Online Matching in Advertisement Auctions *

Nikhil R. Devanur Aranyak Mehta

1 Introduction

Advertising has become a large source of revenue for many internet services, and at the same time internet technologies have completely transformed advertising. At the time of writing, worldwide digital advertising spend is estimated to be close to half a trillion dollars per year, and accounts for a majority of all media ad spending. It is the ad revenue that allows internet services companies to offer *for free* invaluable services such as search, social media, news, video, email, maps, operating systems, and all kinds of apps.

Among the different channels in internet advertising, search advertising deserves special mention. When a user issues a search query with the intent to look for certain goods or services, the platform matches them to firms selling those goods or services, by displaying the relevant ads. Thus, firms are more certain to have their advertising budget spent on opportunities which are likely to result in a good return. Indeed, payment from an advertiser to a search engine is usually per-click, which is already a strong indication that the user has found the ad useful. Even display and video advertising have moved to a world of real-time bidding, which allows sophisticated algorithms to differentially evaluate every single ad opportunity. Whenever user activity generates an ad-slot, ads compete in a fully automated auction to win that slot.

Given the size of the ad market and its importance to online commerce, deeper understanding and optimization of the underlying mechanisms has an out-sized impact. In this chapter, we will describe an important aspect of this market design, namely that of online matching in the auction context.

Auctions: Being a market of a highly heterogeneous set of goods (e.g., the huge range of possible queries in search advertising), ads are sold via auctions rather than fixed prices. For each new ad-slot, retrieval and machine learning systems find the relevant ad candidates with their bids and quality scores, and an auction determines which ad gets shown, and how much the advertiser pays.

*This chapter will appear in:
Online and Matching-Based Market Design. Federico Echenique, Nicole Immorlica and Vijay V. Vazirani, Editors. Cambridge University Press.

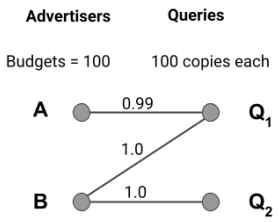

Figure 1: A bad instance for the straightforward auction approach

Online matching: As such, auctions for any two ad-slots are independent of each other. However, demand constraints from the advertisers connect the different auctions. A very commonly used demand constraint is a budget: Besides a per-click bid for each ad-slot (expressed via keywords), an advertiser can also express an overall budget, denoting the maximum they are willing to pay in a time-period (typically a day). Consequently, decisions taken in one auction affect other auctions. The obvious way for the platform to deal with budgets is to run a simple auction in which each ad-slot goes to the highest bidder, and remove advertisers from consideration as they finish their budgets. However, this approach is not ideal: Advertisers would end up finishing their budgets early in the day, giving them a biased sample of ad opportunities. Further, this would lead to uneven competition in the auctions with competition thinning out later in the day. This dynamic can be clearly seen in the following extreme example:

Example 1. There are two advertisers (A and B) and 200 queries (see Fig. 1). The advertisers have the same budget, $B_A = B_B = 100$. The first 100 queries Q_1 are identical (e.g., adslots for the keyword "flowers") for which A and B both bid similarly: $b_{Aj} = 0.99, b_{Bj} = 1,\ \forall j \in Q_1$. The second 100 queries Q_2 are also identical (the keyword "gifts"), with only B interested in them: $b_{Aj} = 0, b_{Bj} = 1,\ \forall j \in Q_2$. The straightforward auction approach matches all $j \in Q_1$ to B, at which point $\text{spend}_B = B_B$ (for simplicity assume first price). Now when queries in Q_2 arrive, B can not be matched any more, while A bids 0 for them. This results in a revenue of 100. An **optimal matching** instead matches all $j \in Q_1$ to A and all $j \in Q_2$ to B, with a revenue of 199.

Thus, running the straightforward auction and simply imposing the budget constraints leads to suboptimal efficiency and revenue; we need to explicitly consider the interactions across auctions and pace the spend over the day. This leads to an *online bipartite matching* formulation: The two sides of the bipartite graph are the demand (advertisers) and the supply (ad-slots), edges codify the bids and quality scores, and the budgets place constraints on each advertiser vertex. Note that the problem is necessarily online, i.e., each ad-slot needs to be matched to an advertiser as soon as it arrives, without knowledge of the future. We clarify here that we use the term matching in the more general sense of an allocation, determining which ads are shown for which queries; e.g., we are not

necessarily constrained to match each advertiser vertex to exactly one neighbor.

Centralized vs Decentralized solutions: There are two different approaches to solving this matching problem: A *centralized approach* treats the problem as an (online) optimization problem, and makes the matching decisions in each auction. We begin this chapter with a focus on the centralized approach, describing the problem, algorithms, and results. A *decentralized approach* considers the problem from a bidding perspective, in which each advertiser changes its bids under its budget constraints to optimize for itself. This raises the question of what happens when all advertisers bid in this way. What kind of matching would that result in? We will discuss this approach in Sec. 7.

2 The AdWords problem

We formulate the centralized matching question as the following online budgeted allocation problem, referred to in the literature as the AdWords problem.

Definition 2. In the **AdWords problem**, there is a bipartite graph of n advertisers and m queries. Advertiser i has a budget of B_i on its total spend. The bid of advertiser i for query j is denoted by b_{ij}, which can be considered the weight of the edge (i, j). Consider a matching M of queries to advertisers where $M(i)$ denotes the set of queries matched to advertiser i. The total reward of M (we will also refer to this as the *revenue*) is defined as

$$\sum_{i \in [n]} \min\{B_i, \sum_{j \in M(i)} b_{ij}\} \tag{1}$$

The algorithm starts off knowing only the set of advertisers and their budgets, but not the set of queries, which arrive sequentially, nor the potential bids. When a query j arrives, the algorithm is revealed the bids b_{ij}, for all i, and has to immediately decide to match j to one advertiser (or leave j unmatched), irrevocably. The goal is to maximize the reward (1) at the end of the sequence.

We can also define the problem *iteratively* as follows. Initially, the *spend* of each advertiser i, spend_i, is 0. When a query j arrives, if the algorithm can match it to some advertiser i with $\text{spend}_i < B_i$. The algorithm accrues a reward of $\min\{b_{ij}, B_i - \text{spend}_i\}$, and spend_i increases by the same amount.

Remark 3. Note that the AdWords problem as defined above is quite general. For example, if we set every budget B_i to 1, and every bid b_{ij} to either 0 or 1, then we recover the online bipartite matching problem from Chapter **??**. The general problem is hard to solve exactly even in an *offline* setting, i.e., when we know the entire graph in advance: it is NP-hard to approximate the optimal solution with a multiplicative approximation factor better than $\frac{15}{16}$.

Competitive Analysis: We evaluate an algorithm by its competitive ratio, which is the minimum ratio (over all instances of the problem) of the reward obtained by the algorithm to the optimal achievable reward in hindsight, i.e., if we knew the entire graph. We will refer to the latter as OPT throughout.

Let us recall the straightforward strategy of picking the highest bidder among those with remaining budget.

Highest-bidder: Match the arriving query j to the advertiser i with the highest value of $\min\{b_{ij}, B_i - \text{spend}_i\}$.

In Example 1, we saw that this strategy achieves no more than a half of the optimal matching in hindsight (to be precise, we get to a half by increasing the value of the budgets and the length of the query sequences). This means the competitive ratio of Highest-bidder is no more than $1/2$; in fact it is precisely $1/2$ as it can be shown that it achieves at least half of OPT in every instance.

In light of this example, the goal is to find an algorithm with a competitive ratio strictly better than a half. It turns out that this is a very difficult problem in general, and the first such algorithm, achieving a slightly improved ratio of 0.5016, was discovered only very recently. Instead, we make the following reasonable practical assumption, which results in a much more tractable problem, allowing for an algorithm with a much improved competitive ratio, and also yielding a heuristic with significant impact in practice.

Definition 4. Let $\rho = \max_{i\in[n],j\in[m]} \frac{b_{ij}}{B_i}$. In the **small-bids assumption**, we assume that ρ is small, i.e., every bid is much smaller than the corresponding budget. The performance of the algorithms is evaluated as $\rho \to 0$.

