar{Q}_i \geq \frac{\log(1/\epsilon) - \log(h_{max}^{(\tilde{n})})}{2(1 - \log(\varphi P'_+))} - 1 \geq \frac{\log(1/\epsilon) - \log(h_{max}^{(\tilde{n})})}{2(1 - \log(\varphi P_+))} - 1.$$

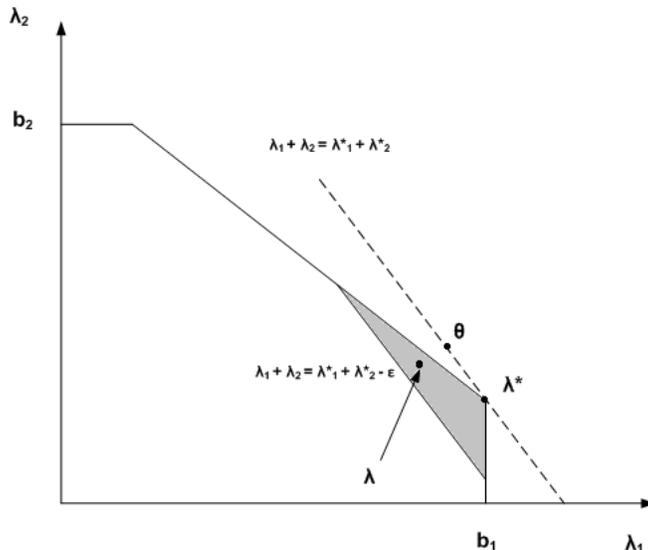


Fig. 2: An illustration of the idea in the proof of Theorem 3

Since  $\sum_i h_i^{(\tilde{n})} \bar{Q}_i \leq h_{\max}^{(\tilde{n})} \sum_i \bar{Q}_i$ , it is further concluded that

$$\sum_i \bar{Q}_i \geq \frac{\log(1/\epsilon) - \log(h_{\max}^{(\tilde{n})})}{2h_{\max}^{(\tilde{n})}(1 - \log(\varphi P_+))} - 1 \geq \frac{\log(1/\epsilon) - C_2}{C_1} - 1,$$

where the universal constants are defined in (23), and it is guaranteed that  $C_1 \in (0, \infty)$  and  $C_2 \in [0, \infty)$ . This completes the proof. ■

*Remark 3:* We briefly explain the idea behind choosing  $\theta$  in the above proof: For those  $\lambda \in \mathcal{N}_\epsilon$  such that  $\lambda_i \leq \lambda_i^*$  for all  $i$  (at least one is strict),  $\theta$  can be directly chosen as  $\lambda^*$  to make inequality (a) in (24) hold. But for the other  $\lambda \in \mathcal{N}_\epsilon$  which do not satisfy the above condition, it is necessary to introduce a  $\theta$  other than  $\lambda^*$ , which both lies on the “maximum revenue line” (i.e.,  $\sum_i \theta_i = \sum_i \lambda_i^*$ ) and dominates  $\lambda$  component-wise, in order to derive inequality (24). Note that  $\theta$  is not unique and furthermore,  $\theta$  lies either on  $\partial\mathcal{C}$  or in the exterior of  $\mathcal{C}$  and it can be chosen as a boundary point only if the optimal revenue point is not unique. Figure 2 illustrates this idea using an example with one keyword, two clients and one webpage slot, specifically for showing where such a  $\theta$  is located. ◇

The basic idea in our proof is to use Theorem 2 to first get a lower bound for those new single queues written as a “weighted sum” of the original queues (described above). This idea is similar to one part in the proof for the lower bound on the expected queue length of a departure-controlled multi-queue system in [26], but some technique in their proof cannot directly apply to arrival-controlled queues like ours.

### C. Tightness of the Lower Bound

We want to show that the  $\Omega(\log(1/\epsilon))$  universal lower bound is tight, i.e., achievable by some algorithms. Consider the following simple queueing model: the arrival process  $a(k)$  is i.i.d. across time,  $a(k) = 2$  w.p.  $\nu$  and  $a(k) = 0$  otherwise. The service rate is constant and equal

to 1. Assume that  $\nu \in (1/2, 1)$ . With the controlled arrival process  $\hat{a}(k)$ , we want to achieve a throughput  $E[\hat{a}(k)] \geq 1 - \epsilon$  for a given small  $\epsilon > 0$ . A “threshold policy” based on a threshold  $T$  is proposed below:

- When  $Q(k) > T$ , reject all arrivals;
- When  $Q(k) = T$ , accept one arrival w.p.  $p_1$ , accept two arrivals w.p.  $p_2$ , and reject all of them otherwise.
- When  $Q(k) < T$ , accept all arrivals.

Defining  $\pi_i$  as the steady-state probability that  $Q(k) = i$  ( $0 \leq i \leq T + 1$ ) for the resulting Markov chain, the local balance equations are given below:

$$\begin{aligned} \pi_i \nu &= \pi_{i+1} (1 - \nu), \quad \forall 0 \leq i \leq T - 2; \\ \pi_{T-1} \cdot p_1 \nu &= \pi_T (1 - (p_1 + p_2) \nu); \\ \pi_T \cdot p_2 \nu &= \pi_{T+1}; \\ \sum_{i=0}^{T+1} \pi_i &= 1. \end{aligned} \tag{25}$$

Combining these equations with the throughput requirement, we get

$$\nu \left[ 2 \sum_{i=0}^{T-1} \pi_i + \pi_T (2p_2 + p_1) \right] = 1 - \epsilon, \tag{26}$$

and one can finally show that (ignoring detailed calculations)

$$T = \frac{\log(1/\epsilon) + \log C(\epsilon)}{\log\left(\frac{\nu}{1-\nu}\right)},$$

where

$$C(\epsilon) \triangleq \frac{(2\nu - 1 + \epsilon)(1 - \nu(p_1 + p_2))}{\nu(2 - 2(1 - \nu)p_2 - p_1)}.$$

The above result further implies that  $\bar{Q} \sim \Theta(\log(1/\epsilon))$ . we can also see that as  $\nu \rightarrow 1$ ,  $T \rightarrow 0$ , which is consistent with the fact the lower bound given in Theorem 2 goes to 0 as the “zero arrival probability”  $\varphi \rightarrow 0$ .

Another example showing the tightness of an  $\Omega(\log(1/\epsilon))$  bound is the dynamic packet dropping algorithm in [25] (note that this universal lower bound is proved based on a strict convexity assumption as mentioned before in Subsection IV-A).

## V. CLICK-THROUGH RATE MAXIMIZATION PROBLEM

In this section, we consider another online ads model, in which the objective is to maximize the long-term average total click-through rate of all queries. Instead of average budget, client  $i$  specifies in the contract an average “*impression requirement*”  $m_i$ , which is the minimum number of times an ad of this client should be posted by the service provider per “*requirement cycle*” (equal to  $N$  time slots) on average. The other parameters are the same as in the model proposed in Section I for the revenue maximization problem.

