Online Matching with Concave Returns

[Extended Abstract]

Nikhil R. Devanur
Microsoft Research
1 Microsoft Way
Redmond, WA 98052.
nikdev@microsoft.com

Kamal Jain[*]
Ebay Research
2065 Hamilton Avenue
San Jose, California 95125
kamaljain@gmail.com

ABSTRACT

We consider a *significant* generalization of the Adwords problem by *allowing arbitrary concave returns, and we characterize the optimal competitive ratio achievable*. The problem considers a sequence of items arriving online that have to be allocated to agents, with different agents bidding different amounts. The objective function is the sum, over each agent i, of a monotonically non-decreasing concave function $M_i : \mathbf{R}_+ \to \mathbf{R}_+$ of the total amount allocated to i. All variants of online matching problems (including the Adwords problem) studied in the literature consider the special case of budgeted linear functions, that is, functions of the form $M_i(u_i) = \min\{u_i, B_i\}$ for some constant B_i. The distinguishing feature of this paper is in allowing arbitrary concave returns. The main result of this paper is that for each concave function M, there exists a constant $F(M) \le 1$ such that

- there exists an algorithm with competitive ratio of $\min_i\{F(M_i)\}$, independent of the sequence of items.
- No algorithm has a competitive ratio larger than $F(M)$ over all instances with $M_i = M$ for all i.

Our algorithm is based on the primal-dual paradigm and makes use of convex programming duality. The upper bounds are obtained by formulating the task of finding the right counterexample as an optimization problem. This path takes us through the calculus of variations which deals with optimizing over continuous functions. The algorithm and the upper bound are related to each other via a set of differential equations, which points to a certain kind of duality between them.

Categories and Subject Descriptors

F.1.2 [**Modes of Computation**]: Online computation

General Terms

Algorithms, Theory

[*]Part of the work done while the author was at Microsoft Research.

Keywords

Adwords, Online, Bipartite Matching, Concave returns

1. INTRODUCTION

In recent years, variants of the online bipartite matching problem (most notably the *Adwords* problem) have been the subject of intense investigation. These problems are motivated by their application to online advertising. They are also considered fundamental problems in the theory of online algorithms. In this paper we consider a *significant* generalization of the Adwords problem by *allowing arbitrary concave returns, and we characterize the optimal competitive ratio achievable*.

Consider the following problem, the *online matching problem with concave returns*. An instance of the problem is given by

- a set of agents, L. For each agent i, a monotonically non-decreasing concave function $M_i : \mathbf{R}_+ \to \mathbf{R}_+$.
- A *sequence* of items. For each item j the bid of each agent i, given by b_{ij}. R denotes the set of all items.

An algorithm for the problem allocates items to agents. Fractional allocations are allowed. Let x_{ij} be the fraction of item j allocated to agent i, with $\sum_i x_{ij} \le 1$. The allocation of item j can only depend on the items earlier in the sequence,[1] making this an "online" problem. The goal of the algorithm is to maximize its revenue, denoted by ALG.

$$\text{ALG} = \sum_i M_i\left(\sum_j b_{ij}x_{ij}\right).$$

The performance of an algorithm is measured by *competitive analysis*. ALG is compared with the offline optimum revenue, OPT, which is the optimal revenue taken over all valid allocations, given the entire set of items. An algorithm is F*-competitive*, or equivalently, has a *competitive ratio* of F if for all instances, $\text{ALG}/\text{OPT} \ge F$. The competitive ratio may also be defined on subsets of instances, such as by restricting the M_i's to belong to a particular family of concave functions.

This problem generalizes the Adwords problem. In the Adwords problem the returns are given by *budgeted linear functions*, that is, functions of the form $M_i(u_i) = \min\{u_i, B_i\}$ for some constant B_i, the *budget* of i. All variants of online matching problems studied in the literature (as briefly summarized in Section 4) consider budgeted linear functions, and the distinguishing feature of this paper is in allowing arbitrary concave returns. The motivation for considering concave returns is discussed in Section 1.1.

[1]and, of course, the functions M_i

For the Adwords problem it is known that the optimal competitive ratio is $1 - 1/e$ [16].[2] That is, there is an algorithm that has a competitive ratio of $1 - 1/e$ and no algorithm has a better competitive ratio. We generalize this result to our problem. The main result of this paper is that for each concave function M, there exists a constant $F(M) \leq 1$ such that

- there exists an algorithm with competitive ratio of $\min_i\{F(M_i)\}$, independent of the sequence of items.
- No algorithm has a competitive ratio larger than $F(M)$ over all instances with $M_i = M$ for all i.

All budgeted linear functions have $F = 1 - 1/e$, so our result does generalize the Adwords result. In fact, $F(M) \geq 1 - 1/e$ for all concave functions, and could be much larger. For instance for $M(x) = x^\delta$ for any $\delta \in [0, 1]$, $F(M) = \delta^\delta$. For $\delta = 0.5, \delta^\delta \simeq 0.7 > 1 - 1/e \simeq 0.63$. In general, $F(M)$ does not have a closed form (in fact, M itself is arbitrary, so that is not surprising).

