Foundations and Trends® in
Theoretical Computer Science
Vol. 8, No. 4 (2012) 265–368

DOI: 10.1561/0400000057

now
the essence of knowledge

Online Matching and Ad Allocation

By Aranyak Mehta

Contents

Foundations and Trends® in
Theoretical Computer Science
Vol. 8, No. 4 (2012) 265–368

DOI: 10.1561/0400000057

now
the essence of knowledge

Online Matching and Ad Allocation

Aranyak Mehta

Google Research, 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA, aranyak@google.com

Abstract

Matching is a classic problem with a rich history and a significant impact, both on the theory of algorithms and in practice. Recently there has been a surge of interest in the online version of matching and its generalizations, due to the important new application domain of Internet advertising. The theory of online matching and allocation has played a critical role in designing algorithms for ad allocation. This monograph surveys the key problems, models and algorithms from online matchings, as well as their implication in the practice of ad allocation. The goal is to provide a classification of the problems in this area, an introduction into the techniques used, a glimpse into the practical impact, and to provide direction in terms of open questions. Matching continues to find core applications in diverse domains, and the advent of massive online and streaming data emphasizes the future applicability of the algorithms and techniques surveyed here.

1

Introduction

A matching in a graph $G(V, E)$ is a set of edges $M \subseteq E$ such that for every $v \in V$, there is at most one edge in M incident on v. A maximum matching is a matching with the largest size. The problem of finding a maximum matching in a graph is a classic one, rich in history and central to algorithms and complexity. The elegance and complexity of the theory of matching is equally complemented by a rich set of important applications; indeed this problem arises whenever we need to connect any pairs of entities, for example, applicants to jobs, spouses to each other, goods to buyers, or organ donors to recipients.

In this monograph we will focus on the online version of the problem, in bipartite graphs. There has been considerable interest recently in online bipartite matching and its generalizations, driven by the important new applications of Ad Allocation in Internet Advertising, corresponding to matching ad impressions to ad slots. We will describe this motivating application first, before giving a brief overview of the history and foundations of matching.

1.1 Ad Allocation

Internet advertising constitutes perhaps the largest matching problem in the world, both in terms of dollars and number of items. Ads are sold either by auction or through contracts, and the resulting supply and demand constraints lead directly to the question of finding an optimal matching between the ad slots and the advertisers. The problem is inherently online, since we have to show an ad as soon as the request for an ad slot arrives, and we do not have complete information about the arriving ad slots in advance. Furthermore, offline optimization techniques are not even feasible due to the size of the problems, especially given the fact that dealing effectively with the long tail of ad requests is of critical business importance.

The problem of online matching and allocation has generated a lot of interest in the algorithms community, with the introduction of a large number of new problems, models and algorithmic techniques. This is not only due to the importance of the motivation but also due to the new and elegant questions and techniques that emerge. The first objective of this monograph is to provide a systematic survey of this literature.

This theoretical work has had an influential effect on the algorithmic framework used by virtually all of the companies which are in the Internet advertising space. The major contribution has been the introduction of the technique of *bid-scaling*. In this technique we scale the relevant parameter, for example, the bid, by a scaling function, and then choose that edge to match which has the highest scaled bid. This is to be compared to the greedy strategy which simply chooses the edge with the highest bid. The design of optimal algorithms in the online model has also led to the formulation of bid-scaling heuristics. Section 9 provides a brief survey of applications of these algorithms and heuristics, including the domain specific details. Let us quickly note that in the practical problem, there are typically three players in the market: the users of the service, the platform (for example, the search engine), and the advertisers. Thus there are three objective functions to consider: the quality of the ads shown, the revenue to the platform and the return on investment to the advertisers. We will consider these in more

detail later, but for most of the survey we will focus on maximizing the efficiency (the total size or weight) of the matching, which can be a good proxy for all relevant objective functions. A second point to note is that different advertising platforms have their own specific settings, for example, second-price auctions vs first-price, single slot vs position auctions, contracts vs auctions, etc. We will abstract these details out for the most part, and mention how they can be modeled, in Section 9.

1.2 Background on Matching: Applications, History and Offline Algorithms

The problem of matching is relevant to a wide variety of important application domains, besides our motivating application of ad allocation. In Economics, matching is relevant whenever there is a two-sided market (see, for example, [86]). One important formulation is the problem of finding a *stable matching* or a *Pareto efficient* matching in a graph [48]. This has found several important applications in the real world: it is used in matching of residents to hospitals (starting with [85]), students to high schools [1], and even kidney donors to recipients (see Kidney Exchanges [84]); Roth and Shapley were awarded the 2012 Nobel Prize in Economics for their influential and impactful work on this topic. Matching, with its generalizations, pervades Computer Science as a core algorithmic problem. For example, in Networking, an important problem is that of finding a good switch scheduling algorithm in input queued (IQ) switch architectures (see [76], among others). This reduces to that of finding a maximum matching to match input ports of a switch to its output ports at every time step. As another example, matching is core to resource allocation problems of various types from the scheduling and Operations Research literature, for example, allocating jobs to machines in cloud computing. Recently, the online matching algorithms from this survey have found applications [54] in crowdsourcing markets.