The corresponding optimization formulation now becomes

$$\max_{\mathbf{p} \in \mathcal{F}} \bar{J}(\mathbf{p}) = \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_{i,s} M_{is} c_{qis} \tag{27}$$

where the feasible set  $\mathcal{F}$  is characterized by

$$N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_s M_{is} \geq m_i, \quad \forall i; \quad (28)$$

$$0 \leq p_{qM} \leq 1, \quad \forall q, M \in \mathcal{M}_q; \quad (29)$$

$$\sum_{M \in \mathcal{M}_q} p_{qM} \leq 1, \quad \forall q. \quad (30)$$

Different from the revenue maximization problem, here the feasible set can become empty if some  $m_i$  is too high. Basically, without constraint (28),  $\mathcal{F}$  is relaxed to

$$\mathcal{F}_0 \triangleq \{\mathbf{p} : 0 \leq p_{qM} \leq 1, \forall q, M \in \mathcal{M}_q; \sum_{M \in \mathcal{M}_q} p_{qM} \leq 1, \forall q\}. \quad (31)$$

We can then define the following capacity region which characterizes how large the average number of impressions can be achieved for each client per requirement cycle:

$$\mathcal{C} \triangleq \left\{ \boldsymbol{\mu} : \mu_i = N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_s M_{is}, \forall i, \text{ s.t. } \mathbf{p} \in \mathcal{F}_0 \right\}.$$

Clearly,  $\mathbf{m} \in \mathcal{C}$  must hold to ensure the existence of a solution for the above optimization problem.

Through a similar approach as in Subsection III-A, we can write down a similar online algorithm based on the same stochastic model as defined in Subsection III-B. We define  $\mathbf{q}(k) \triangleq \{\tilde{q}(t), \text{ for } kN \leq t \leq kN + N - 1\}$ . Similar to  $\tilde{b}_i(k)$ ,  $\tilde{m}(k) = \lceil m_i \rceil$  w.p.  $m_i - \lfloor m_i \rfloor$  and  $\tilde{m}(k) = \lfloor m_i \rfloor$  otherwise.

**Online Algorithm:** (in each requirement cycle  $k \geq 0$ )

In each time slot  $t \in [kN, kN + N - 1]$ , if  $\tilde{q}(t) > 0$ , choose the assignment matrix

$$\tilde{M}^*(t, \tilde{q}(t), \mathbf{Q}(k)) \in \arg \max_{M \in \mathcal{M}_{\tilde{q}(t)}} \sum_{i,s} M_{is} \left( \frac{C_{\tilde{q}(t)is}}{\epsilon} + Q_i(k) \right). \quad (32)$$

At the end of requirement cycle  $k$ , for each client  $i$ , update

$$Q_i(k+1) = [Q_i(k) + \tilde{m}(k) - S_i(k, \mathbf{Q}(k), \mathbf{q}(k))]^+,$$

where

$$S_i(k, \mathbf{Q}(k), \mathbf{q}(k)) \triangleq \sum_{t=kN}^{kN+N-1} \sum_s [\tilde{M}^*(t, \tilde{q}(t), \mathbf{Q}(k))]_{is}. \quad (33)$$

In real online advertising business, some clients may only have short-term contracts, i.e., clients may not be interested in the average number of impressions per time slot but may be interested in a minimum number of impressions in a given duration (such as a day). Further, query arrivals may not form a stationary process. In fact, they are more likely to vary depending on the time of day. These extensions are considered in Appendix E. Such extensions also make sense for the revenue maximization model considered in the previous sections, but the approach is similar to Appendix E and so will not be considered here.

### A. Performance Evaluation

$S_i(k, \mathbf{Q}(k), \mathbf{q}(k))$  defined in (33) represents the actual number of impressions for client  $i$ 's ads during requirement cycle  $k$ . The queue length increases when the average impression requirements in a particular requirement cycle cannot be fulfilled. Hence, a positive queue represents accumulated ‘‘credits,’’ which enhances the chance of being assigned with a webpage slot in the future, much like a negative queue in the revenue maximization problem. We thus call this queue a ‘‘credit queue.’’

Unlike the revenue maximization problem in which an  $O(1/\epsilon)$  upper bound on the transient queue length is automatically imposed by the online algorithm, here we need to prove the stability of the queues and show an upper bound on the mean queue length. Since  $\{\mathbf{Q}(k)\}$  defines an irreducible and aperiodic Markov chain, in order to prove its stability (positive recurrence), we will first bound the expected drift of  $\mathbf{Q}(k)$  for a suitable Lyapunov function.

*Lemma 2:* Consider the Lyapunov function  $V(\mathbf{Q}) = \frac{1}{2} \sum_i Q_i^2$ . For any  $\epsilon > 0$  and each requirement cycle  $k$ ,

$$E[V(\mathbf{Q}(k+1)) | \mathbf{Q}(k) = \mathbf{Q}] - V(\mathbf{Q}) \leq \frac{D_3}{\epsilon} + D_1 - D_2 \sum_i Q_i. \quad (34)$$

Here,

$$D_1 \triangleq \frac{1}{2} \left( N(N-1)L^2 + NL + \sum_i [m_i]^2 (m_i - [m_i]) + [m_i]^2 (1 - m_i + [m_i]) \right), \quad (35)$$

where  $L$  is the number of webpage slots;

$$D_2 \triangleq \min_i \left\{ N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} \hat{p}_{qM} \sum_s M_{is} - m_i \right\}, \quad (36)$$

for some  $\hat{\mathbf{p}} \in \mathcal{F}$  such that  $D_2 > 0$ ; and

$$D_3 \triangleq N \cdot \max_{\mathbf{p} \in \mathcal{F}_0} \bar{J}(\mathbf{p}) \quad (37)$$

where  $\mathcal{F}_0$  is defined in (31).  $\diamond$

The proof is similar to the proof of Lemma 1 with some modifications in the final steps, which will be given briefly in Appendix D. With this lemma, we can conclude that  $\mathbf{Q}(k)$  is positive recurrent because the expected Lyapunov drift is negative except for a finite set of values of  $\mathbf{Q}(k)$ , according to Foster-Lyapunov theorem ([2], [23]).

*Remark 4:* Note that compared to the definition of  $B_2$  in (11) of Lemma 1 where  $B_2 \geq 0$ ,  $D_2$  needs to be strictly positive in order to prove the stability of queues. Such a  $\hat{\mathbf{p}}$  in the definition of  $D_2$  can always be found unless  $\mathcal{F}$  is a degenerate set with at most one element.  $\diamond$

The stability of the queues directly implies the following corollary:

*Corollary 2 (Overservices in the long term):*

$$\lim_{K \rightarrow \infty} E \left[ \frac{1}{K} \sum_{k=1}^K S_i(k, \mathbf{Q}(k), \mathbf{q}(k)) \right] \geq m_i, \quad \forall i. \quad \diamond$$

In addition to proving stability, Lemma 2 will be used to evaluate the upper bound on the expected total queue length in the steady state, as shown in the following theorem:

*Theorem 4:* Under the online algorithm,

$$E \left[ \sum_i Q_i(\infty) \right] \leq \frac{1}{D_2^*} \left( D_1 + \frac{D_3}{\epsilon} \right), \quad (38)$$

where  $D_1$  and  $D_3$  are respectively defined in (35) and (37);  $D_2^*$  is defined as

$$D_2^* \triangleq \max_{\mathbf{p} \in \mathcal{F}_0} D_2(\mathbf{p}). \quad (39)$$

where  $D_2$  is defined in (36) (regarded as a function of  $\mathbf{p}$ ).  $\diamond$