In order to understand the difficulty in proving something like the above, let us look at how it is proven for the Adwords problem. An algorithm is designed and an analysis shows that the competitive ratio is $1 - 1/e$. A family of instances is designed and analyzed to argue that no algorithm has a competitive ratio better than $1 - 1/e$. However there is no insight into why we got the same number in both cases, except that it seems like a lucky coincidence. How do we get lucky when this number is not a nice number such as $1 - 1/e$ or even something we can explicitly calculate? Our analysis peels off a layer of mystery shrouding this coincidence: *we show that these two numbers are the same because they correspond to solutions of the same differential equations.* The coincidence is removed from being "the same number" to being "the same differential equations". Why this is so is still a bit of a mystery and we leave this as an open problem.

We overcome several technical hurdles in proving the results. Our algorithm is based on the primal-dual paradigm and makes use of convex programming duality. The standard form of convex programming duality that uses conjugate functions is somewhat complicated and unwieldy. We give a significantly simpler form that is easy to understand and is a better fit for our algorithm. Another important feature of our algorithm is that once you adopt certain basic principles of primal-dual algorithm design, the rest of the algorithm follows naturally with no *non-deterministic* choices. This "organic" approach leads to certain differential equations that govern the competitive ratio of the algorithm.

The bigger difficulty lies in showing matching upper bounds on the competitive ratio. It is not natural, in fact it is not at all clear how the upper bound would be related to the differential equations from the algorithm. Once again, looking to the Adwords problem for inspiration, the upper bound for the Adwords problem comes from what is known as the *upper triangular graph.* The intuition about why this graph is hard for the Adwords problem carries over for any concave function, so it is natural to use the same graph for the more general case. Unfortunately there are concave functions for which this graph does not give a matching upper bound.[3] One breakthrough was the realization that we can create a family of instances by parametrizing the upper triangular graph by a certain "supply function", and this can give better upper bounds on the competitive ratio. One could try to *guess* the right supply function to get the desired bounds, but this too is quite messy, even for simple concave functions. It turns out that the correct way is to formulate the task of finding the right supply function as an optimization problem. What happens then is that the optimality conditions can be shown to be equivalent to the differential equations from the algorithm, finally giving the sought after connection. This path takes us through the calculus of variations which deals with optimizing over continuous functions (rather than optimizing over a finite set of variables). This connection points to a certain kind of duality between the algorithm and these instances used in the upper bound; this duality warrants further exploration.

[2]with the assumption that bid $\ll$ budget

[3]At this point it was very tempting to conjecture that there is a better algorithm.

1.1 Motivations for concave returns

Concave returns are actually very natural in the setting of online matching. They capture the commonly occurring "diminishing returns" property. We present here some specific examples where concave returns come up.

Display ads: A popular form of selling display ads is via "guaranteed delivery", where the publisher promises to deliver a certain number of impressions to an advertiser. On not delivering on such a promise, the publisher pays an *under-delivery penalty* which is a convex function of the under-delivered amount. When translated into returns as a function of the number of allocated impressions, this becomes a concave function.

Pay-per-click advertisements: In pay-per-click advertisements, the publisher allocates *impressions* to advertisers, but gets paid only for *clicks*. It has been empirically observed that the number of clicks is a concave function of the number of impressions.

Soft budgets: Budget constraints are treated as hard constraints in the Adwords problem, but often in practice there is some leeway. If getting more items is still profitable, an agent could borrow money (and pay an interest rate on it) to pay beyond his budget. This translates to a concave profit function.

Proportional fairness One may want to optimize objectives other than revenue, especially some measure that incorporates fairness of allocation. One such measure, popular in the networking community, is *Proportional fairness*, introduced by Kelly [13]. Proportional fairness maximizes a weighted sum of logs, a concave function.

Organization: The rest of the paper is organized as follows: Section 2 has the description and analysis of the algorithm. Section 3 gives the matching upper bounds. In Section 4 we discuss flexibility of our algorithmic approach, related work, and open problems and directions for future research.

2. THE ALGORITHM

For the rest of the paper we make the assumption that the M_i's have continuous second derivatives. Our approach can actually handle non-smooth functions, but this involves sub-gradients, etc. We make this assumption to keep the exposition simple.