Besides its high applicability, matching is a central problem in the development of the field of algorithms, and indeed of Theoretical CS. We briefly overview this history next; the rest of this section can be skipped by readers with a strong background in classic matching theory.

The basic algorithms rely on the definition of *augmenting paths*: given a matching M in the graph, an augmenting path is an odd-length (simple) path with its edges alternating between being in M and not, and with the two end edges not in M. Berge's Theorem [17] states that:

Theorem 1.1 (Berge). A matching M is maximum iff it does not admit an augmenting path.

If a matching M admits an augmenting path P, then M can be *augmented* by flipping the membership of the edges of P in and not in M. This transforms M into a matching M' whose size is one more than that of M. An algorithm can proceed in this manner, by starting with any matching, and iteratively finding an augmenting path, and augmenting the matching.

This approach relies on being able to find augmenting paths efficiently. This is possible in bipartite graphs: one can find augmenting paths in bipartite graphs in time $O(|E|)$, by constructing breadth-first search trees (with alternating levels) from unmatched vertices. On bipartite graphs, the problem also has a close relationship with the maximum flow problem; one can reduce unweighted bipartite matching to a max-flow problem by adding a source and a sink to the graph appropriately. The fastest algorithms for this problem [39, 55] run in $O(\sqrt{|V|}|E|)$ time.

The question of finding a maximum matching in general (non-bipartite) graphs is a lot more difficult. Edmonds [42] presented the *Blossom* algorithm to compute a maximum matching in a general graph in polynomial time. The difficulty in general graphs comes precisely due to the presence of odd cycles. The algorithm proceeds by identifying structures, called *blossoms*, with respect to the current matching. A blossom consists of an odd cycle of, say, $2k + 1$ edges, of which exactly k edges belong to the matching, such that there further exists an even length alternating path, called the *stem*, starting with a matched edge at a vertex of the cycle. The algorithm starts with any matching and searches for an augmenting path, which can immediately augment the matching. If it finds a blossom instead, it contracts the blossom into

a single vertex and proceeds recursively. If it finds an augmenting path with vertices corresponding to contracted blossoms, then it expands the blossoms (recursively) finding a real augmenting path in the original graph. The running time of this algorithm, with appropriate data structures, is $O(|V|^2|E|)$. The fastest algorithm for matching in general graphs, due to Micali and Vazirani [91], runs in $O(\sqrt{|V|}|E|)$ time.

Let us also quickly note a property of *maximal* matchings, defined as those which cannot be improved upon by only adding more edges.

Theorem 1.2. If M is a maximal matching, and M^* a maximum matching, then $|M| \geq \frac{1}{2}|M^*|$.

This is fairly easy to see: since M is maximal, none of the edges in M^* can be added to it while keeping it a matching. Hence, every edge in M^* uniquely shares an end-point with an edge in M. Thus the number of vertices in M is at least the number of edges in M^*, giving the result. We will generalize this theorem later, to give a bound for greedy online algorithms for all the generalizations of matching that we will study.

In the problem of edge-weighted matching, the edges of G have weights, and the goal is to find a matching with the highest sum of weights of the edges in the matching (in the bipartite case, this is known as the *Assignment Problem*). The algorithm for the edge-weighted bipartite version is more complex than the unweighted problem. It works by updating the matching solution simultaneously with a set of weights on the vertices. This is known as the Hungarian Algorithm [68] (due to Kuhn, based on the work of König and Egerváry), and it is possibly the first example of a primal–dual update algorithm for Linear Programming (here the LP is to maximize the total weight of the matching over the polytope of all fractional matchings, and the weights on the vertices that the algorithm uses correspond to the dual variables of the LP). One observation to make is that all these algorithms are highly offline, that is, not easily adapted to the online setting, a point we will return to in the next section.

As is well-known, there was no fixed formulation of an efficient algorithm at the time that the Blossom algorithm was invented. The Blossom algorithm directly led to the formalization of polynomial time

as the correct definition. The impact of this definition is obviously immense to the fields of algorithms, complexity and Computer Science in general, essentially giving us the definition of the complexity class P. Furthermore, the definition of the class $\#P$ is also closely related to matching theory, as Valiant [90] proved that finding the number of perfect matchings in a graph (equivalently, the Permanent of a matrix) is NP-hard, and in fact complete for $\#P$. Matching is also a canonical problem for the study of randomized parallel algorithms and the class RNC; Karp et al. [62], and Mulmuley et al. [82] gave RNC algorithms for finding a maximum matching. The history and algorithms for offline matching have been excellently documented, for example, in the book [71] by Lovász and Plummer.