Remark 5. With this assumption, the AdWords problem no longer captures the online bipartite matching problem. Further, the offline version becomes much more approximable: solving the corresponding Linear Program optimally yields an almost integral solution, and the few fractional matches can be discarded without any significant loss in reward.

Remark 6. Arrival Models. We have not assumed anything about the query sequence in Definition 2. In the most general model called the *adversarial* model, the sequence is allowed to be completely arbitrary (as in Example 1; we consider this model in Sec. 4). We will also consider *stochastic* models of the query sequence (in Sec. 5), in which we can get algorithms with competitive ratio close to 1.

3 A family of algorithms

In this section we introduce a family of algorithms that are simple, intuitive, and often give near-optimal competitive ratios.

Definition 7. Auction-based algorithms: These algorithms have an interpretation as an auction, where a query j is allocated to the advertiser with the highest *normalized* bid $\nu_i b_{ij}$. Each advertiser i maintains a *bid normalizer* ν_i,

which we may update after each query is matched. Different algorithms in this family differ on how the normalizers are initialized, and how they are updated. The spend of the advertiser is still incremented by the original bid b_{ij}. The normalized bid is only used to make the allocation decision.

For the sake of simplicity, if we match query j to advertiser i where $b_{ij} > B_i - \text{spend}_i > 0$, we assume that the algorithm's reward is b_{ij}. This may result in at most a ρ difference in the competitive ratio, because these extra credits may account for at most a ρ fraction of the algorithm's revenue. Since we have made the small-bids assumption, this difference is not significant.

Intuition: Auction-based algorithms have a simple intuition. The bid normalizers provide a simple lever to make the budget of an advertiser last longer: the lower the normalizer of an advertiser, the fewer the queries matched to the advertiser, and the longer the budget lasts. They allow us to trade off an immediate gain of b_{ij} with a future advantage of keeping the budget of advertiser i alive.

We can interpret such algorithms as a generalization of the algorithm in Chapter **??**, where the value generated from a match is broken down into a revenue component and a utility component. In this context, the advertiser can be thought of as making an offer of $\nu_i b_{ij}$ (the revenue component) to the query, and the query chooses the advertiser with the highest offer.

Linear Programming (LP) duality: The family of Auction-based algorithms is not too restrictive because there exist fixed normalizers for each advertiser such that the resulting algorithm is almost optimal. This fact follows from the theory of *strong LP duality*, but we will not prove it here.[1] We only use what is called *weak LP duality*, to get an upper bound on OPT. The following is an LP relaxation of the problem, and its dual.

$$\begin{array}{c|c} \text{Primal: } \max \sum_{i,j} b_{ij} x_{ij} \text{ s.t.} & \text{Dual: } \min \sum_i \alpha_i B_i + \sum_j \beta_j \text{ s.t.} \\ \forall\ i, \sum_j b_{ij} x_{ij} \le B_i, & \forall\ i,j, \alpha_i b_{ij} + \beta_j \ge b_{ij}, \\ \forall\ j, \sum_i x_{ij} \le 1, & \\ \forall\ i,j, x_{ij} \ge 0. & \forall\ i,j, \alpha_i, \beta_j \ge 0. \end{array} \tag{2}$$

It is easy to see that the optimal allocation gives a feasible solution to the primal LP, by letting $x_{ij} = 1$ if and only if query j is matched to advertiser i. This gives us the following instantiation of weak LP duality, that any feasible dual solution gives an upper bound on the primal optimal.

Theorem 8. *For any feasible solution to the Dual LP, the dual objective function, $\sum_i \alpha_i B_i + \sum_j \beta_j$, is an upper bound on the Primal optimal, which in itself is an upper bound on* OPT.

[1] The argument involves some delicate tie breaking in degenerate cases, which will be a digression here.

The connection between this pair of LPs and Auction-based algorithms is via the *primal complementary slackness* condition:

$$x_{ij} > 0 \Rightarrow \alpha_i b_{ij} + \beta_j = b_{ij}.$$

Rewriting this as $\beta_j = b_{ij}(1 - \alpha_i)$, and the constraint in the Dual LP as $\beta_j \geq b_{ij}(1 - \alpha_i)$, we see that $x_{ij} > 0$ implies that i must maximize $b_{ij}(1 - \alpha_i)$ among all advertisers. This is exactly the allocation in an Auction-based algorithm if we set

$$\nu_i = 1 - \alpha_i \tag{3}$$

Lagrangian Relaxation: Auction-based algorithms are also equivalent to the algorithms you get using the technique of *Lagrangian relaxation*. This technique relaxes a constrained optimization problem to an unconstrained optimization problem, by moving the constraints into the objective function. A constraint such as $f(x) < c$ in a maximization problem goes into the objective function as $\lambda(c - f(x))$: thus violating a constraint is penalized. The parameter λ is called the Lagrangian multiplier, and it is equivalent to the dual variable α_i in our formulation. The objective function of this unconstrained optimization problem is called the Lagrangian function. We do a partial Lagrangian relaxation, where we only move the budget constraints into the objective function; this gives the Lagrangian function

$$L(\mathbf{x}, \alpha) = \sum_{i,j} b_{ij} x_{ij} + \sum_i \alpha_i (B_i - \sum_j b_{ij} x_{ij}) = \sum_{i,j} b_{ij} x_{ij} (1 - \alpha_i) + \sum_i \alpha_i B_i. \tag{4}$$

For a fixed α, this Lagrangian function is now separable over the queries. An Auction-based algorithm with normalizers $\nu_i = 1 - \alpha_i$ maximizes this Lagrangian function for each query j, subject to the constraint $\sum_i x_{ij} \leq 1$.

Our problem is online, and we don't know the optimal α_i values, therefore our algorithms will update the α_is as we go along.

4 Adversarial model

The classic model of query arrival is the *Adversarial* or *worst-case* model. Here, the algorithm has no advance knowledge about the query-side of the bipartite graph, and an adversary who knows the algorithm constructs the worst query sequence. In fact, for a deterministic algorithm like the one we will describe, one can even imagine the adversary generating each query after seeing the previous decisions of the algorithm.

The MSVV Algorithm: We present an Auction-based algorithm called MSVV which has the optimal performance in this model. Recall from Sec. 3, in order to specify an Auction-based algorithm for the AdWords problem, we need to define the bid normalizers ν_i for each advertiser i, over time as the query sequence unfolds. At any time t, let spend_i denote the current spend of advertiser

i. The algorithm takes the normalizer ν_i at time t as a function of the spend and its budget B_i, as follows:

$$\nu_i := 1 - e^{-\left(1-\frac{\text{spend}_i}{B_i}\right)} \tag{5}$$

The algorithm allocates the incoming query j to the advertiser i which maximizes $\nu_i b_{ij}$. After query j gets allocated (say to advertiser i), spend_i increases by b_{ij}, and consequently ν_i gets reduced correspondingly. The algorithm does not allocate the query to any advertiser if all $\nu_i b_{ij} = 0$.

Intuition: Note that initially, $\forall\ i : \text{spend}_i = 0$ and therefore $\nu_i = 1 - \frac{1}{e}$. Thus the very first query j to arrive will be allocated to the advertiser i with the highest bid b_{ij}. Over time, advertisers who have spent a greater fraction of their budget will have a lower normalizer, and would have to bid much higher to get allocated compared to advertisers who have not spent much of their budget. This trades off the immediate reward from a high bid with the goal of keeping advertiser budgets available for future queries in which some advertiser may have a high bid. Once an advertiser i finishes its entire budget, then $\nu_i = 0$, and hence it is not allocated any further queries. From a **pricing** or a **bidding** perspective, one can interpret the algorithm as saying that the offer-price or bid of an advertiser is shaded down dynamically based on the fraction of budget spent at that time.

One can prove that no matter what the query sequence is, MSVV achieves a revenue no worse than $1 - 1/e \simeq 0.63$ of OPT.