*Proof:* Averaging both sides of inequality (34) over  $0 \leq k \leq K - 1$ , taking  $K \rightarrow \infty$  and doing some simple algebra, one obtains

$$\limsup_{K \rightarrow \infty} \frac{1}{K} \sum_{k=0}^{K-1} E \left[ \sum_i Q_i(k) \right] \leq \frac{1}{D_2} \left( D_1 + \frac{D_3}{\epsilon} \right).$$

The LHS equals to  $E[\sum_i Q_i(\infty)]$  according to Theorem 15.0.1 in [23]. The RHS is minimized through maximizing  $D_2$  over all  $\mathbf{p} \in \mathcal{F}_0$  (which will certainly satisfy  $\mathbf{p} \in \mathcal{F}$  and  $D_2 > 0$ ). This completes our proof.  $\blacksquare$

The following theorem shows that the online algorithm proposed above achieves a long-term average click-through rate within  $O(\epsilon)$  of the offline optimum. The proof is similar to the one for Theorem 1 and hence will be omitted.

*Theorem 5:* For any  $\epsilon > 0$ ,

$$0 \leq \lim_{K \rightarrow \infty} E \left[ \bar{J}(\mathbf{p}^*) - \frac{1}{KN} \sum_{k=0}^{K-1} J(k) \right] \leq \frac{D_1 \epsilon}{N},$$

for some constant  $D_1 > 0$  (defined in (35) in Lemma 2). Here,  $J(k)$  is defined as the total number of click-through events within requirement cycle  $k$ .  $\diamond$

### B. Customizing Impression Requirements $\{m_i\}$ Based on Query Arrival Rates $\{\nu_q\}$

Since a positive queue measures how much the service provider “owes” a client, reducing the coefficient of the  $1/\epsilon$  term in the upper bound on the mean queue length becomes important. Besides, we also need to guarantee  $\mathbf{m} \in \mathcal{C}$ . In order to handle these two issues, we introduce an approach to customizing  $\{m_i\}$  based on known (or estimated) query arrival rates  $\{\nu_q\}$ ,

Replacing  $D_2^*$  in Theorem 4 by a common  $D_2$  defined in equation (36), if we want the expected total queue length to be upper bounded by  $Q_{max}$ , it suffices to let

$$D_2 \geq \xi \triangleq \frac{1}{Q_{max}} \left( D_1 + \frac{D_3}{\epsilon} \right), \quad (40)$$

where  $D_3$  is already determined, and  $D_1$  does not matter much given a small  $\epsilon$  although it includes unknown  $\{m_i\}$ . We then solve the following optimization problem to determine  $\{m_i\}$ :

$$\begin{aligned} & \max_{\mathbf{p} \in \mathcal{F}_0, \mathbf{m}} \sum_i \log m_i \\ \text{s.t. } & N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM} \sum_s M_{is} - m_i \geq \xi, \quad \forall i. \end{aligned}$$

Here we use  $\sum_i \log m_i$  as the objective function in order to guarantee a unique optimal solution and impose a certain fairness rule called ‘‘proportional fairness’’ (see e.g. [17]). Note that  $\xi$  cannot be set too large (i.e.,  $Q_{max}$  cannot be set too small), otherwise there may not exist a feasible solution.

Naturally, a question would arise: now that we need to solve some mathematical programming like the above one based on knowledge of query arrival rates, why not also directly solve the original linear programming in (27) and use the offline optimal solution  $\mathbf{p}^*$  to assign ads? The answer to this is similar to the max-weight algorithm for wireless networks. In [27] and [29], it has been shown that adaptive algorithms lead to much better queueing performance compared to static offline algorithms. We verify this assertion in our context through simulations in the next subsection.

### C. Queue Update in a Faster Time Scale

In the original algorithm, the queue length is updated only at the end of each requirement cycle and used in the max-weight matching for the next whole requirement cycle. The longer a requirement cycle lasts, the more obsolete the queue length information becomes, so with a large  $N$ , short-term performances may not be so good even if long-term performances are still guaranteed.

We then propose a solution which updates queue lengths in a faster time scale. Specifically, we divide each requirement cycle into  $T$  *queueing cycles* with equal lengths (assuming  $N/T \in \mathcal{Z}^+$  without loss of generality). We use  $\{\hat{\mathbf{Q}}(k, \tau) : 0 \leq \tau \leq T\}_{k \geq 0}$  to denote this new queueing system and assume  $\hat{\mathbf{Q}}(-1, T) = \mathbf{0}$ . At the beginning of each requirement cycle  $k$  before any decision, update

$$\hat{\mathbf{Q}}(k, 0) = \hat{\mathbf{Q}}(k-1, T) + \tilde{\mathbf{m}}(k),$$

and at the end of the  $\tau^{\text{th}}$  queueing cycle within this requirement cycle ( $1 \leq \tau \leq T$ ), for all client  $i$ ,

$$\hat{Q}_i(k, \tau) = \left[ \hat{Q}_i(k, \tau-1) - \sum_{t=kN+(\tau-1)\frac{N}{T}}^{kN+\tau\frac{N}{T}-1} \sum_s [\tilde{M}^*(t, \tilde{q}(t), \hat{\mathbf{Q}}(k, \tau))]_{is} \right]^+.$$

Since  $\|\hat{\mathbf{Q}}(k, T) - \mathbf{Q}(k)\| \leq B$  for some constant  $B$  independent of the queue lengths, it can be shown that the long-term performances evaluated in Subsection V-A are still guaranteed (the idea behind such a proof would be similar to the one in [7] and so is omitted).

Next, we use simulations to compare three different algorithms, namely a randomized algorithm following the offline optimal solution (labeled as OPT) and two versions of our online algorithm ‘‘max-weight matching’’ with and without ‘‘fast queue update’’ respectively (labeled as MWM-Fast and MWM respectively). In each scenario we test, all the parameters are randomly

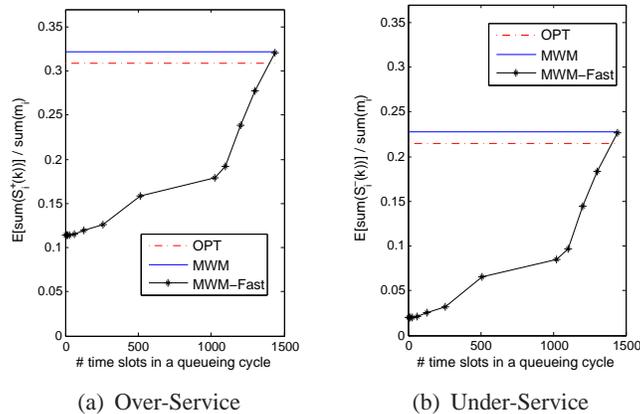


Fig. 3: Average overall over-service and under-service (normalized by the total impression requirement) impacted by the “fast queue update”

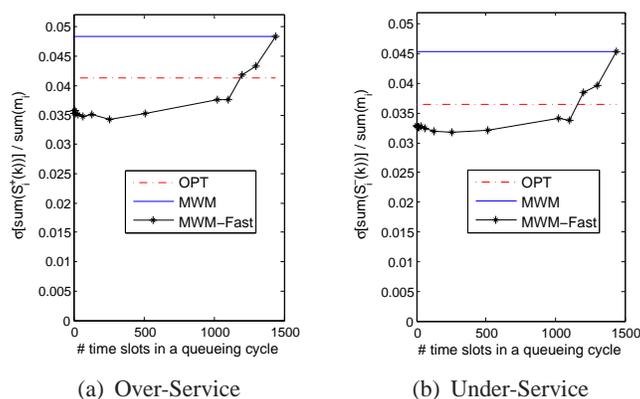


Fig. 4: The standard variance of overall over-service and under-service (normalized by the total impression requirement) impacted by the “fast queue update”

generated. The impression requirements  $\{m_i\}$  are chosen through the approach in Subsection V-B.