The algorithm is based on an extension of the primal-dual paradigm to convex programs. OPT is the optimum value of a convex program, called the *Primal* program (the one on the left below). The first step is to define a dual program to the Primal convex program. We present a simple and direct construction of the dual program and a proof of (weak) duality. These are based on the simple geometric fact that a concave function is upper bounded by any of its tangents. Let $Y_i(v_i) := M_i(v_i) - v_i M_i'(v_i)$. This is the y-intercept of the tangent to $M_i()$ at v_i, that is, if you draw a tangent to $M_i()$

at v_i then $(0, Y_i(v_i))$ is the point of intersection of the tangent with the y-axis. The *Dual* program, with v_i's and β_j's as variables is stated below on the right.

$$\begin{array}{ll}
\text{maximize} \sum_i M_i(u_i) \text{ s.t.} & \text{minimize} \sum_i Y_i(v_i) + \sum_j \beta_j \text{ s.t.} \\
\forall\, i, u_i = \sum_j b_{ij} x_{ij}. & \forall\, i, j, \beta_j \geq b_{ij} M_i'(v_i). \\
\forall\, j, \sum_i x_{ij} \leq 1. & \forall\, i, j, v_i, \beta_j \geq 0. \\
\forall\, i, j, x_{ij} \geq 0. &
\end{array}$$

LEMMA 1. *(Weak duality) The optimum value of the Dual program $\geq$ OPT.*

PROOF. Since M_i is concave, a tangent to M_i at any point v_i (in the domain of M_i) is an upper bound on M_i.

$$\forall\, i, \forall\, v_i \geq 0, M_i(u_i) \leq M_i(v_i) + (u_i - v_i) M_i'(v_i)$$

$$= M_i'(v_i) u_i + Y_i(v_i).$$

Suppose we fix v_i for all i, and replace M_i by the upper bound in the inequality above, thus linearizing the objective function of the Primal program. Let $D(\boldsymbol{v})$ be the value of the following linear program (on the left). The dual of this LP is on the right, the optimum value of which is also $D(\boldsymbol{v})$ by strong LP duality. ($\boldsymbol{v}$ is the vector of v_i's.)

$$\begin{array}{ll}
\text{maximize} \sum_i M_i'(v_i) u_i \text{ s.t.} & \text{minimize} \sum_j \beta_j \text{ s.t.} \\
\forall\, i, u_i = \sum_j b_{ij} x_{ij}. & \forall\, i, j, \beta_j \geq b_{ij} M_i'(v_i). \\
\forall\, j, \sum_i x_{ij} \leq 1. & \forall\, j, \beta_j \geq 0. \\
\forall\, i, j, x_{ij} \geq 0. &
\end{array}$$

For every choice of $\boldsymbol{v}$ we get an upper bound on OPT.

$$\forall\, \boldsymbol{v}, \text{OPT} \leq D(\boldsymbol{v}) + \sum_i Y_i(v_i).$$

In the above LPs (in particular, the dual LP), v_i's are considered as constants. We can instead think of them as variables and obtain a single mathematical program with v_i's and β_j's as variables. This is exactly what the Dual program is, $\min_{\boldsymbol{v}} \{D(\boldsymbol{v}) + \sum_i Y_i(v_i)\}$. The lemma follows. □

The primal-dual paradigm as applied to online algorithms is to maintain a set of dual variables that guide the primal solution. The evolution of the primal solution in turn determines how the dual variables are updated. The part of dual variables guiding the primal solution usually follows very naturally from the complementary slackness conditions (or with the duals being interpreted as costs, etc). The non-trivial part is how the dual variables are updated. This requires that

- the dual variables remain feasible and that
- the ratio of the values of the primal and the dual solutions remains within desirable bounds.

The algorithm design principle we espouse in this paper is that *once we determine that the dual variables satisfy the above conditions, their dependence on the primal solution can be "reverse-engineered"*. The above conditions often take the form of a differential equation which can be solved (either in a closed form or numerically) to give the dual update method.

We now describe the algorithm. Recall that the algorithm has to determine the allocation x_{ij} in an online manner. The algorithm also constructs a solution to the Dual program. The algorithm maintains for each i, the variable $u_i = \sum_j b_{ij} x_{ij}$. The dual variable v_i is set as a (smooth) function of u_i, such that v_i is monotonically non-decreasing. The exact dependence of v_i on u_i is left unspecified for now. The dual variables guide the primal allocation via complementary slackness conditions. For instance, one complementary slackness condition says that $x_{ij} > 0$ implies $\beta_j = b_{ij} M_i'(v_i)$. This means we must allocate j to $\arg\max_i \{b_{ij} M_i'(v_i)\}$. However the $\arg\max$ may change as a *result* of the allocation, since allocating j to i increases u_i, which increases v_i, which decreases $M_i'(v_i)$. If the $\arg\max$ is unique and does not change even after allocating j completely to i, then j is allocated completely to i (that is $x_{ij} = 1$). If either the $\arg\max$ is not unique or it changes when we allocate j to i, we need to allocate fractionally. The allocation is best described as a continuous process. As you start allocating j to i, u_i increases, which changes v_i which in turn could change the $\arg\max$. To describe the process, let t denote time and we specify x_{ij} as a function of t. In turn, u_i and v_i are also functions of t. In fact the allocations change smoothly over time and we only specify $\frac{dx_{ij}}{dt}$. We set $\frac{dx_{ij}}{dt} = 0$ if $i \notin \arg\max_i \{b_{ij} M_i'(v_i)\}$. Otherwise $\frac{dx_{ij}}{dt} > 0$ is such that $b_{ij} \frac{dM_i'(v_i(t))}{dt}$ is the same for all the agents in the $\arg\max$. Note that this only specifies $\frac{dx_{ij}}{dt}$ upto a constant factor. This does not matter since the "speed of time" is immaterial to the algorithm. Allocation stops when $\sum_j x_{ij} = 1$. The dual variable β_j is set as follows.

$$\beta_j = \int \sum_i \{b_{ij} M_i'(v_i)\} \frac{dx_{ij}}{dt} dt. \tag{1}$$

β_j is only needed for the analysis of the algorithm and does not affect the run of the algorithm itself.