Theorem 9. *Algorithm MSVV achieves a competitive ratio of $1-1/e$, and this ratio is optimal among all algorithms, including randomized algorithms.*

Proof. We will follow the Primal-Dual LP framework from Sec 3. Recall that we do not know the LPs in advance, instead they are gradually revealed to us. We begin with primal and dual solutions x, α, β all set to 0. As we allocate the queries to advertisers using the algorithm's rule, we will update the primal and dual solutions in a way that three invariants hold throughout:

Invariant 1. x is a feasible solution for the (partial) primal LP.

Invariant 2. α, β form a feasible solution for the (partial) dual LP.

Invariant 3. The increase in the primal objective function is a $1 - 1/e$ fraction of the increase in the dual objective.

Note that the primal objective achieved at the end is precisely the algorithm's revenue. Thus, these invariants along with Theorem 8 immediately imply a $1 - 1/e$ competitive ratio:

$$\text{ALG} = \text{Primal} \geq (1 - 1/e)\ \text{Dual} \geq (1 - 1/e)\ \text{Primal optimal} \geq (1 - 1/e)\ \text{OPT}$$

It remains to define the updates which go hand-in-hand with the algorithm decisions. Define $\zeta := \frac{1}{1-1/e}$. If the algorithm allocates an incoming query j to an advertiser i, then we set x_{ij} and β_j, and increase α_i as follows:

$$x_{ij} = 1, \quad \beta_j = \zeta \nu_i b_{ij}, \quad \Delta\alpha_i = \tfrac{1}{B_i}\zeta(1-\nu_i)b_{ij} \tag{6}$$

We are ready to prove that the three invariants hold:

- Invariant 1 holds by the choice of the algorithm: it does not allocate a query to any advertiser who does not have sufficient budget remaining.

- Invariant 3 holds because the value of the primal LP increases by b_{ij}, while that of the dual LP increases by $\Delta\alpha_i B_i + \beta_j = \zeta(1-\nu_i)b_{ij} + \zeta\nu_i b_{ij} = \zeta b_{ij}$.

- Invariant 2 will be shown to hold iteratively: assuming that existing dual constraints are satisfied until the time some query j^* arrives (and is allocated to some advertiser i^*), we show that the update after query j^* keeps them feasible. Consider such a constraint for advertiser i and query j:

$$\alpha_i b_{ij} + \beta_j \geq b_{ij}$$

For a constraint pertaining to a previously arrived query $j \neq j^*$ and some i, the β_j remains unchanged, while the update for query j^* can only potentially increase α_i (and only for $i = i^*$). This keeps the inequality satisfied.
Thus the only interesting constraints are the ones corresponding to the new query j^*. For this, we will compute (for any i) the value of α_i after j^* has been allocated. Using spend_i^j and ν_i^j to denote the value of spend_i and ν_i at the time j arrives, we get, using the update rule from (6) and the algorithm's choice of normalizer from (5):

$$\alpha_i \;=\; \sum_{j \leq j^*:\; x_{ij}=1} \tfrac{1}{B_i}\zeta(1-\nu_i^j)b_{ij} \;=\; \sum_{j \leq j^*:\; x_{ij}=1} \tfrac{1}{B_i}\zeta e^{-\left(1-\frac{\text{spend}_i^j}{B_i}\right)} b_{ij}$$

With the small-bids assumption, we can approximate this by an integral, giving:

$$\alpha_i = \int_{x=0}^{\text{spend}_i^{j^*}} \tfrac{1}{B_i}\zeta e^{-\left(1-\frac{x}{B_i}\right)} dx \;=\; \frac{e^{\frac{\text{spend}_i^{j^*}}{B_i}} - 1}{e-1} = 1 - \zeta\nu_i \tag{7}$$

Since i^* is the advertiser maximizing $\nu_i b_{ij}$, we also have, from (6):

$$\beta_{j^*} = \zeta \nu_{i^*} b_{i^* j^*} \;\geq\; \zeta \nu_i b_{ij^*}, \quad \forall i \tag{8}$$

From (7) and (8), we get $\forall i: \;\; \beta_{j^*} \geq (1-\alpha_i)b_{ij^*}$, thus proving the dual constraints corresponding to j^* are also satisfied after j^* is allocated.

Optimality: To prove that no other algorithm can achieve a better competitive ratio, we find a distribution over input instances, and show that no deterministic algorithm can achieve a revenue better than a $1 - 1/e$ fraction of OPT, in expectation over this distribution, and then use the Minimax theorem to prove

the desired optimality statement. While we leave the details as out of scope, we mention that this is the same approach as used in Chapter **??** to prove optimality in the basic online bipartite matching problem, and the distribution over instances also turns out to be very similar to the one used there. □

Remark 10. As opposed to an offline algorithm which has access to an optimal set of normalizers ν^* (as in Sec. 3), MSVV has to update its normalizers as the query sequence unfolds. The function (5) computes the normalizer as a function of the spend, and can be considered to be an online approximation to the optimal ν^* for every instance. Since we can only hope to find an approximate dual, it also turns out that the relation between the dual α_i and the normalizer ν_i in (7) is a scaled version of the optimal relation (3).

5 Stochastic models

The adversarial model requires that the algorithm hedge against all possible inputs. As a result the MSVV algorithm can be sub-optimal for benign instances that are likely to occur in real applications. Real world instances often follow repeated patterns, therefore an algorithm that uses historic data can get a better revenue. With simple stochastic assumptions about the arrival of queries, we can breach the $1 - 1/e$ upper bound on the competitive ratio in the adversarial model. The most significant result here is that the competitive ratio approaches 1 as the bid to budget ratio tends to 0. In this section, we demonstrate one such algorithm in the following stochastic model, which is similar to the model for the "Secretary problem" discussed in Chapter **??**.

Definition 11. In the *Random permutation model*, we assume that an adversary still picks the *set* of queries, the advertiser bids, and budgets, but the *order* of arrival is permuted uniformly at random. This does not change OPT, but we now measure the algorithm's performance in expectation over this random permutation.

While we are interested in algorithms which achieve a ratio approaching 1, we note here that even the algorithms that we have already seen perform better in this model. The simple highest-bidder strategy now achieves a competitive ratio of $1 - 1/e$, and MSVV achieves a ratio of at least 0.76.

Compared to the adversarial model, we make two additional assumptions. The first is required for the results, while the second is for ease of exposition.

Additional assumptions:

- We assume that the number of queries m is known to the algorithm; without such an assumption, no algorithm can get a competitive ratio that approaches 1 as the bid to budget ratio tends to 0. We also assume that m is large enough: $m \geq 1/\rho$ (recall that ρ is the bid to budget ratio, defined in Section 2).

- For simplicity of exposition, we assume that we know OPT. In fact, it is sufficient to have an estimate $\widehat{\text{OPT}}$ such that $\text{OPT} \leq \widehat{\text{OPT}} \leq c\text{OPT}$ for some universal constant c. We will prove a competitive ratio of the form $1 - O(\epsilon)$, and the constant c appears inside the $O(\epsilon)$ term. We can get such an estimate from the first ϵ fraction of the queries. Ignoring these queries decreases the competitive ratio by an additional ϵ factor.

A control mechanism: We use an Auction-based algorithm. In contrast to update (6), the normalizers can both increase as well as decrease. The update acts as a control mechanism, trying to maintain a constant rate of budget spent. If the budget of an advertiser is spent too quickly, it decreases the normalizer, resulting in fewer queries matched, and a lower rate of spend; conversely, if the budget is not spent quickly enough, it increases the normalizer. To determine the precise update formula, we use the *multiplicative weight update (MWU)* algorithm for what is known as the *learning from experts* problem. We define these next.

Definition 12. *Learning from experts* is another online problem, where there are N experts, and in each round $t \in [T]$, each expert $i \in [N]$ gets a reward of $r_{i,t}$. The algorithm has to pick one expert in each round and gets the same reward as that expert, but it has to choose the expert before seeing the rewards. The goal is to maximize the total reward of the experts picked in all the rounds, in order to minimize the *regret*, which is the difference between the expected reward of the algorithm and the reward of the best single expert in hindsight.