We take an example scenario with 2 webpage slots, 5 keywords and 10 clients. The probability that a query arrives in a time slot equals 0.7. Specifically, for the five keywords, the query arrival rates are  $\nu = [0.2364, 0.0594, 0.1669, 0.0714, 0.1659]$ . Table I shows the click-through rates for the ten clients ( $C_1 \sim C_{10}$ ) corresponding to each keyword ( $q_1 \sim q_5$ ), on webpage slots 1 and 2 respectively (a zero click-through rate indicates that the corresponding client is not related to this keyword). We use  $N = 1440$  (say, one time slot is one minute and one requirement cycle is one day),  $\epsilon = 10^{-4}$  and  $Q_{max} = 20/\epsilon$  (recall that  $Q_{max}$  is used to set up an upper bound on the mean queue length by the heuristic in Subsection V-B). The simulation has been run for 1000 requirement cycles.

To compare the performances of all the three algorithms, instead of considering the long-term performance requirements that we have used in the theory, we introduce two new metrics: over-service  $S_i^+(k) \triangleq [S_i(k) - \tilde{m}_i(k)]^+$  and under-service  $S_i^-(k) \triangleq [\tilde{m}_i(k) - S_i(k)]^+$  to client  $i$

	$C_1$	$C_2$	$C_3$	$C_4$	$C_5$	$C_6$	$C_7$	$C_8$	$C_9$	$C_{10}$
Webpage Slot 1										
$q_1$	0	0.519	0.973	0	0.649	0	0	0	0.800	0
$q_2$	0	0	0	0.340	0	0	0.952	0	0	0
$q_3$	0.982	0.645	0.856	0.461	0.190	0	0.369	0.669	0.156	0
$q_4$	0.423	0	0	0	0.599	0	0.179	0	0.471	0.094
$q_5$	0	0	0	0.875	0	0.518	0	0	0	0
Webpage Slot 2										
$q_1$	0	0.235	0.421	0	0.536	0	0	0	0.067	0
$q_2$	0	0	0	0.312	0	0	0.050	0	0	0
$q_3$	0.118	0.248	0.194	0.222	0.036	0	0.158	0.252	0.092	0
$q_4$	0.296	0	0	0	0.020	0	0.124	0	0.032	0.060
$q_5$	0	0	0	0.826	0	0.330	0	0	0	0

TABLE I: Click-through rates for all the clients' ads

during requirement cycle  $k$ . Note that these metrics measure deviations from the guarantees over short time scales and so are more stringent requirements than the long-term guarantees used in the theory.

We show respectively in Figures 3(a) and 3(b) that the average overall over-service and under-service normalized by the total impression requirement, i.e.,  $E[\sum_i S_i^+(k)]/\sum_i m_i$  and  $E[\sum_i S_i^-(k)]/\sum_i m_i$ , are both reduced by the fast queue update. Similarly, a ‘‘variance reduction’’ effect is shown by the fast queue update based on the statistics  $\sqrt{\text{var}[\sum_i S_i^+(k)]/\sum_i m_i}$  and  $\sqrt{\text{var}[\sum_i S_i^-(k)]/\sum_i m_i}$ , respectively in Figures 4(a) and 4(b). In terms of the overall click-through rate, our simulation has verified that the three algorithms achieve approximately the same performance (the figure is omitted here) and further demonstrated in Figure 5 that the fast queue update can also reduce its variance. Note that these performances of each individual client also improve and we simply omit the figures here.

Observed from Figure 6, the offline optimal solution leads to very unstable queue dynamics. This essentially arises from the fact that the algorithm operates on an optimal point  $\mathbf{p}^*$  for which some inequalities in constraint (28) may be tight. In contrast, our online algorithm guarantees the stability of queues, and the faster the queues update, the more stable the queue dynamics become (as an example we use  $T = 24$ , i.e., the number of time slots per queueing cycle equals 60). This is consistent with the above results which show a reduction of over-service and under-service in both mean and variance since these metrics directly measure the level of deviations around the equilibrium point of each stable queue.

*Remark 5:* While a long-term client may only be concerned with average performances, a short-term client cares about both mean (the average level for all the clients of its type) and variance (related to its own individual level), especially for the performances of under-service and click-through rate.<sup>5</sup> All of these are well handled by our online algorithm with fast queue updates.

## VI. CONCLUSIONS

In this paper, we propose a stochastic model to describe how search service providers charge client companies based on users' queries for the keywords related to these companies' ads

<sup>5</sup>Over-service are cared about by the online ads service provider.

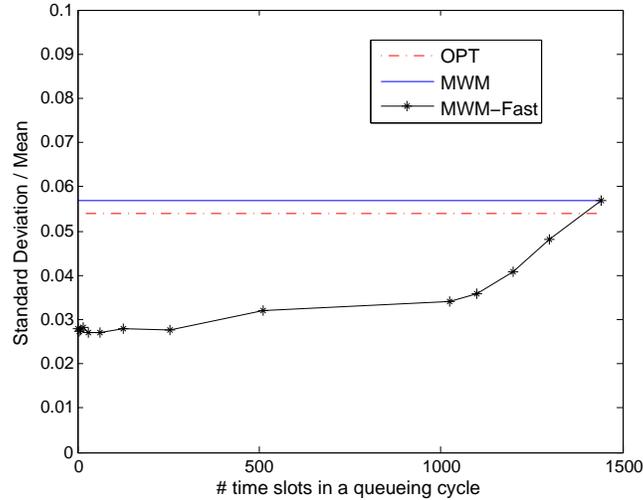


Fig. 5: The “standard variance to mean ratio” of overall click-through rate impacted by the “fast queue update”

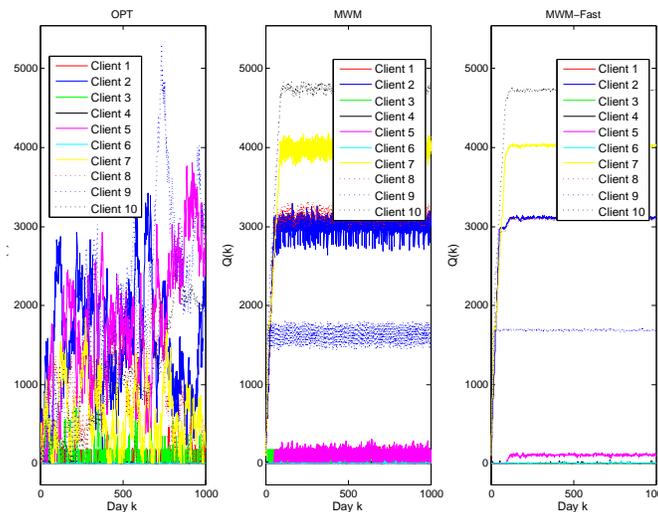


Fig. 6: Queue dynamics under three algorithms

by using certain advertisement assignment strategies. We formulate an optimization problem to maximize the long-term average revenue for the service provider under each client’s long-term average budget constraint, and design an online algorithm which captures the stochastic properties of users’ queries and click-through behaviors. We solve the optimization problem by making connections to scheduling problems in wireless networks, queueing theory and stochastic networks. Our online algorithm is entirely oblivious to query arrivals and fully adaptive, so even non-stationary query arrival patterns and short-term clients can be handled.