The description of the algorithm is completed by specifying the dependence of v_i on u_i. As mentioned before these are reverse-engineered from the properties we want the dual solution to satisfy (Lemmas 2 and 3 here). These take on the form of a solution to a first order differential equation, which we present below. Let $F > 0$ be some constant. A solution to the following differential equation gives v as a function of u.

$$M'(u)/F = Y'(v) \frac{dv}{du} + M'(v). \tag{2}$$

$$\frac{dv}{du} \geq 0.$$

Boundary conditions: when $u = 0, Y(v) = 0$. $u, v \geq 0$.

Suppose that for some constant F, for all agents i, (2) has a solution with $M = M_i$. Let the dependence of v_i on u_i in the algorithm be given by such a solution. The algorithm is summarized below.

The analysis of the algorithm is by bounding the ratio of the primal and dual solutions constructed. Let P and D be respectively the values of the primal and the dual solutions created by the algorithm throughout its run.

Algorithm 1: Algorithm for the online matching problem with concave returns.

```
Input: functions $M_i$ for all $i$.
Let $F$ be a constant such that for all agents $i$, (2) has a solution with $M = M_i$ ;
Throughout the algorithm, maintain the following invariants
for each $i$ do
    $u_i = \sum_j b_{ij} x_{ij}$;
    $v_i$ is a function of $u_i$ that satisfies (2);
for each $j$ when $j$ arrives do
    allocate $j$ to $\arg\max_i \{b_{ij} M_i'(v_i)\}$ in a continuous fashion ;
    set $\beta_j$ as in (1);
```

LEMMA 2. *The following is an invariant throughout the algorithm.*

$$\frac{dP}{dD} = F.$$

PROOF. The differential equation (2) has already been defined to make this lemma go through. Suppose dx of j is allocated to i. Then

$$dP = M_i'(u_i) b_{ij} dx = M_i'(u_i) du_i.$$

$$dD = Y_i'(v_i) dv_i + b_{ij} M_i'(v_i) dx = Y_i'(v_i) dv_i + M_i'(v_i) du_i.$$

Lemma now follows from (2). □

LEMMA 3. *The variables v_i and β_j constructed by the algorithm form a feasible solution to the Dual program.*

PROOF. This is easy to see since we allocate j only to $\arg\max_i \{b_{ij} M_i'(v_i)\}$, and the fact that $M_i'(v_i)$ is monotonically non-increasing throughout the algorithm. □

THEOREM 4. *The algorithm has a competitive ratio of F.*

PROOF. The theorem follows immediately from Lemmas 1, 2 and 3, and the fact that when $P = 0$, $D = 0$. □

Note: It is easy to see for instance, that the greedy algorithm has a competitive ratio of $\frac{1}{2}$. The greedy algorithm corresponds to the solution $v = u$. In this case, consider the integral form of (2) with $F = \frac{1}{2}$ and note that the left hand side is $2M(u)$ and the right hand side is $Y(u) + M(u)$. Since $M(u) \geq Y(u)$, the conclusion follows. This also shows that $F(M) \geq \frac{1}{2}$ for all M. In fact it can be shown that $F(M) \geq 1 - 1/e$ for all M, via an indirect argument which we will not go into here.

Note that this completes the first part of the main result. For a given concave function M define $F(M)$ to be the largest constant such that (2) has a solution. Then the algorithm has a competitive ratio of $\min_i\{F(M_i)\}$.

3. THE UPPER BOUND

In this secion we prove the second part of our main result, by showing a matching upper bound on the competitive ratio of any algorithm. Such an upper bound is proven by constructing instances and arguing that no algorithm can perform well on all the instances. We begin by some simplifying observations.

In order to prove an upper bound on the competitive ratio, we need to consider randomized algorithms as well. In case of randomized algorithms, we consider the expected revenue of the algorithm. However for this problem, randomization does not help. To see this, for any randomized algorithm, consider the deterministic algorithm whose allocation is equal to the expected allocation of the given randomized algorithm. The revenue of the deterministic algorithm is no lower than the expected revenue of the randomized algorithm due to the concavity of the M_i's. Thus we can restrict ourselves to deterministic algorithms only.