We will also study the online versions of several generalizations of the basic bipartite matching problem. Most of these are special cases of the Linear Programming problem, which has a vast literature of its own (see, for example, [30]). The classification of these problems and the LP formulations for the offline versions are described in Section 2.

1.3 Online Input

In this monograph we will focus on the online version of the bipartite matching problem and its generalizations. The area of online algorithms and competitive analysis has been very useful in abstracting and studying problems in which the input is not known in advance but is revealed incrementally, even as the algorithm makes its own decisions (see the book by Borodin and El-Yaniv [19]). This is precisely the situation in our motivating applications in which ad slots arrive online, and have to be allocated ads upon arrival, with zero, partial, or stochastic knowledge of the ad slots yet to arrive. We will model our applications via different problems and online input models. In the simplest version of the problem (online bipartite matching), there is a bipartite graph $G(U,V,E)$, in which U is known to the algorithm, vertices in V are unknown, but arrive one at a time, revealing the edges incident on them as they arrive. The algorithm has to match (or forgo) a vertex as soon as it arrives. Furthermore, all matches made are irrevocable;

this is to capture the fact that the arriving vertex v corresponds to an ad-slot on a web page viewed by a user.

Note that all the offline algorithms described in Section 1.2 are "highly offline". They typically involve initialization with some arbitrary matching and subsequent iterative improvements, via augmenting paths or guidance from dual variables. Thus they are not applicable to the online problem where the matches have to be made incrementally as vertices arrive, and are irrevocable. As we will see, the online algorithms work very differently, and often can provide only an approximate solution, that is, with a competitive ratio less than 1.

While our motivation for the online problem comes from ad allocation, large matching questions are becoming more prevalent. Often, the problem is online in nature, for example, the matching of arriving tasks to workers in crowdsourcing applications. Even in applications which are not strictly online, we often face problems with massive data, for example, in a streaming setting. Again, the offline algorithms are not applicable, and we need fast, simple, possibly approximate solutions, for example, in a streaming setting, rather than complex optimal algorithms. We expect that the algorithms surveyed here, or further variants, will be found to be useful in future applications.

Section 2 provides a classification of the different problems and models. Sections 3–8 treats the different problems in detail, giving the different algorithmic techniques. Section 9 describes the application setting and the algorithms and heuristics based on the theoretical results. We will provide open questions throughout the survey, and conclude in Section 10 with a list of additional open problems and future directions.

2

Classification: Problems and Models

This section provides an overview of the various problems and input models that we will study in this survey. The basic online matching problem is that of Online Bipartite Matching, which was first studied by Karp et al. [63]. In this problem, there is a bipartite graph $G(U, V, E)$ where one side U is known to us in advance and the other V arrives online, one vertex at a time. When a vertex $v \in V$ arrives, its neighbors in U are revealed. The arriving vertex needs to be matched to an available neighbor (if any). A match once made cannot be revoked. The objective is to maximize the size of the matching we obtain at the end of the arrival sequence. The difficulty in this problem arises from its online nature: the algorithm has to make a choice for the arriving vertex v without knowing the rest of the input graph. Choosing to match v to u instead of u' may result in a regret later on, if u turns out to be the only possible choice for some later vertex v'.

2.1 The Landscape of Problems

In recent years there have been many generalizations of this basic problem that have been studied. To help get a handle on this increasing

literature, we first provide a classification of the five main problems in this area. Note that each of these problems has been studied in different input models of vertex arrival; we will introduce these in the next section. There have been a few other problems beyond these main ones; we will mention them when relevant, and also give a short summary of these related problems in Section 10.

The first problem is online bipartite matching, as described above. This is the first problem to be studied, introduced by Karp et al. [63] in 1990.

(1) **Online bipartite matching:** There is a graph $G(U, V, E)$. U is known in advance and V arrives online. The goal is to maximize the size of the matching.

The simplest generalization of this problem is when the vertices in U have weights. This was introduced by Aggarwal et al. [5] in 2011, chronologically the last problem among these to be defined.

(2) **Online vertex-weighted bipartite matching:** A generalization of (1), in which each vertex $u \in U$ has a non-negative weight w_u, and the goal is to maximize the sum of weight of vertices in U which are matched.

Clearly, setting $w_u = 1,\ \forall u \in U$ gives us the unweighted version as a special case. Next, the motivating application of ad allocation was first captured by Mehta et al. [78] in the following problem, called Adwords. This was the first generalization of online bipartite matching introduced in the literature, 15 years after [63].