With a small customizable parameter  $\epsilon$  which is the step size used in each iteration of the online algorithm, we have shown that our online algorithm achieves a long-term average revenue which is within  $O(\epsilon)$  of the optimal revenue and the overdraft level of this algorithm is upper

bounded by  $O(1/\epsilon)$ . By allowing negative values for the length of overdraft queues, we can eliminate overdraft.

When estimated click-through rates instead of true ones are used in our online algorithm, we show that the achievable fraction of the offline optimal revenue is lower bounded by  $\frac{1-\Delta}{1+\Delta}$ , where  $\Delta$  is the relative error in click-through rate estimation.

We also show that in the long run, an expected overdraft level of  $\Omega(\log(1/\epsilon))$  is unavoidable (a universal lower bound) under any stationary ad assignment algorithm which achieves a long-term average revenue within  $O(\epsilon)$  of the offline optimum. The tightness of this universal lower bound is also shown for a simple queueing model using a threshold policy.

In another optimization formulation where the objective is to maximize the long-term average click-through rate and the constraints include a minimum impression requirement for each client, we further propose an approach to set impression requirements which make the contract feasible and limit the average accumulated under-service to clients. Simulations show that making queues update in a faster time scale will reduce both over-service and under-service, which benefits a system involving short-term clients.

## REFERENCES

- [1] S. Agrawal, Z. Wang, and Y. Ye. A dynamic near-optimal algorithm for online linear programming. *Submitted to Mathematics of Operations Research*, 2010.
- [2] S. Asmussen. *Applied probability and queues*. Springer-Verlag, 2003.
- [3] R. A. Berry and R. G. Gallager. Communication over fading channels with delay constraints. *IEEE Transactions on Information Theory*, 48(5):1135–1149, May 2002.
- [4] V. Borkar. *Stochastic approximation: a dynamical systems viewpoint*. Cambridge University Press, 2008.
- [5] N. Buchbinder, K. Jain, and J. S. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. *In Proc. of the 15th annual European Conference on Algorithms (ESA), Lecture Notes in Computer Science (LNCS)*, 4698:253–264, 2007.
- [6] N. R. Devenur and T. P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. *In Proc. of the 10th ACM Conference on Electronic Commerce (EC)*, pages 71–78, Stanford, CA, USA, 2009.
- [7] A. Eryilmaz, R. Srikant, and J. Perkins. Stable scheduling policies for fading wireless channels. *IEEE/ACM Transactions on Networking*, 13(2):411–424, Apr. 2005.
- [8] J. Feldman, M. Henzinger, N. Korula, V. S. Mirrokni, and C. Stein. Online stochastic packing applied to display ad allocation. *Algorithms–ESA, Lecture Notes in Computer Science*, 6346:182–194, 2010.
- [9] J. Feldman, N. Korula, V. Mirrokni, S. Muthukrishnan, and M. Pál. Online ad assignment with free disposal. *In the 5th Workshop on Internet and Network Economics (WINE), Lecture Notes in Computer Science (LNCS)*, 5929:374–385, Dec. 2009.
- [10] J. Feldman and S. Muthukrishnan. Algorithmic methods for sponsored search auctions. *SIGMETRICS Tutorial. Chapter in Performance Modeling and Engineering*, 2008.
- [11] L. Georgiadis, M. J. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, 1(1):1–144, 2006.
- [12] L. Huang, S. Moeller, M. Neely, and B. Krishnamachari. LIFO-Backpressure achieves near optimal utility-delay tradeoff. *In Proc. of 9th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2011.
- [13] G. Iyengar and A. Kumar. Characterizing optimal adword auctions. *Technical Report*, Nov. 2006. <http://arxiv.org/abs/cs.GT/0611063>.
- [14] J. J. Jaramillo and R. Srikant. Optimal scheduling for fair resource allocation in ad hoc networks with elastic and inelastic traffic. *In Proc. of IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010.
- [15] B. Kalyanasundaram and K. Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1–2):319–325, 2000.
- [16] R. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for on-line bipartite matching. *In Proc. of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358, 1990.
- [17] F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *The Journal of the Operational Research Society*, 49(3):237–252, 1998.
- [18] X. Lin, N. B. Shroff, and R. Srikant. A tutorial on cross-layer optimization in wireless networks. *IEEE Journal on Selected Areas in Communications*, 24(8):1452–1463, Aug. 2006.
- [19] M. Mahdian, H. Nazerzadeh, and A. Saberi. Allocating online advertisement space with unreliable estimates. *In Proc. of the 8th ACM conference on Electronic Commerce (EC)*, pages 288–294, 2007.
- [20] A. Mehta, A. Saberi, U. V. Vazirani, and V. Vazirani. Adwords and generalized online matching. *In Proc. of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 264–273, 2005.

- [21] A. Mehta, A. Saberi, U. V. Vazirani, and V. Vazirani. Adwords and generalized online matching. *Journal of the ACM*, 54(5), Oct. 2007. Article No. 22.
- [22] I. Menache, A. Ozdaglar, R. Srikant, and D. Acemoglu. Dynamic online-advertising auctions as stochastic scheduling. In *Proc. of the Workshop on the Economics of Networks, Systems and Computation (NetEcon)*, Stanford, CA, USA, Jul. 2009.
- [23] S. Meyn and R. Tweedie. *Markov chains and stochastic stability*. Springer-Verlag, London, 1993.
- [24] M. J. Neely. Optimal energy and delay tradeoffs for multiuser wireless downlinks. *IEEE Transactions on Information Theory*, 53(9):3095–3113, 2007.
- [25] M. J. Neely. Intelligent packet dropping for optimal energy-delay tradeoffs in wireless downlinks. *IEEE Transactions on Automatic Control*, 54(3):565–579, Mar. 2009.
- [26] D. Shah and D. Wischik. Lower bound and optimality in switched networks. In *Proc. of 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 1262–1269, 2008.
- [27] S. Shakkottai. Effective capacity and QoS for wireless scheduling. *IEEE Transactions on Automatic Control*, 53(3):749–761, Apr. 2008.
- [28] B. R. Tan and R. Srikant. Online advertisement, optimization and stochastic networks. In *Proc. of IEEE Conference on Decision and Control (CDC)*, Dec. 2011.
- [29] L. Ying, R. Srikant, A. Eryilmaz, and G. Dullerud. A large deviations analysis of scheduling in wireless networks. *IEEE Transactions on Information Theory*, 52(11):5088–5098, Nov. 2006.