We show that for a certain family of instances we can characterize the optimal algorithm, the one with the largest competitive ratio. This enables us to prove the upper bound by simply bounding the competitive ratio of the optimal algorithm. Consider a bipartite graph $\mathcal{G} = (L, R, E)$, with vertex sets $L = R = \{1, 2, \ldots, n\}$ and edges $(i, j) \in E$ iff $i \geq j$. This graph is commonly called the "upper triangular" graph, since its adjacency matrix is upper triangular. Fix a concave function M. Consider instances of the problem with the set of agents being isomorphic to L, modulo the identity of agents. That is, we have n agents and consider all possible bijections of agents to L. $M_i = M$ for all i. The items are the vertices in R, arriving in the increasing order. $b_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise. The first item is adjacent to all the agents. Every subsequent item is adjacent to all the agents the previous item was adjacent to, except one. The algorithm has to make its allocation without knowing which agent will be "left out" for the next item.

We claim that the algorithm that divides the items equally among all its neighbors is the optimal algorithm for these instances. That is, this algorithm has the largest competitive ratio taken over all these instances. If an algorithm has an inequitable allocation then an adversary picking the instance can take advantage of this by picking the agent with the lowest allocation to be left out for the next item. This results in a lower objective function than the equitable allocation because of concavity of returns. Thus, it is sufficient to show an upper bound on the competitive ratio of the equitable algorithm. That is, if we show that for this algorithm ALG/OPT $\leq F$ then no algorithm can have a competitive ratio better than F on these instances, and in turn on all instances with $M_i = M$ for all i.

In fact, we extend the above to instances where the items could have an arbitrary "supply". Suppose item j has supply f_j, then the supply constraint is that $\sum_i x_{ij} \leq f_j$. It is easy to see that the assertion about the equitable algorithm being optimal still holds.

With these simplifying observations, the aim is to construct for any M, a supply function f_j such that ALG/OPT for that instance is arbitrarily close to $F(M)$, the largest constant such that (2) has a solution. A direct way to show that would be to consider any constant larger than $F(M)$ and construct an f so that ALG/OPT is equal to that constant. Alas we don't know of any such direct construction; the difficulty is in relating this to (2). It turns out that the relation of (2) with this family of instances is somewhat indirect. *We show that if you formulate the task of finding the f that minimizes ALG/OPT as an optimization problem, then the conditions that characterize the optimal choice are equivalent to (2)!!* From this it follows that for the optimal choice of f we have that ALG/OPT $= F(M)$, giving the result we were after.

In fact, to follow the plan as outlined above, we consider a continuous version of the upper triangular graph. We define a bipartite graph $\mathcal{G} = (L, R, E)$, with the vertex sets $L = R = [0, 1]$. $(x, y) \in E$ iff $x \geq y$. Many of the concepts in the discrete version are based on summations, but they break down since we cannot take summations in the continuous version. We now specify how to extend all the concepts to their continuous counterparts. The "supply" of items is given by a function $f : R \to \mathbf{R}_+$. An allocation is given by a function $A(x, y)$. The supply constraint translates to

$$\forall y \in R, \int_y^1 A(x,y)dx = f(y).$$

The amount allocated to x, say $u(x)$ is given by

$$u(x) = \int_0^x A(x,y)dy.$$

The revenue of the algorithm is

$$\text{ALG} = \int_0^1 M(u(x))dx.$$

The optimal algorithm has the allocation

$$A(x,y) = \frac{f(y)}{1-y} \ \forall x \geq y.$$

$$\text{Thus } u(x) = \int_0^x \frac{f(y)}{1-y}dy.$$

The offline optimal allocation is

$$A(x,y) = f(y)\delta_y(x)$$

where $\delta_y(x)$ is the dirac-delta function with peak at y. With this allocation the amount allocated to x is $f(x)$. Thus

$$\text{OPT} = \int_0^1 M(f(x))dx.$$

3.1 Calculus of variations

As mentioned before, the key to the upper bound proof is to formulate the task of finding an f that minimizes the ratio ALG/OPT as an optimization problem. A quantity that is a function of a function, such as ALG/OPT, is called a *functional* (and is denoted ALG/OPT$[f]$). The subject of finding extremal functions, that is functions that optimize a functional is called the *calculus of variations*. The calculus of variations is used for instance to find geodesics on surfaces, or in physics to apply the "principle of least action". It will be used here crucially.

We will consider functions from $\mathcal{C}$, the class of all real valued functions defined on the domain $[0, 1]$, that are *right-continuous*. The dot product of two functions is defined as

$$f \cdot g := \int_0^1 f(x)g(x)dx.$$

The gradient of a functional H (at some f_0) is another function ∇H such that for any other function η, the rate of change of H in the direction of η starting from f is $\nabla H \cdot \eta$. That is,

$$\frac{dH[f_0 + \epsilon\eta]}{d\epsilon}|_{\epsilon=0} = \nabla H \cdot \eta.$$

It is easy to compute the gradient of OPT:

$$\nabla\text{OPT}(x) = M'(f(x)).$$

It is a little harder to compute the gradient of ALG.