The MWU algorithm: The MWU algorithm is *randomized*: it picks expert i in round t with probability $\theta_{i,t}$. The algorithm maintains a weight for each expert, which are all initialized to 1. After each round, the weights are updated multiplicatively with an exponent proportional to the rewards. The probabilities $\theta_{i,t}$ are normalized versions of the weights $w_{i,t}$, so that they sum up to 1. Let $\epsilon \in [0, 1/2]$ be a parameter of the algorithm. The update is

$$w_{i,t+1} = \begin{cases} w_{i,t}(1+\epsilon)^{r_{i,t}} & \text{if } r_{i,t} \geq 0. \\ w_{i,t}(1-\epsilon)^{-r_{i,t}} & \text{if } r_{i,t} < 0. \end{cases} \tag{9}$$

The MWU algorithm has the following regret guarantee.

Theorem 13. *Suppose that the rewards are all in* $[-1, 1]$*. For any expert* $i \in [N]$*, we have that*

$$\sum_{t \in [T], i \in [N]} \theta_{i,t} r_{i,t} \geq \sum_t r_{i,t} - \epsilon \sum_t |r_{i,t}| - \frac{\ln N}{\epsilon}. \tag{10}$$

The Algorithm: We abuse notation and denote the query that appears at time step t as $j(t)$; here $j(\cdot)$ is a random permutation of $[m]$. We denote $x_{ij} = 1$ to indicate that the algorithm matched query j to advertiser i, and set $x_{ij} = 0$

otherwise. We have one expert for each advertiser, and an additional expert that corresponds to not matching the query to anyone, i.e., $N = n + 1$. We set the rewards as follows.

$$\forall i \in [n], r_{i,t} = \left(\frac{b_{i,j(t)} x_{i,j(t)}}{B_i} - \frac{1}{m} \right) \frac{1}{\rho}. \tag{11}$$

The scaling by $1/\rho$ makes the rewards as large as possible while still keeping them in $[-1, 1]$. The reward of the expert $n + 1$ is always 0. We can see that the cumulative reward at any time is proportional to the difference between the fraction of the budget spent and the fraction of the time elapsed.

The Lagrange multipliers in an Auction-based algorithm capture the trade-off between the opportunity to increase the objective vs. the risk of violating a constraint, but the $\theta_{i,t}$s being a probability distribution do not have the right scale. We define $\alpha_{i,t}$ by multiplying $\theta_{i,t}$ by a factor of OPT, to get it to the scale of the objective, and dividing by B_i, to capture the scale of the constraint.

$$\alpha_{i,t} := \frac{\theta_{i,t} \text{OPT}}{B_i}. \tag{12}$$

As mentioned in Section 3, the normalizer is $\nu_{i,t} = 1 - \alpha_{i,t}$. When all the $\alpha_{i,t}$s are greater than 1, we do not match the query to any advertiser.

Initialize parameter $\epsilon \in [0, \frac{1}{2}]$.
Initialize for all $i \in [n + 1]$, $\theta_{i,1}$ and $w_{i,1}$ as in the MWU algorithm.
for all $t = 1, ..., m$ **do**
  For all $i \in [n]$, set $\alpha_{i,t}$ as in (12).
  if $\forall i \in [n]$, $\alpha_{i,t} > 1$ **then**
    Set $x_{i,j(t)} = 0$ for all $i \in [n]$.
  else
    Let i^* be the highest normalized bidder, $\arg\max_{i \in [n]} b_{i,j(t)}(1 - \alpha_{i,t})$.
    if Matching $j(t)$ to i^* exceeds their budget **then**
      Exit.
    else
      Match $j(t)$ to i^*. Set $x_{i^*,j(t)} = 1$, and $x_{i,j(t)} = 0$ for all $i \neq i^*$.
    end if
  end if
  $\forall i \in [n]$, set rewards $r_{i,t}$ as in (11). Set $r_{n+1,t} = 0$.
  Update $w_{i,t+1}, \theta_{i,t+1}$ as in the MWU algorithm (9).
end for

Algorithm 1: Algorithm for Random permutation model

Intuition: While it may seem counter-intuitive that spending more results in higher rewards, notice that a higher probability $\theta_{i,t}$ of picking an expert i leads to a lower normalizer $\nu_{i,t}$, which is the desired direction of change. The Lagrangian interpretation of this is that a higher spend results in a higher

chance of violating the constraint, and therefore we should increase the *penalty* α_i. For advertisers that are budget constrained, the algorithm tries to keep the fraction of budget spent close to the fraction of time elapsed. For the rest of the advertisers, we expect a negative reward on average, which drives the weight down to zero, and the normalizer to one; clearly, there is no benefit from discounting the latter's bid.

Let ALG denote the objective realized by Algorithm 1. The following theorem shows that it achieves a competitive ratio of $1 - o(1)$.

Theorem 14. *In the random permutation model, for any $\epsilon \in [0, \frac{1}{2}]$ such that $\epsilon^2 \geq \rho \ln(n+1)$, we have that $\mathbb{E}[\text{ALG}] \geq \text{OPT}(1 - O(\epsilon))$, i.e., there is a universal constant c such that $\mathbb{E}[\text{ALG}] \geq \text{OPT}(1 - c\epsilon)$.*

Proof. For the sake of simplicity, we analyze the algorithm assuming that the queries are sampled *with replacement*, instead of sampling without replacement as in the random permutation model. In the rest of this proof, we assume that each query $j(t)$ is an i.i.d. sample from the set $[m]$ of all queries. Handling sampling without replacement results in additional terms that can also be bounded by $O(\epsilon)\text{OPT}$, but this is outside the scope of this chapter.

Let $x^*_{i,j}$ denote the offline optimal matching. Let T be the last query that we succesfully matched, i.e., matching query $j(T+1)$ would exceed the chosen advertiser's budget. If this never happens, then let $T = m$. The proof is broken down into 3 steps.

1. Auction: Since for each $t \in [T]$, the algorithm matches the query $j(t)$ to the advertiser with the highest normalized bid $(1 - \alpha_{i,t})b_{i,j(t)}$, the normalized bid of the advertiser matched to $j(t)$ in the offline optimal matching is only lower:

$$\sum_{i \in [n]} b_{i,j(t)}(1 - \alpha_{i,t}) x_{i,j(t)} \geq \sum_{i \in [n]} b_{i,j(t)}(1 - \alpha_{i,t}) x^*_{i,j(t)}.$$

Summing up this inequality over all $t \in [T]$, and noting that $\text{ALG} = \sum_{t \in [T], i \in [n]} b_{i,j(t)} x_{i,j(t)}$, we have that

$$\text{ALG} \geq \sum_{t \in [T], i \in [n]} \left(b_{i,j(t)}(1 - \alpha_{i,t}) x^*_{i,j(t)} + b_{i,j(t)} \alpha_{i,t} x_{i,j(t)} \right). \tag{13}$$

2. Stochasticity: The assumption of sampling queries with replacement helps us replace the terms corresponding to the offline optimal matching in (13) with their expectation.