## APPENDIX

### A. Proof of Lemma 1

$$\begin{aligned}
& E[V(\mathbf{Q}(k+1)) | \mathbf{Q}(k) = \mathbf{Q}] - V(\mathbf{Q}) \\
&= \frac{1}{2} E \left[ \sum_i \left( \left[ Q_i + A_i(k, \mathbf{Q}, \mathbf{u}(k)) - \tilde{b}_i(k) \right]^+ \right)^2 - Q_i^2 \right] \\
&\leq \frac{1}{2} E \left[ \sum_i \left( Q_i + A_i(k, \mathbf{Q}, \mathbf{u}(k)) - \tilde{b}_i(k) \right)^2 - Q_i^2 \right] \\
&= E \left[ \sum_i Q_i \left( A_i(k, \mathbf{Q}, \mathbf{u}(k)) - \tilde{b}_i(k) \right) + \frac{1}{2} \sum_i \left( A_i(k, \mathbf{Q}, \mathbf{u}(k)) - \tilde{b}_i(k) \right)^2 \right] \\
&\leq \sum_i Q_i (\lambda_i(k, \mathbf{Q}) - b_i) + \frac{1}{2} \sum_i (E[A_i^2(k, \mathbf{Q}, \mathbf{u}(k))] + E[\tilde{b}_i^2(k)]), \tag{41}
\end{aligned}$$

where it was already defined in equation (8) that for all  $i$ ,

$$A_i(k, \mathbf{Q}(k), \mathbf{u}(k)) = \sum_{t=kN}^{kN+N-1} \sum_s [\tilde{M}^*(t, \tilde{q}(t), \mathbf{Q}(k))]_{is} \cdot \tilde{c}_{\tilde{q}(t)is}(t) \cdot r_{\tilde{q}(t)i}.$$

and we further define

$$\lambda_i(k, \mathbf{Q}(k)) \triangleq E[A_i(k, \mathbf{Q}(k), \mathbf{u}(k)) | \mathbf{Q}(k)] = N \sum_q \nu_q \sum_s [\tilde{M}^*(q, t, \mathbf{Q}(k))]_{is} c_{qis} r_{qi}.$$

Since each client can at most get one webpage slot for each query, we can further bound

$$\sum_i A_i^2(k, \mathbf{Q}, \mathbf{u}(k)) \leq (N(N-1)L^2 + NL) (\arg \max_{q,i,s} \{c_{qis} r_{qi}\})^2.$$

Besides,

$$E[b_i^2(k)] = \lceil b_i \rceil^2 \rho_i + \lfloor b_i \rfloor^2 (1 - \rho_i) = \lceil b_i \rceil^2 (b_i - \lfloor b_i \rfloor) + \lfloor b_i \rfloor^2 (1 - b_i + \lfloor b_i \rfloor).$$

Thus, by defining

$$B_1 \triangleq \frac{1}{2} \left( (N(N-1)L^2 + NL) (\arg \max_{q,i,s} \{c_{qis} r_{qi}\})^2 + \sum_i [b_i]^2 (b_i - [b_i]) + [b_i]^2 (1 - b_i + [b_i]) \right),$$

and continuing from inequality (41), we have

$$\begin{aligned} & E[V(\mathbf{Q}(k+1)) | \mathbf{Q}(k) = \mathbf{Q}] - V(\mathbf{Q}) \\ & \leq N \sum_q \nu_q \sum_{i,s} Q_i [\tilde{M}^*(q, t, \mathbf{Q})]_{is} c_{qis} r_{qi} - \sum_i Q_i b_i + B_1 \\ & = -N \sum_q \nu_q \sum_{i,s} \left( \frac{1}{\epsilon} - Q_i \right) [\tilde{M}^*(q, t, \mathbf{Q})]_{is} c_{qis} r_{qi} \\ & \quad + \frac{N}{\epsilon} \sum_q \nu_q \sum_{i,s} [\tilde{M}^*(q, t, \mathbf{Q})]_{is} c_{qis} r_{qi} + B_1 - \sum_i Q_i b_i \\ & = -N \sum_q \nu_q \sum_{i,s} \left( \frac{1}{\epsilon} - Q_i \right) [\tilde{M}^*(q, t, \mathbf{Q})]_{is} c_{qis} r_{qi} \\ & \quad + \frac{N}{\epsilon} \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q})) + B_1 - \sum_i Q_i b_i \end{aligned} \quad (42)$$

$$\begin{aligned} & \stackrel{(a)}{\leq} -N \sum_q \nu_q \sum_{i,s} \left( \frac{1}{\epsilon} - Q_i \right) \sum_{M \in \mathcal{M}_q} p_{qM}^* M_{is} c_{qis} r_{qi} + \frac{N}{\epsilon} \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q})) + B_1 - \sum_i Q_i b_i \\ & = -\frac{N}{\epsilon} (\bar{R}(\mathbf{p}^*) - \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q}))) + B_1 \\ & \quad - \sum_i Q_i \cdot \left( b_i - N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM}^* \sum_s M_{is} c_{qis} r_{qi} \right), \end{aligned} \quad (43)$$

where inequality (a) holds because equation (6) in the online algorithm is equivalent to

$$\forall q, \tilde{\mathbf{p}}_q^*(k, \mathbf{Q}(k)) \in \arg \max_{\substack{\{p_{qM}^*, \\ M \in \mathcal{M}_q\}}} \sum_{M \in \mathcal{M}_q} p_{qM} \sum_{i,s} M_{is} c_{qis} r_{qi} \left( \frac{1}{\epsilon} - Q_i(k) \right), \quad (44)$$

which means that evaluating the objective function in (44) with  $\mathbf{p} = \mathbf{p}^*$  cannot achieve a larger value. Letting

$$B_2 \triangleq \min_i \{ b_i - N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM}^* \sum_s M_{is} c_{qis} r_{qi} \},$$

from inequality (43), we finally obtain

$$E[V(\mathbf{Q}(k+1)) | \mathbf{Q}(k) = \mathbf{Q}] - V(\mathbf{Q}) \leq -\frac{N}{\epsilon} (\bar{R}(\mathbf{p}^*) - \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q}))) + B_1 - B_2 \sum_i Q_i.$$

### B. Proof of Theorem 1

The first inequality which shows that the online algorithm cannot do better than the offline optimal solution is too obvious, so we just ignore it here (proving it in a very rigorous way is also very easy, after defining the “per-client revenue region” in Subsection IV-B and then using the fact that the average revenue vector  $\lambda$  corresponding to our online algorithm falls inside that region, according to inequality (9) which is implied by stability).