LEMMA 5.

$$\nabla ALG(x) = \frac{1}{1-x}\int_x^1 M'(u(t))dt.$$

PROOF. Recall that

$$\text{ALG}[f] = \int_0^1 M(u(x))dx, \text{ where}$$

$$u(x) = \int_0^x \frac{f(y)}{1-y}dy.$$

Let $f = f_0 + \epsilon\eta$.

$$\frac{d\text{ALG}[f]}{d\epsilon} = \int_0^1 \frac{dM(u(x))}{d\epsilon}dx = \int_0^1 M'(u(x))\frac{du(x)}{d\epsilon}dx.$$

$$\frac{du(x)}{d\epsilon} = \int_0^x \frac{d}{d\epsilon}\frac{f(y)}{1-y}dy = \int_0^x \frac{\eta(y)}{1-y}dy.$$

Putting the two together and changing the order of integration gives

$$\frac{d\text{ALG}[f]}{d\epsilon} = \int_0^1 M'(u(x))\int_0^x \frac{\eta(y)}{1-y}dydx$$

$$= \int_0^1 \frac{\eta(y)}{1-y}\int_y^1 M'(u(x))dxdy$$

$$= \int_0^1 \frac{\eta(x)}{1-x}\int_x^1 M'(u(t))dtdx.$$

The last equality is a result of change of integration variables. □

For a given $L > 0$, let $\mathcal{C}_L$ be the class of all real-valued functions f defined on the domain $[0, 1]$ that are right-continuous, with $f(0) = 0$ and bounded above by L.[4] From now on, let H denote the functional ALG/OPT. For every L, and $f \in \mathcal{C}_L$, we get an upper bound of $H[f]$ on the competitive ratio of any algorithm. We are interested in the least upper bound we can obtain this way. We define the following quantities for ease of notation.

$$\inf_{f\in\mathcal{C}_L} H[f] =: \lambda_L^*, \text{and}$$

$$\inf_L \lambda_L^* =: \lambda^*.$$

We first establish the existence of a minimizer of H in the class $\mathcal{C}_L$.

THEOREM 6. *$\exists f^* \in \mathcal{C}_L$ such that $H[f^*] = \lambda_L^*$.*

The proof of the above theorem is broken down into Lemmas 7–9. To begin with, the definition of λ_L^* implies there exists a sequence of functions $f_1, f_2, \ldots, f_n, \ldots \in \mathcal{C}_L$ such that

$$\lim_{n\to\infty} H[f_n] = \lambda_L^*.$$

LEMMA 7. *We may assume without loss of generality that the functions f_n are monotonically non-decreasing.*

PROOF. Suppose f_n is not monotonically non-decreasing. Suppose for simplicity that there is only one interval where f_n is decreasing. Let $\hat{f}_n$ be the "ironed"[5] version of f_n. The function $\hat{f}_n$ is monotonically non-decreasing, f_n and $\hat{f}_n$ agree except on an interval $[a, b]$ and $\int_a^b f_n dx = \int_a^b \hat{f}_n dx$. Let $\eta = \hat{f}_n - f_n$. We will argue that $\nabla\text{ALG} \cdot \eta \leq 0$ and $\nabla\text{OPT} \cdot \eta \geq 0$, which implies that $\nabla H \cdot \eta \leq 0$. Therefore $H[\hat{f}_n] \leq H[f_n]$ and f_n can be replaced by $\hat{f}_n$ in the sequence.

Further subdivide the interval $[a, b]$ as $[a, c]$ and $[c, b]$ such that $\eta \leq 0$ in $[a, c]$ and $\eta \geq 0$ in $[c, b]$. For $x \in [a, c]$, $f(x) \geq f(c)$, therefore $M'(f(x)) \leq M'(f(c))$. Further, since $\eta(x) \leq 0$, $\eta(x)M'(f(x)) \geq \eta(x)M'(f(c))$. Vice versa, for $x \in [c, b]$,

[4]Strictly speaking we want to let $f(0) = \sup\{v : Y(v) = 0\}$, but we will ignore this for the sake of ease of notation.

[5]This is the same as Myerson's ironing [17] used in Bayesian mechanism design.

$f(x) \le f(c)$, therefore $M'(f(x)) \ge M'(f(c))$. Here $\eta(x) \ge 0$, so once again $\eta(x)M'(f(x)) \ge \eta(x)M'(f(c))$. Therefore

$$\nabla \mathrm{OPT} \cdot \eta = \int_a^b \eta(x) M'(f(x)) dx$$

$$\ge M'(f(c)) \int_a^b \eta(x) = 0.$$

Similarly, it is easy to check that ∇ALG is a non-increasing function of x and hence for $x \in [a, c]$, $\nabla \mathrm{ALG}(x) \ge \nabla \mathrm{ALG}(c)$, and for $x \in [c, b]$, $\nabla \mathrm{ALG}(x) \le \nabla \mathrm{ALG}(c)$. Using an argument that is similar to the one above, we get that $\nabla \mathrm{ALG} \cdot \eta \le 0$. □