For each i and t, conditioned on all the queries that appeared up to time $t-1$, the expectation of $b_{i,j(t)} x^*_{i,j(t)}$ is less than $\frac{B_i}{m}$. This is because each new draw is independent of everything that came before, and $\sum_{j \in [m]} b_{ij} x^*_{i,j} \leq B_i$. Similarly, for each t, the expectation of $\sum_{i \in [n]} b_{i,j(t)} x^*_{i,j(t)}$ is $\frac{1}{m}\text{OPT}$. Taking expectations, and using these to replace the terms corresponding to the optimal

allocation in (13), we have that

$$\mathbb{E}\left[\text{ALG}\right] \geq \mathbb{E}\left[\frac{T}{m}\text{OPT} + \sum_{t\in[T],i\in[n]} \alpha_{i,t}\left(b_{i,j(t)}x_{i,j(t)} - \frac{B_i}{m}\right)\right]. \tag{14}$$

3. The MWU guarantee: After proper scaling, the second term inside the expectation on the RHS in (14) is exactly the reward of the MWU algorithm, which we can bound using the regret guarantee in Theorem 13. In fact, we can bound these terms with probability 1, and not just in expectation, to show that the RHS of (14) is at least $(1 - O(\epsilon))\text{OPT}$. This bound is summarized in Lemma 15, which completes the proof. □

Lemma 15.

$$\frac{T}{m}\text{OPT} + \sum_{t\in[T],i\in[n]} \alpha_{i,t}\left(b_{i,j(t)}x_{i,j(t)} - \frac{B_i}{m}\right) \geq (1 - O(\epsilon))\text{OPT}.$$

Proof. The main idea behind this proof is to apply the regret guarantee for the MWU algorithm as stated in Theorem 13, by suitably choosing the expert to compare with. The rewards for the experts problem have been defined so that the second term in the LHS in the lemma statement is proportional to the expected reward of the MWU algorithm. This follows from using the definition of $\alpha_{i,t}$ and $r_{i,t}$:

$$\begin{aligned}\sum_{t\in[T],i\in[n]} \alpha_{i,t}\left(b_{i,j(t)}x_{i,j(t)} - \frac{B_i}{m}\right) &= \sum_{t\in[T],i\in[n]} B_i\alpha_{i,t}\left(\frac{b_{i,j(t)}x_{i,j(t)}}{B_i} - \frac{1}{m}\right) \\ &= \rho\text{OPT}\sum_{t\in[T],i\in[n]} \theta_{i,t}r_{i,t}. \end{aligned}\tag{15}$$

Next, using Theorem 13, we can relate it to the reward of any one expert $i' \in [n+1]$.

$$\begin{aligned}\rho\text{OPT}\sum_{t\in[T],i\in[n]} \theta_{i,t}r_{i,t} &\geq \left(\sum_{t\in[T]}(r_{i',t} - \epsilon|r_{i',t}|) - \tfrac{\ln(n+1)}{\epsilon}\right)\rho\text{OPT} \\ &\geq \left(\sum_{t\in[T]}(r_{i',t} - \epsilon|r_{i',t}|)\right)\rho\text{OPT} - \epsilon\text{OPT} \end{aligned}\tag{16}$$

Inequality (16) follows from the fact that $\epsilon^2 \geq \rho\ln(n+1)$, as assumed in the hypothesis of Theorem 14. Now comes the main part, where we instantiate the choice of expert i' appropriately, so that the RHS of (16) is at least $(1 - O(\epsilon) - \frac{T}{m})\text{OPT}$. We consider 2 cases, where we choose either the expert whose spend exceeds the budget, or the $n+1^{\text{st}}$ expert, when no advertiser runs out of budget.

Case 1, no advertiser runs out of budget: $T = m$. In this case, we choose $i' = n + 1$, for whom the rewards are always zero. We therefore have that

$$\sum_{t \in [T]} (r_{i',t} - \epsilon |r_{i',t}|) = 0 \geq 1 - \epsilon - \tfrac{T}{m},$$

because $T = m$. Hence, the lemma follows.

Case 2, some advertiser runs out of budget: In this case, the algorithm stops because in time step $T+1$, matching $j(T+1)$ as per the algorithm would exceed the chosen advertiser's budget. Let this advertiser be i'. From the definition of ρ, we have that advertiser i' has spent at least a $1 - \rho$ fraction of their budget:

$$\sum_{t \in [T]} b_{i',j(t)} x_{i',j(t)} \geq B_{i'} - b_{i',j(T+1)} \geq B_{i'}(1 - \rho). \tag{17}$$

From the definition of the reward in (11), this implies that ρ times the total reward of expert i' up to time T is at least $1 - \rho - \frac{T}{m} \geq 1 - \epsilon - \frac{T}{m}$, because as per the hypothesis in Theorem 14, we have that $\rho \leq \epsilon^2 / \log n < \epsilon$. If (16) only had the reward terms, we would be done, but we also need to bound the terms $\epsilon |r_{i',t}|$. Nonetheless, a similar argument works. From the definition of the reward in (11), and triangle inequality,

$$\begin{aligned} |r_{i',t}| \rho &\leq \frac{b_{i',j(t)} x_{i',j(t)}}{B_{i'}} + \frac{1}{m} \\ \Rightarrow (r_{i',t} - \epsilon |r_{i',t}|) \rho &\geq (1 - \epsilon) \frac{b_{i',j(t)} x_{i',j(t)}}{B_{i'}} - \frac{1 + \epsilon}{m}. \end{aligned}$$

Summing this over all $t \in [T]$, and using (17), we get that

$$\begin{aligned} \sum_{t \in [T]} (r_{i',t} - \epsilon |r_{i',t}|) \rho &\geq \sum_{t \in [T]} \left((1 - \epsilon) \frac{b_{i',j(t)} x_{i',j(t)}}{B_{i'}} - \frac{1 + \epsilon}{m} \right) \\ &\geq (1 - \epsilon)(1 - \rho) - (1 + \epsilon) \frac{T}{m} \\ &\geq 1 - 3\epsilon - \frac{T}{m}, \end{aligned}$$

where the last inequality is true because $\rho \leq \epsilon$ and $T \leq m$. This implies that the RHS of (16) is at least $(1 - 4\epsilon - \frac{T}{m})\mathrm{OPT}$. The lemma follows.

□

6 Packing mixed integer linear programs

Besides the AdWords problem, there are a variety of problems that arise in the space of ad allocation, mostly because of different types of constraints and

input models. In one variant, instead of a budget constraint, we have a *capacity* constraint: a limit on the number of matches to an advertiser. Such constraints occur widely in *display advertising*, which refers to banner ads shown on web pages and apps. The capacity constraint gives rise to the problem of *online edge weighted bipartite matching problem.* A $1 - \frac{1}{e}$ worst case competitive algorithm for this problem, with a *free disposal* assumption, is presented in Chapter **??**. In this section we present a general class of problems that captures many such variants.

***Packing Mixed Integer Linear Programs* (Packing MILPs):** In this abstraction, each $j \in [m]$ corresponds to a request for ad allocation. We have *local* constraints (20) that specify all the different ways we may fulfill this request. We abstract these local constraints by saying that the vector $(x_{ij})_{i\in[n]}$ belongs to the set $\mathcal{A}_j$; this is typically modeled as a mixed integer-linear constraint. More generally, this set could be any discrete or continuous set, and we abstract out the details by requiring that we have a computationally efficient algorithm that maximizes a linear objective function over this set. We assume that not fulfilling a request is always an option, by requiring that the 0 vector is always in $\mathcal{A}_j$. The equivalent of the small bids assumption here is that $\mathcal{A}_j \subseteq [0, \rho]^n$. The *global* constraints (19) tie the different ad allocation requests together in the form of packing constraints. We consider a normalized version of these constraints, where the right hand side values are all 1s. Such a problem is captured by the following mathematical program.

$$\text{Maximize} \sum_{i\in[n], j\in[m]} v_{ij}x_{ij} \text{ subject to:} \tag{18}$$

$$\forall i \in [n], \sum_{j\in[m]} x_{ij} \leq 1 \tag{19}$$

$$\forall j \in [m],\ (x_{ij})_{i\in[n]} \in \mathcal{A}_j \subseteq [0, \rho]^n \tag{20}$$

Matching problems as special cases: In the AdWords problem, the global constraints are the budget constraints. The local constraint is that a query may be matched to at most one advertiser. This is modeled by defining $\mathcal{A}_j$ to be a set of $n + 1$ vectors in n dimensions, where the i^{th} vector has the bid as a fraction of the budget for the i^{th} advertiser in the i^{th} dimension, and 0 for all other dimensions. The $n + 1^{\text{st}}$ vector is the 0 vector. In the edge weighted bipartite matching problem, the global constraints are the capacity constraints, and the local constraints are matching constraints. The set $\mathcal{A}_j$ is a set of $n + 1$ vectors in n dimensions, where the i^{th} vector has the inverse of the capacity of the i^{th} advertiser in the i^{th} dimension, and 0 for all other dimensions, with the $n + 1^{\text{st}}$ vector being 0.