We now focus on the second inequality, i.e., the  $O(\epsilon)$  convergence bound. From Lemma 1,

$$\begin{aligned} & E [\bar{R}(\mathbf{p}^*) - \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q}(k)))] \\ & \leq \frac{\epsilon}{N} \cdot E \left[ B_1 - B_2 \sum_i \mathbf{Q}(k) + V(\mathbf{Q}(k)) - E[V(\mathbf{Q}(k+1))|\mathbf{Q}(k)] \right] \\ & \leq \frac{\epsilon}{N} \cdot (B_1 - E[V(\mathbf{Q}(k))] - E[V(\mathbf{Q}(k+1))]), \end{aligned}$$

Adding the terms for  $0 \leq k \leq K-1$  and dividing by  $K$ , we get

$$\begin{aligned} \frac{1}{K} \sum_{k=0}^{K-1} E [\bar{R}(\mathbf{p}^*) - \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q}(k)))] & \leq \frac{\epsilon}{N} \left( B_1 - \frac{E[V(\mathbf{Q}(K))]}{K} + \frac{V(\mathbf{Q}(0))}{K} \right) \\ & \leq \frac{\epsilon}{N} \left( B_1 + \frac{V(\mathbf{Q}(0))}{K} \right). \end{aligned}$$

Since  $V(\mathbf{Q}(0)) < \infty$ , we get the following limit expression:

$$\lim_{K \rightarrow \infty} \frac{1}{K} \sum_{k=0}^{K-1} E [\bar{R}(\mathbf{p}^*) - \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q}(k)))] \leq \frac{B_1 \epsilon}{N}. \quad (45)$$

Finally, because

$$E [N\bar{R}(\mathbf{p}^*) - R(k)] = E [E [N\bar{R}(\mathbf{p}^*) - R(k)|\mathbf{Q}(k)]] = N \cdot E [\bar{R}(\mathbf{p}^*) - \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q}(k)))] ,$$

inequality (45) is equivalent to

$$\lim_{K \rightarrow \infty} E \left[ \bar{R}(\mathbf{p}^*) - \frac{1}{KN} \sum_{k=0}^{K-1} R(k) \right] \leq \frac{B_1 \epsilon}{N}.$$

### C. Proof of Corollary 1

Continuing from inequality (42) in Appendix A (the proof of Lemma 1), we get

$$\begin{aligned} & E[V(\mathbf{Q}(k+1))|\mathbf{Q}(k) = \mathbf{Q}] - V(\mathbf{Q}) \\ & \stackrel{(a)}{\leq} -\frac{1}{1+\Delta} N \sum_q \nu_q \sum_{i,s} \left( \frac{1}{\epsilon} - Q_i \right) [\tilde{M}^*(q, t, \mathbf{Q})]_{is} \hat{c}_{qis} r_{qi} + \frac{N}{\epsilon} \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q})) + B_1 - \sum_i Q_i b_i \\ & \stackrel{(b)}{\leq} -\frac{1}{1+\Delta} N \sum_q \nu_q \sum_{i,s} \left( \frac{1}{\epsilon} - Q_i \right) \sum_{M \in \mathcal{M}_q} p_{qM}^* M_{is} \hat{c}_{qis} r_{qi} + \frac{N}{\epsilon} \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q})) + B_1 - \sum_i Q_i b_i \end{aligned}$$

$$\begin{aligned}
&\stackrel{(c)}{\leq} -\frac{1-\Delta}{1+\Delta}N \sum_q \nu_q \sum_{i,s} \left(\frac{1}{\epsilon} - Q_i\right) \sum_{M \in \mathcal{M}_q} p_{qM}^* M_{is} c_{qis} r_{qi} + \frac{N}{\epsilon} \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q})) + B_1 - \sum_i Q_i b_i \\
&= -\frac{N}{\epsilon} \left( \frac{1-\Delta}{1+\Delta} \bar{R}(\mathbf{p}^*) - \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q})) \right) + B_1 \\
&\quad - \sum_i Q_i \cdot \left( b_i - \frac{1-\Delta}{1+\Delta} N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM}^* \sum_s M_{is} c_{qis} r_{qi} \right). \tag{46}
\end{aligned}$$

Here, inequalities (a) and (c) hold respectively because  $\hat{\mathbf{c}} \leq \mathbf{c}(1 + \Delta)$  and  $\hat{\mathbf{c}} \geq \mathbf{c}(1 - \Delta)$ , with the fact that all the coefficients in this summation are nonnegative. Inequality (b) holds because equation (12) in the online algorithm with estimated click-through rates is equivalent to

$$\forall q, \tilde{\mathbf{p}}_q^*(k, \mathbf{Q}(k)) \in \arg \max_{\{p_{qM}, M \in \mathcal{M}_q\}} \sum_{M \in \mathcal{M}_q} p_{qM} \sum_{i,s} M_{is} \hat{c}_{qis} r_{qi} \left( \frac{1}{\epsilon} - Q_i(k) \right), \tag{47}$$

which means that evaluating the objective function in (47) with  $\mathbf{p} = \mathbf{p}^*$  cannot achieve a larger value. Letting

$$B'_2 \triangleq \min_i \left\{ b_i - \frac{1-\Delta}{1+\Delta} \cdot N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} p_{qM}^* \sum_s M_{is} c_{qis} r_{qi} \right\},$$

from inequality (43), we finally obtain

$$E[V(\mathbf{Q}(k+1)) | \mathbf{Q}(k) = \mathbf{Q}] - V(\mathbf{Q}) \leq -\frac{N}{\epsilon} \left( \frac{1-\Delta}{1+\Delta} \bar{R}(\mathbf{p}^*) - \bar{R}(\tilde{\mathbf{p}}^*(k, \mathbf{Q})) \right) + B_1 - B'_2 \sum_i Q_i.$$

Therefore, similarly as in the proof of Theorem 1, we can finally show that

$$\lim_{K \rightarrow \infty} E \left[ \frac{1}{KN} \sum_{k=0}^{K-1} R(k) \right] \geq \left( \frac{1-\Delta}{1+\Delta} \right) \cdot \bar{R}(\mathbf{p}^*) - \frac{B_1 \epsilon}{N}.$$

#### D. Proof of Lemma 2

By a similar approach as in the proof of Lemma 1 (Appendix A), we have

$$\begin{aligned}
&E[V(\mathbf{Q}(k+1)) | \mathbf{Q}(k) = \mathbf{Q}] - V(\mathbf{Q}) \\
&\leq -N \sum_q \nu_q \sum_{i,s} \left( Q_i + \frac{c_{qis}}{\epsilon} \right) [\tilde{M}^*(q, t, \mathbf{Q})]_{is} + \frac{N}{\epsilon} \bar{J}(\tilde{\mathbf{p}}^*(k, \mathbf{Q})) + D_1 + \sum_i Q_i m_i \\
&\stackrel{(a)}{\leq} -N \sum_q \nu_q \sum_{i,s} \left( Q_i + \frac{c_{qis}}{\epsilon} \right) \sum_{M \in \mathcal{M}_q} \hat{p}_{qM} M_{is} + \frac{N}{\epsilon} \bar{J}(\tilde{\mathbf{p}}^*(k, \mathbf{Q})) + D_1 + \sum_i Q_i m_i \\
&= -\frac{N}{\epsilon} (\bar{J}(\hat{\mathbf{p}}) - \bar{J}(\tilde{\mathbf{p}}^*(k, \mathbf{Q}))) + D_1 - \sum_i Q_i \left( N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} \hat{p}_{qM} \sum_s M_{is} - m_i \right), \tag{48}
\end{aligned}$$

where  $D_1$  is an upper bound on  $\frac{1}{2} \sum_i (E[S_i^2(k, \mathbf{Q}, \mathbf{u}(k))] + E[\tilde{m}_i^2(k)])$  and defined as

$$D_1 \triangleq \frac{1}{2} \left( N(N-1)L^2 + NL + \sum_i [m_i]^2 (m_i - \lfloor m_i \rfloor) + \lfloor m_i \rfloor^2 (1 - m_i + \lfloor m_i \rfloor) \right).$$