LEMMA 8. *(Hally-Bray. See also [19], Section 17.5.) Given a sequence of non-decreasing functions (f_n) in $\mathcal{C}_L$, there exists a convergent subsequence $(f_{n_i} : i \in \mathbb{N}) \to f^*$ with pointwise convergence. $f^* \in \mathcal{C}_L$ is non-decreasing, but may not be continuous.*

LEMMA 9. *H is continuous. That is, if a sequence $(f_n : n \in \mathbb{N}) \to f$ pointwise, then $(H[f_n]) \to H[f]$.*

PROOF. It is easy to see that ALG and OPT are continuous. Also H is always in $[\frac{1}{2}, 1]$. Hence H is continuous. □

Theorem 6 now follows since

$$\lim_{i \to \infty} H[f_{n_i}] = H[f^*] = \lambda_L^*.$$

Now we show several properties of f^* with the goal of relating f^* to (2). Let $l := \inf\{x : f^*(x) = L\}$. By right continuity of f^*, it follows that $f^*(l) = L$.

THEOREM 10. *([18])*

$$\nabla H[f^*](x) = 0 \, \forall \, x \in [0, l].$$

PROOF. The above theorem is the standard theorem in the calculus of variations, giving necessary conditions for extremal functions. The exact form stated here is obtained by considering all variations η such that $\eta(0) = 0 = \eta(x)$ for all $x \in [l, 1]$. □

LEMMA 11.

$$\nabla ALG[f^*] = \lambda_L^* \nabla OPT[f^*] \, \forall \, x \in [0, l].$$

PROOF. The lemma follows essentially from the quotient rule for differentiation along with Theorem 10. □

We next show that in fact f^* is a differentiable function.

LEMMA 12. *f^* is differentiable in $[0, 1)$ (and is therefore continuous).*

PROOF. Note that $\nabla \mathrm{ALG}[f^*]$ is differentiable in $[0, 1)$. From Lemma 11 so is $\nabla \mathrm{OPT}[f^*]$. However, $\nabla \mathrm{OPT}[f^*] = M'(f^*)$, and M' is differentiable. Therefore it follows that f^* is differentiable. □

We have established that for every L, the minimizer f^* of H in $\mathcal{C}_L$ exists, is non-decreasing, and is also differentiable in $[0, 1)$. We are now ready to relate f^* to the primal differential equation. We define a relaxation of (2) by requiring the conditions to hold only for $v \le L$. To be precise consider the system of equations,

$$M'(u)/F = Y'(v)\frac{dv}{du} + M'(v), \tag{3}$$

for all u such that $v(u) \le L$.

$$\frac{dv}{du} \ge 0.$$

Boundary conditions: when $u = 0, Y(v) = 0$. $u, v \ge 0$.

Let F_L^* be the supremum of all F such that (3) has a solution. Clearly $F_L^* \ge F(M)$ since a solution to (2) also satisfies (3). It is also easy to see that

$$\inf_L F_L^* = F(M).$$

We now state a lemma that shows that f^* can be used to construct a solution to (3).

LEMMA 13.

$$\forall \, L, \lambda_L^* \le F_L^*.$$

PROOF. Recall that a solution to (3) is to contruct v as a function of u. Note that $\frac{du}{dx} > 0$ for $u > 0$ and hence one can invert the function $u(x)$ and think of x as a function of u. Define $v(u) = f^*(x(u))$. We now show that this solution satisfies (3) with $F = \lambda_L^*$. The lemma then follows from the definition of F_L^*.

We start with the conclusion of Lemma 11: for all $x \in [0, l]$,

$$\nabla \mathrm{ALG}[f^*] = \lambda_L^* \nabla \mathrm{OPT}[f^*].$$

$$\tfrac{1}{1-x} \int_x^\infty M'(u(t))dt = \lambda_L^* M'(f^*(x)).$$

$$\tfrac{1}{\lambda_L^*} \int_x^\infty M'(u(t))dt = (1-x) M'(f^*(x)).$$

Differentiating both sides w.r.t. x,

$$\tfrac{-1}{\lambda_L^*} M'(u(x)) = -M'(f^*(x)) + (1-x)M''(f^*(x))\frac{df^*}{dx}.$$

Use the fact that $\frac{df^*}{dx} = \frac{df^*}{du}\frac{du}{dx}$ and $\frac{du}{dx} = \frac{f^*}{1-x}$.

$$\tfrac{1}{\lambda_L^*} M'(u(x)) = M'(f^*(x)) - f^* M''(f^*(x))\frac{df^*}{du}.$$

Substitute $v = f^*$ and use $Y'(v) = -vM''(v)$.

$$\tfrac{1}{\lambda_L^*} M'(u) = M'(v) + Y'(v)\frac{dv}{du},$$

for all u such that $v(u) \le L$. □

The main theorem of this section now follows from Lemma 13 easily, by taking infimums.