An algorithm for the stochastic model: For the class of Packing MILPs, it is impossible to get *any* constant factor approximation in the adversarial

model without problem specific assumptions such as in the edge weighted matching with free disposal problem. In the stochastic model, we can generalize the algorithm from Sec. 5 to achieve a $1 - O(\epsilon)$ competitive ratio. The algorithm follows an identical framework. As before, we assume that we know OPT. We again use the MWU algorithm as a subroutine, with $n+1$ experts, one for each constraint, and one corresponding to not fulfilling the request. At any time $t \in [m]$, we maintain Lagrange multipliers $\alpha_{i,t}$ for each $i \in [n]$, by scaling the probability $\theta_{i,t}$ of playing expert $i \in [n]$ as per the MWU algorithm, as follows: $\alpha_{i,t} := \text{OPT}\theta_{i,t}$. We choose the vector $(x_{ij})_{i\in[n]} \in \mathcal{A}_j$ by maximizing the Lagrangian function restricted to that request:

$$\textstyle\sum_{i\in[n]}(v_{ij} - \alpha_{i,t})x_{ij}.$$

If at any time, this choice causes a constraint to be violated, then we skip fulfilling this and all subsequent requests. Otherwise, we set the reward of expert $i \in [n]$ as follows, and update the expert probabilities for the next step using the MWU algorithm. The reward of expert $n+1$ is always 0.

$$\forall i \in [n], r_{i,t} = \left(x_{i,j(t)} - \tfrac{1}{m}\right)\tfrac{1}{\rho}. \tag{21}$$

We can show that this algorithm is $1 - O(\epsilon)$ competitive for the random permutation model. The theorem statement and proof are very similar to that of Thoerem 14; a detailed presentation is outside the scope of this chapter.

7 Autobidding: A decentralized approach to matching

So far, we have framed ad allocation as a centralized matching problem, in which the auctioneer factors advertiser budgets into the matching decisions. In this section, we take a different view of the problem: here, the auctioneer runs a simple auction for each query, and does not even have knowledge of the budgets. Instead, the constraints are managed via *autobidding*. As opposed to manual bidding in which an advertiser specifies a per-keyword bid, an autobidding system allows advertisers to express their high level goals and constraints, and automatically translates those into per-auction bids.

We will first formalize this problem (Sec. 7.1) and present an optimal bidding algorithm from one advertiser's point of view (Sec. 7.2). We then connect this back to the matching problem, and present analytical results on the quality of the matching derived in such a decentralized approach (Sec. 7.3).

7.1 Formulation of autobidding under constraints

We present a very general formulation of the autobidding problem, but it is useful to keep in mind an important motivating example called Target Cost-per-Acquisition (henceforth, TCPA). Here the advertiser's goal is to maximize the acquisitions (sales derived from the ad campaign, also known as conversions),

subject to an upper bound on the average cost-per-acquisition (CPA). The upper bound (itself called the target-CPA) will be denoted as T.

Fix an advertiser for whom we are designing the bidding agent. Let p_j be the price of an ad on query j for this advertiser (for simplicity, we will assume in this chapter that there is only a single ad-slot per query). Note that p_j depends on the bids of the other advertisers who may themselves be solving such an optimization problem via a bidding agent; we will visit this interaction in Sec. 7.3. For now assume that the price p_j of each query j is fixed. We further assume for now that we know the entire query sequence in advance, as well as the values of the p_js. Then, we can formulate the following *selection problem*, i.e., which queries would the advertiser like to buy so as to maximize their objective while staying within their constraints.

Consider the following abstract LP (on the left) with a set of constraints indexed by $\mathcal{C}$, and non-negative constants v_j, B^c, and w_j^c, for queries j and constraints $c \in \mathcal{C}$. The x_j are decision variables for whether or not to buy query j at a cost of p_j. The constant v_j stands for the value that the advertiser derives from the ad on query j, and the other constants are set to capture the different constraints. The LP on the right is its dual, which we will use shortly.

$$\begin{aligned} \text{Maximize } & \textstyle\sum_j v_j x_j \quad \text{s.t.} && (22) \\ \forall c \in \mathcal{C} : & \textstyle\sum_j p_j x_j \le B^c + \sum_j w_j^c x_j \\ & x_j \le 1 && (23) \\ & x_j \ge 0 \end{aligned} \qquad \begin{aligned} \text{Minimize } & \textstyle\sum_j \delta_j + \sum_c \alpha_c B^c \quad \text{s.t.} && (24) \\ \forall\ j : & \ \delta_j \ge \textstyle\sum_c \alpha_c (w_j^c - p_j) + v_j && (25) \\ & \forall\ j : \delta_j \ge 0 \\ \forall\ c \in \mathcal{C} : & \ \alpha_c \ge 0 \end{aligned}$$

Note that this is a fractional version of the selection problem, not integral ($x_j \in \{0, 1\}$). We will disregard this difference, since an optimal solution for an instance in general position has at most $|\mathcal{C}|$ non integral x_j which can be set to 0 without much loss in objective value or constraint violation for large markets.

Examples: This primal LP captures a wide range of autobidding strategies that are offered to advertisers. For example, TCPA is a special case: Let cvr_j denote the conversion-rate, which is the probability that an ad for the advertiser on query j results in a conversion; this prediction is made by a machine learning system at the time when query j arrives. Then the objective is obtained by setting $v_j = \text{cvr}_j$, and the TCPA constraint by setting the corresponding $B^c = 0$ and $w_j^c = T \cdot \text{cvr}_j$. As a further example, we can add a budget constraint by setting the corresponding B^c as the budget and $w_j^c = 0$.

7.2 Optimal bidding algorithm

We now leverage the LP formulation to come up with a bidding formula which can achieve the same optimal choice of queries as in the selection problem. The

dual constraint (25) can be re-written as:

$$\forall\ j : \frac{\delta_j}{\sum_c \alpha_c} \geq \left(\frac{v_j + \sum_c \alpha_c w_j^c}{\sum_c \alpha_c} - p_j \right) \tag{26}$$

We will use the right-hand side (rhs) of (26) as our bidding formula (assuming we know an optimal dual solution). Set the bid for query j to be

$$\text{bid}(j) := \frac{v_j + \sum_c \alpha_c w_j^c}{\sum_c \alpha_c} \tag{27}$$

Theorem 16. *Assuming that we have access to optimal values of the dual variables α_c, the bid formula (27) results in an auction outcome identical to an optimal primal solution x_j, if the underlying auction is truthful.*

Proof. With this bid, if the advertiser wins the query j in the auction, that means that the rhs of (26) is positive (ignoring ties), and therefore $\delta_j > 0$ in the optimal solution to the LP. By complementary slackness conditions, noting that δ_j is the dual variable for the primal constraint (23), the following holds in an optimal solution: $\delta_j > 0 \Rightarrow x_j = 1$. Thus, the bid only wins queries j which are in the optimal primal solution.

On the other hand, if the advertiser loses query j in the auction, this means the the rhs of (26) is negative, and since $\delta_j \geq 0$, the constraint (26) can not be tight. Again by complementary slackness condition, noting that x_{ij} is the primal variable corresponding to this constraint, we get that $x_{ij} = 0$ in the optimal primal solution. Thus, the advertiser's bid does not lose any queries from the optimal selection.

In summary, since the underlying auction is truthful, the advertiser wins precisely the auctions in which the bid (27) is at least the price (highest bid among other bidders), and also wins precisely the queries chosen by the optimal primal solution. □

Remark 17. Note that while we have presented the simple case of one ad-slot per query, the proof holds more generally, e.g., in a position auction when the advertiser's ad can be placed in one of multiple ad-slots in a query. The optimal bid results in both selecting the highest utility option (due to the truthfulness of the auction), and making the same selection as the optimal primal solution (due to the complementary slackness conditions).