Note that inequality (48) has the same form as inequality (43) in the proof of Lemma 1, except that the offline optimum  $\mathbf{p}^*$  is replaced by some  $\hat{\mathbf{p}} \in \mathcal{F}$ . Letting

$$D_2 \triangleq \min_i \left\{ N \sum_q \nu_q \sum_{M \in \mathcal{M}_q} \hat{p}_{qM} \sum_s M_{is} - m_i \right\},$$

it is always possible to pick a  $\hat{\mathbf{p}} \in \mathcal{F}$  such that  $D_2 > 0$  (unless  $\mathcal{F}$  is a degenerated set which has at most one element). We further bound the above inequality as

$$E[V(\mathbf{Q}(k+1)) | \mathbf{Q}(k) = \mathbf{Q}] - V(\mathbf{Q}) \leq \frac{D_3}{\epsilon} + D_1 - D_2 \sum_i Q_i.$$

Here,  $D_3 \triangleq N \cdot \max_{\mathbf{p} \in \mathcal{F}_0} \bar{J}(\mathbf{p})$  where  $\mathcal{F}_0$  is defined in (31). This concludes our proof.

### E. Short-Term Clients and Non-Stationary Query Arrivals

We focus on the click-through rate maximization problem, although a similar model and solution can be used for revenue maximization problem.

First, consider how to include short-term clients in the system. Let us index long-term clients from 1 to  $n$ , the  $i^{\text{th}}$  of which has an average impression requirement of  $m_i$  per requirement cycle. There are further  $\tilde{n}$  types of short-term clients indexed by  $n+1 \leq i \leq n+\tilde{n}$ . Each short-term client of type  $i$  has a impression requirement of  $l_i$  per contract term. Without loss of generality, we assume that the contract term of any short-term client is equal to one requirement cycle. In each requirement cycle  $k$ , there are  $X_i(k)$  clients of type  $i$  in the system, where  $X_i(k)$  follows a stationary stochastic process with mean  $x_i$  and  $X_i(k)$  is known at the beginning of requirement cycle  $k$ .

Correspondingly in an ad assignment matrix  $M$ , the first  $n$  rows and the subsequent  $\tilde{n}$  rows represent the  $n$  long-term clients and the  $\tilde{n}$  types of short-term clients, respectively. If short-term type  $j$  is assigned to some webpage slot, one out of  $X_j(k)$  clients of this type is chosen uniformly at random due to their homogeneity.

Additionally, for a short-term client of type  $i$ , the algorithm is actually aimed to satisfy at least only  $(1 - \alpha_i)l_i$ , where  $\alpha_i \in [0, 1]$  is called “unfulfilled rate” for clients of type  $i$  and to be determined by the algorithm. A strictly convex and monotonically increasing function  $\phi(\alpha_i) \in [0, \infty)$  is then introduced to measure the “unhappiness” of short-term clients about unfulfilled impression requirements, and deducted from the original objective function “overall average click-through rate” in (27) after scaled by some predetermined weight  $w_i$  which reflects the importance of the new metric “unfulfilled rate.”

The second extension from the original model is to consider a more general query arrival pattern. We introduce a new time scale “stationary-arrival period” between the fast one “time slot”  $t$  and the slow one “requirement cycle”  $k$ , namely one requirement cycle equals  $H$  stationary-arrival periods (assuming that  $N/H \in \mathcal{Z}^+$  and usually  $N/H \gg 1$ ), and we assume that query arrivals with respect to each keyword  $q$  form a stationary stochastic process with rate  $\nu_q(h)$  within the  $h^{\text{th}}$  stationary-arrival period in one requirement cycle for all  $1 \leq h \leq H$ . This is a more reasonable assumption for the query arrival pattern in the real Internet. For example, in one day, the query arrivals are stationary within each individual hour, non-stationary across different

hours, and stationary in the same hour across different days. This corresponds to  $H = 24$ , although setting a contract term (already assumed to be equal to one requirement cycle) as one day would only be a simplification for ease of exposition. Based on this example, in the following text we are going to use “day” and “hour” instead of “requirement cycle” and “stationary-arrival period” to better describe the basic ideas.

In summary, the new optimization problem is formulated as

$$\max_{\{\mathbf{p}(h), \forall h; \alpha\}} \frac{1}{H} \sum_q \sum_{h=1}^H \nu_q(h) \sum_{M \in \mathcal{M}_q} p_{qM}(h) \sum_{1 \leq i \leq n+\tilde{n}, s} M_{is} c_{qis} - \sum_{i=n+1}^{n+\tilde{n}} w_i \phi(\alpha_i)$$

subject to

$$\frac{N}{H} \sum_q \sum_{h=1}^H \nu_q(h) \sum_{M \in \mathcal{M}_q} p_{qM}(h) \sum_s M_{is} \geq \begin{cases} m_i, & \forall 1 \leq i \leq n \\ (1 - \alpha_i) l_i x_i, & \forall n+1 \leq i \leq n+\tilde{n} \end{cases}$$

and

$$0 \leq p_{qM}(h) \leq 1, \forall q, M \in \mathcal{M}_q, 1 \leq h \leq H; \quad \sum_{M \in \mathcal{M}_q} p_{qM}(h) \leq 1, \forall q, 1 \leq h \leq H.$$

The only modification in the online algorithm described in Subsection is to add the following two steps specially for each type of short-term clients:

- At the beginning of the  $k^{\text{th}}$  day, update

$$\alpha_i^*(k) = \psi \left( \frac{l_i X_i(k) \cdot Q_i(k)}{H w_i} \right),$$

which corresponds to the target “unfulfilled rate” for each type of short-term clients in this day. Here, the function  $\psi \triangleq \left[ \frac{d\phi}{d\alpha} \right]^{-1}$ .

- At the end of the  $k^{\text{th}}$  day, “credit queue”  $i$  maintained for type  $i$  of short-term clients is updated as

$$Q_i(k+1) = Q_i(k) + (1 - \alpha_i^*(k)) \cdot l_i X_i(k) - S_i(k, \mathbf{Q}(k), \mathbf{q}(k)),$$

where  $S_i(k, \mathbf{Q}(k), \mathbf{q}(k))$  is defined in (33).

The conclusions and proofs about near-optimality of the objective value, queueing stability and upper bound on the expected queue length are similar as those shown for the original problem in Subsection V-A and hence omitted here.

Note that the online algorithm is still “oblivious” to the query arrivals, when the arrival processes become non-stationary to some extent. This is an artifact of dual decomposition w.r.t. each hour  $h$ , in addition to a decomposition w.r.t. each keyword  $q$  as we have seen before.