THEOREM 14. $\lambda^* = F(M)$.

4. DISCUSSIONS AND CONCLUSION

Flexibility of our approach

Another strength of our algorithmic approach is its flexibility: suppose that even for budgeted linear functions, the algorithm designer knows (based on historic data, for instance) that an agent is guaranteed to get a certain number of items. Our algorithm can be tuned to take advantage of this, by appropriately changing the boundary conditions in the differential equations. This leads to a better competitive ratio. In general this approach opens up the possibility of incorporating other kinds of beliefs about the instance and get the optimal algorithm based on these beliefs.

Related work

The Adwords problem was introduced by [16], who gave a $1 - 1/e$ competitive algorithm, based on a factor revealing LP, with the assumption that bid $\ll$ budget. They also showed that this is the best competitive ratio achievable. [4] gave a primal-dual analysis of the same result. Our algorithm is a generalization of the primal-dual approach of [4]. [12] considered the online bipartite matching problem and gave a $1 - 1/e$ competitive randomized algorithm. This is a special case of Adwords with bids = 0 or 1 and budgets = 1, but this does not satisfy the bid $\ll$ budget assumption. [10] gave a $1 - 1/e$ competitive algorithm for the b-matching problem with large b, a special case of Adwords with budgets = b and bids = 0 or 1. The Adwords problem without the bid $\ll$ budget assumption is open, with the best known algorithm being $1/2$-competitive. A $1 - 1/e$ competitive algorithm is also known for another special case of Adwords without the bid $\ll$ budget assumption, which is the vertex-weighted version of online matching [1].

Most of the papers lately consider *stochastic* variants of online matching, with the dominant theme of "beating" the $1-1/e$ bound. These make some distributional assumptions about the sequence of items and analyze the performance of the algorithm in expectation over the distribution. A partial list of papers is as follows: [9, 5, 2, 7, 6, 8, 15, 3, 11, 14].

Open problems and directions

One open problem is to dig deeper into the coincidence of differential equations. This seems to point to a different kind of duality between the algorithm and the counterexample. Understanding this duality would lead to greater insights into what looks like a fundamental issue in the design of online algorithms. Another open problem is to use this approach to get optimal competitive ratios for other online problems. In fact, an ambitious goal is to be able to reverse engineer: can hard instances (believed to be worst case instances) of problems be used to design the algorithms?

5. ACKNOWLEDGMENTS

We would like to thank Yuval Peres, Alexander Holroyd, Mark Braverman, David Levin and Oded Schramm for useful discussions and pointers to references.

6. REFERENCES

[1] G. Aggarwal, G. Goel, C. Karande, and A. Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *SODA*, pages 1253–1264, 2011.

[2] S. Agrawal, Z. Wang, and Y. Ye. A dynamic near-optimal algorithm for online linear programming. arXiv:0911.2974v1, 2009.

[3] B. Bahmani and M. Kapralov. Improved bounds for online stochastic matching. In *ESA*, pages 170–181, 2010.

[4] N. Buchbinder, K. Jain, and J. S. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *ESA'07: Proceedings of the 15th annual European conference on Algorithms*, pages 253–264, Berlin, Heidelberg, 2007. Springer-Verlag.

[5] N. R. Devanur and T. P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In J. Chuang, L. Fortnow, and P. Pu, editors, *ACM Conference on Electronic Commerce*, pages 71–78. ACM, 2009.

[6] N. R. Devanur, K. Jain, B. Sivan, and C. A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In Y. Shoham, Y. Chen, and T. Roughgarden, editors, *ACM Conference on Electronic Commerce*, pages 29–38. ACM, 2011.

[7] J. Feldman, M. Henzinger, N. Korula, V. S. Mirrokni, and C. Stein. Online stochastic packing applied to display ad allocation. In *ESA*, pages 182–194, 2010.

[8] J. Feldman, A. Mehta, V. Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *FOCS '09: Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 117–126, Washington, DC, USA, 2009. IEEE Computer Society.

[9] G. Goel and A. Mehta. Online budgeted matching in random input models with applications to adwords. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 982–991, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[10] B. Kalyanasundaram and K. R. Pruhs. An optimal deterministic algorithm for online b-matching. *Theor. Comput. Sci.*, 233(1-2):319–325, 2000.

[11] C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *STOC*, pages 587–596, 2011.

[12] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 352–358, 1990.

[13] F. P. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.

[14] M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *STOC*, pages 597–606, 2011.

[15] V. H. Manshadi, S. O. Gharan, and A. Saberi. Online stochastic matching: Online actions based on offline statistics. In *SODA*, pages 1285–1294, 2011.

[16] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized on-line matching. In *In FOCS âĂŹ05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273. IEEE Computer Society, 2005.

[17] R. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.

[18] H. Sagan. *Introduction to the Calculus of Variations*. Dover Publications Inc., 1969.

[19] D. WIlliams. *Probability with Martingales*. Cambridge University Press, 1991.