Examples: For TCPA, with α_T as the dual for the TCPA constraint, we get:

$$\text{bid}_{\text{TCPA}}(j) = \left(T + \frac{1}{\alpha_T} \right) \text{cvr}_j \tag{28}$$

Adding a budget constraint to TCPA gives another dual variable α_B yielding

$$\text{bid}(j) = \left(\frac{1 + T \cdot \alpha_T}{\alpha_T + \alpha_B} \right) \text{cvr}_j$$

The bid formula depends on the knowledge of the optimal duals α_c. In practice, the duals can be estimated from past data logs, and updated online in a control loop depending on the state of the corresponding constraint. In fact LP (22) is a special case of the MILP (18) in Sec. 6, therefore we can use the algorithm from that section.

7.3 The price of anarchy: sub-optimality of the decentralized approach

We now ask what would happen if all advertisers use the optimal autobidding agents to bid. What would be the efficiency of the eventual matching? This is tricky to answer because each advertiser's bidding LP depends on all the other advertisers' LPs, as the price p_j is determined from all the bids in the auction for j. For ease of exposition, we restrict the analysis in this section to the special case of TCPA autobidding; a slightly more involved analysis works for the general autobidding LP (22).

Firstly, it can be shown that under a large market assumption, an equilibrium does exist for any setting of advertiser target-CPA values. That is, if each advertiser uses the bidding formula (28) to convert their target-CPA value to auction bids, then there is a stable outcome: a set of dual variables which are consistent with each other via the auction prices. The proof is via an application of Brouwer's fixed point theorem and we do not include it here. The question we are interested in is, what is the loss in efficiency of the matching produced by this decentralized (bidding- and auction-based) system, compared to an optimal centralized matching solution. For this we need to define the metric for efficiency of a matching in this setting.

Constrained Welfare: Since each advertiser in TCPA autobidding aims to maximize the number of conversions, a natural efficiency metric would be the total number of conversions across all the advertisers. However, this turns out not to be a very useful metric. This is because matchings obtained via bidding and auction have to satisfy the advertisers' spend constraints, while an optimal matching can achieve a very high value by allocating queries for free. For example, if there is an advertiser with a high conversion-rate for all queries but with very stringent constraints on spend then the optimal matching can achieve an unreasonably high objective by allocating all queries to this advertiser for free. To bring the comparison between an optimal matching and an auction-implemented matching to common ground, we introduce the constrained welfare of a matching, defined for the general bidding LP (22) as follows.

Definition 18.

$$\text{Constrained Welfare } := \sum_{i \in [n]} \min_{c \in \mathcal{C}} (B_i^c + \sum_j w_{ij}^c x_{ij})$$

Here, recall that $[n]$ is the set of advertisers, and $\mathcal{C}$ is the set of constraints. The x_{ij} are the decision variables for whether query j gets matched to advertiser i,

and the B_i^c and w_{ij}^c are the constants in the ith advertiser's LP. The definition takes the smallest right-hand side of the constraints for each advertiser and adds them up. Noting that all the left-hand sides are the spends of the advertisers, we can see that this captures the *maximum willingness to spend* for any given matching. Note that for the special case of TCPA autobidding, this becomes:

$$\text{Constrained Welfare (TCPA)} = \sum_{i\in[n]} T(i) \cdot \sum_j x_{ij}\text{cvr}_{ij} \tag{29}$$

where $T(i)$ is the target-CPA bound for advertiser i, and cvr_{ij} is the conversion rate for i's ad on query j. This is the total *target-CPA-weighted* conversions, which is intuitively the correct aggregation of conversions across advertisers.

Definition 19. With this definition of welfare, we can now define our sub-optimality measure, the **Constrained Price of Anarchy** as:

$$\text{CPoA} := \max_{I\in\mathcal{I}} \min_{\text{EQ}(I)} \frac{\text{CW}(\text{OPT}(I))}{\text{CW}(\text{EQ}(I))}$$

Here $\mathcal{I}$ is the set of all possible instances, CW denotes constrained welfare, $\text{OPT}(I)$ is the matching which maximizes CW for I, and $\text{EQ}(I)$ denotes the set of matchings achieved in any equilibrium of the decentralized approach. We can now bound sub-optimality of the equilibrium matching.

Theorem 20. *For the general autobidding setting (LP(22)),* $\text{CPoA} \le 2$.

Proof. As mentioned earlier, for simplicity of exposition, we will only prove this here for the special case of TCPA. Fix an optimal matching OPT and any equilibrium matching EQ. Let Q_1 be the set of queries which EQ allocates to the same advertiser as OPT does, and let Q_2 be the rest of the queries. Denote the contribution to the CW (as defined in (29)) from a subset Q of queries in OPT and in EQ as $\text{CW}(\text{OPT} \mid Q)$ and $\text{CW}(\text{EQ} \mid Q)$ respectively. By definition, the contribution of Q_1 is identical for EQ and OPT. Thus

$$\text{CW}(\text{EQ}) \;\ge\; \text{CW}(\text{EQ} \mid Q_1) \;=\; \text{CW}(\text{OPT} \mid Q1) \tag{30}$$

Define $\text{spend}_{eq}(i)$ as the total spend of advertiser i in the equilibrium solution, and $\text{spend}_{eq}(j)$ as the spend on query j in the equilibrium solution. Let $\text{OPTB}(j)$ denote the advertiser to which OPT matches query j. For queries in Q_2, the following holds:

$$\begin{aligned}
\text{CW}(\text{EQ}) \;&\ge\; \sum_{i\in[n]} \text{spend}_{eq}(i) &&= \sum_{j\in Q} \text{spend}_{eq}(j) \;\ge\; \sum_{j\in Q_2} \text{spend}_{eq}(j) \\
&\ge\; \sum_{j\in Q_2} \text{bid}(\text{OPTB}(j), j) &&\ge \sum_{j\in Q_2} T(\text{OPTB}(j)) \cdot \text{cvr}(\text{OPTB}(j), j) \\
&=\; \text{CW}(\text{OPT} \mid Q_2)
\end{aligned} \tag{31}$$

The first inequality is from the general definition of CW (Def. 18) and the constraints in the LP. The third inequality is because the advertiser $\text{OPTB}(j)$

is one of the price setters for j and we have a second price auction. The last inequality is from the bidding formula (28) for the advertiser $\text{OPTB}(j)$. The last equality is from the definition of CW for TCPA (29). Adding up (30) and (31) completes the proof. □

Price of Anarchy for budgeted matching: For the case where we only have a budget constraint, Constrained Welfare is just the sum of the budgets, which is a constant independent of the matching. For this case, we consider a different way of aggregating the values of advertisers, called the **Nash welfare:** it is the budget weighted geometric mean of the values. Let V_i be the total value that the player i gets in the matching, then Nash welfare is defined as

$$\prod_{i \in [n]} V_i^{\frac{B_i}{\sum_{j \in [n]} B_j}}$$

As mentioned earlier, the reported values of advertisers with different budgets are not directly comparable. Nash Welfare elegantly handles this issue with the following nice invariance property: if we multiply all the values of any one advertiser i by a constant c, the Nash welfare of *every* matching that gives a non zero value to advertiser i gets multiplied by $c^{\frac{B_i}{\sum_{j \in [n]} B_j}}$. Thus, such a scaling of values would have no affect on the Price of Anarchy. We can show that the price of anarchy w.r.t. Nash welfare is also 2. The proof is similar in spirit to that of Theorem 20, therefore we do not include it here.

Theorem 21. *The Price of Anarchy with respect to Nash welfare for budgeted matching is at most* 2.

7.4 Equilibrium dynamics under different auctions

If each advertiser uses an algorithm to adjust bids independently, would we eventually reach an equilibrium, or perhaps just go around in cycles? Would we only reach some equilibria, and not others? Even if we do reach an equilibrium eventually, how long would it take to get there? Recently, it was shown that it is PPAD-Hard to find one of these equilibria, which indicates that it is unlikely that such dynamics will always reach an equilibria in polynomial time. Beyond that, these questions are largely unanswered. We illustrate some of these issues for a specific example in Figure 2. We consider the budgeted matching setting, with two advertisers. Each of them has a Lagrange multiplier α_i which they update in a manner similar to the algorithm for the random permutation model: decrease α_i if spend is under budget, and increase α_i if spend is over budget. The axes in the figure are the α_is, and the arrows show the direction of such an update. Each dark line is the contour of α_is where one of the two advertisers spends exactly his budget. The intersections of the two dark lines are exactly the set of equilibria. There are three equilibria, but the one at the center is not **stable**, i.e., if you perturb this equilibrium by a small amount, then the

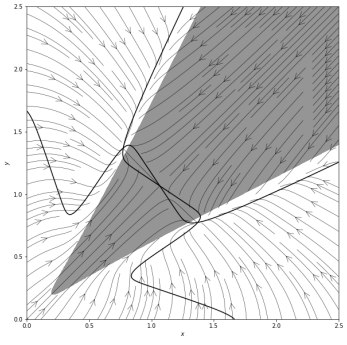

Figure 2: Equilibria in shaded regions are unstable in second price auctions.

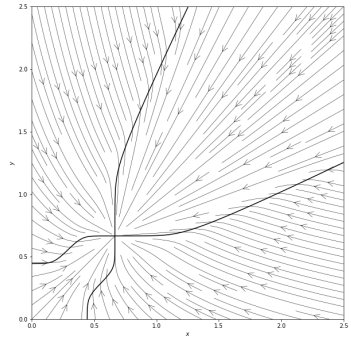

Figure 3: Dynamics converge to the unique equilibrium in first price auctions.

dynamics would move away from it. Thus the dynamics would never converge to this equilibrium, even though the instance is symmetric and this is the only symmetric equilibrium.

The shaded region characterizes when equilibria are stable vs. not. The shaded region only depends on the values, but not the budgets. As we change the budgets, the direction of the updates, and hence the locations of the equilibria change. Whenever an equilibrium happens to be in the colored region, it is unstable, and when it is in the white region, it is stable. Other (empirical) observations are that there are an odd number of equilibria, and they alternate between being stable and unstable.

Figure 3 shows a similar plot when we use a first price auction. Here we have a unique equilibrium, and the dynamics converge to this equilibrium. This can be proven formally via a potential function, where the dynamics become equivalent to gradient descent for this potential function.

8 Bibliographic notes

The AdWords problem was introduced in Mehta et al. (2007) which also provided the optimal MSVV algorithm in the adversarial model. The proof was combinatorial, based on their notion of tradeoff-revealing LPs. Subsequently the connection to online Primal Dual algorithms was made in Buchbinder et al. (2007). The hardness of the offline problem without the small-bids assumption was proved in Chakrabarty and Goel (2010). The first online algorithm achieving a competitive ratio better than a half (0.5016) for the general problem without the small-bids assumption was provided in Huang et al. (2020). The stochastic random order model for AdWords was introduced in Goel and Mehta (2008) which showed that the Greedy algorithm achieves a competitive

ration of $1 - 1/e$. The first asymptotically optimal algorithm for the random order model was provided in Devanur and Hayes (2009), who also showed that we need to know the number of queries m to get such an algorithm. The Display Ads problem and the generalization to packing MILPs was introduced in Feldman et al. (2010); Agrawal et al. (2014). The idea of using the experts problem to solve packing LPs online was introduced in Gupta and Molinaro (2014). The algorithm and analysis in Sec. 5 are adapted from Agrawal and Devanur (2014), who generalized this to convex programming. Theorem 13 on the regret guarantee for MWU is from Arora et al. (2012). The automated machine learning of the algorithms was introduced in Kong et al. (2018); Zuzic et al. (2020). The formulation and results in Sec. 7 were provided in Aggarwal et al. (2019), and the results on the budgeted case in Devanur et al. (2020). The latter also studied the equilibrium dynamics in the budgeted case presented in Sec. 7.4. The PPAD-hardness of finding an equilibria in budgeted second price auctions is due to Chen et al. (2021). For a more comprehensive survey of models and results for ad allocations, see Mehta (2013).

Estimates of advertising spend in the introduction are taken from eMarketer (2020). The section on auto-bidding algorithms is motivated by performance-based autobidding products and strategies provided by most internet advertising companies, e.g., Google (2021).

References

Aggarwal, Gagan, Badanidiyuru, Ashwinkumar, and Mehta, Aranyak. 2019. Autobidding with Constraints. Pages 17–30 of: *International Conference on Web and Internet Economics*. Springer.

Agrawal, Shipra, and Devanur, Nikhil R. 2014. Fast algorithms for online stochastic convex programming. Pages 1405–1424 of: *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. SIAM.

Agrawal, Shipra, Wang, Zizhuo, and Ye, Yinyu. 2014. A dynamic near-optimal algorithm for online linear programming. *Operations Research*, **62**(4), 876–890.

Arora, Sanjeev, Hazan, Elad, and Kale, Satyen. 2012. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, **8**(1), 121–164.

Buchbinder, Niv, Jain, Kamal, and Naor, Joseph Seffi. 2007. Online primal-dual algorithms for maximizing ad-auctions revenue. Pages 253–264 of: *European Symposium on Algorithms*. Springer.

Chakrabarty, Deeparnab, and Goel, Gagan. 2010. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. *SIAM Journal on Computing*, **39**(6), 2189–2211.

Chen, Xi, Kroer, Christian, and Kumar, Rachitesh. 2021. The Complexity of Pacing for Second-Price Auctions. *CoRR*, **abs/2103.13969**.

Devanur, Nikhil, Lykouris, Thodoris, and Tardos, Eva. 2020. *Bid Shading Equilibria under Budget Constraints: Dynamics and the Price of Anarchy.*

Devanur, Nikhil R, and Hayes, Thomas P. 2009. The AdWords problem: online keyword matching with budgeted bidders under random permutations. Pages 71–78 of: *Proceedings of the 10th ACM conference on Electronic commerce.*

eMarketer. 2020. *Worldwide Digital Ad Spending.* `https://forecasts-na1.emarketer.com/5a4d1e53d8690c01349716b8/5a4d1bcfd8690c01349716b6`. Accessed: 2021-03-27.

Feldman, Jon, Henzinger, Monika, Korula, Nitish, Mirrokni, Vahab S, and Stein, Cliff. 2010. Online stochastic packing applied to display ad allocation. Pages 182–194 of: *European Symposium on Algorithms.* Springer.

Goel, Gagan, and Mehta, Aranyak. 2008. Online budgeted matching in random input models with applications to AdWords. Pages 982–991 of: *SODA*, vol. 8.

Google. 2021. *Auto-bidding products support page.* `https://support.google.com/google-ads/answer/2979071`. Accessed: 2021-03-27.

Gupta, Anupam, and Molinaro, Marco. 2014. How experts can solve LPs online. Pages 517–529 of: *European Symposium on Algorithms.* Springer.

Huang, Zhiyi, Zhang, Qiankun, and Zhang, Yuhao. 2020. AdWords in a Panorama. Pages 1416–1426 of: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020.* IEEE.

Kong, Weiwei, Liaw, Christopher, Mehta, Aranyak, and Sivakumar, D. 2018. A new dog learns old tricks: RL finds classic optimization algorithms. In: *International Conference on Learning Representations.*

Mehta, Aranyak. 2013. Online Matching and Ad Allocation. *Foundations and Trends® in Theoretical Computer Science*, **8**(4), 265–368.

Mehta, Aranyak, Saberi, Amin, Vazirani, Umesh, and Vazirani, Vijay. 2007. AdWords and generalized online matching. *Journal of the ACM (JACM)*, **54**(5), 22–es.

Zuzic, Goran, Wang, Di, Mehta, Aranyak, and Sivakumar, D. 2020. Learning Robust Algorithms for Online Allocation Problems Using Adversarial Training. *arXiv e-prints*, Oct., arXiv:2010.08418.