(3) **Adwords:** Each vertex $u \in U$ has a budget B_u, and edges $(u, v) \in E$ have bids bid_{uv}. When we match an arriving vertex v to a neighbor u, then u depletes bid_{uv} amount of its budget. When a vertex depletes its entire budget, then it becomes unavailable. The goal is to maximize the total budget spent. Let us identify an important special case:

- **The small bids assumption:** For each u, v, bid_{uv} is very small compared to B_u.

We will not quantify the smallness of the bids in the assumption just yet, but formalize it later. Online bipartite matching is a special case of Adwords in which all budgets are 1, and all edges have bids equal to 1. In fact, online vertex-weighted bipartite matching is a special case of Adwords in which $B_u = w_u$ for every u, and $\text{bid}_{uv} = w_u$ for each $(u, v) \in E$. However, online vertex-weighted matching and online bipartite matching are not special cases of Adwords once we make the small-bids assumption.

Next, we have the edge-weighted version of the matching problem, which is motivated by the display ads application.

(4) **Display Ads:** Each vertex $u \in U$ has an integral capacity c_u, which is an upper bound on how many vertices $v \in V$ can be matched to u. Each edges $(u, v) \in E$ has a weight w_{uv}. The goal is to maximize the total weight of edges matched. Again, we can identify an important special case:

- **The large capacity assumption:** For each u, c_u is a large number.

Again, we will not quantify the largeness of the capacities in the assumption here, but leave it for later. Also, in some versions, this problem needs the assumption that edges already matched can be removed in the favor of better edges, known as free-disposal. This problem and the notion of free-disposal was introduced by Feldman et al. [46]. Note that online bipartite matching is the special case of Display Ads in which all the c_u are 1 and all the edges have weight 1. Online vertex-weighted bipartite matching is a special case of Display Ads, with $c_u = 1, \forall\ u$ and $w_{uv} = w_u, \forall\ (u, v) \in E$. Adwords and Display Ads are unrelated problems, since Adwords has a constraint on the budget, while Display Ads has a constraint on the capacity.

A generalization of all the problems above is the following:

(5) **Online Submodular Welfare Maximization:** Vertex $u \in U$ has a non-negative monotone submodular valuation function[1] f_u: $2^V \to \mathbb{R}^+$. The goal is to maximize the sum of values

[1] A set function is called non-negative if $f(S) \geq 0\ \ \forall S \subseteq V$. It is called monotone if $f(S) \leq f(T),\ \ \forall S \subseteq T \subseteq V$. It is called submodular if for any two sets X and

of the allocation. More precisely, the online allocation produces a partition of V, where V_u is the set of vertices allocated to u. The goal is to maximize $\sum_u f_u(V_u)$.

To see how this problem generalizes all the previous problems, we restate all of them as online allocation problems, with different valuation functions, as in (5) above: vertices in U are agents, and vertices in V are items. Each agent $u \in U$ has a *value function* f_u: $2^V \to \mathbb{R}$. Items arrive online; each item has to be allocated to one agent as soon as it arrives. The goal is to provide an allocation of V to U, that is, a partition of V, $\{V_u \mid u \in U\}$ which maximizes the total value $\sum_{u \in U} f_u(V_u)$. Table 2.1 gives the valuation functions $f_u(V_u)$ corresponding to different problems. The valuation function for Display Ads assumes the free-disposal model mentioned earlier; we will define this in Section 7. It can be verified that all the functions in the table are submodular, making this problem a common generalization.

Note that some problem are named without the prefix "Online"; this is purely due to historical reasons, indeed all problems are in the online model, and we may drop the prefix if the context is clear.

One important problem which generalizes (1)–(4) and is a special case of (5) is the following:

Generalized assignment problem (GAP): This problem is a generalization of all the problems above except submodular welfare

Table 2.1. Utility functions corresponding to the different problems.

Problem	Valuation function
Bipartite matching	$\min\{1, \lvert V_u \rvert\}$
Vertex-weighted bipartite matching	$w_u \min\{1, \lvert V_u \rvert\}$
Adwords	$\min\{B_u, \sum_{v \in V_u} \text{bid}_{uv}\}$
Display Ads with free-disposal	$\max_{S \subseteq V_u,\ \lvert S \rvert \le c_u} \sum_{v \in S} w_{uv}$
Submodular welfare	Any monotone non-negative submodular function

$Y : f(X \cup Y) + f(X \cap Y) \le f(X) + f(Y)$. This is also equivalent to the following: $\forall\ S \subseteq T \subseteq V,\ x \notin T :\ f(S + x) - f(S) \ge f(T + x) - f(T)$.

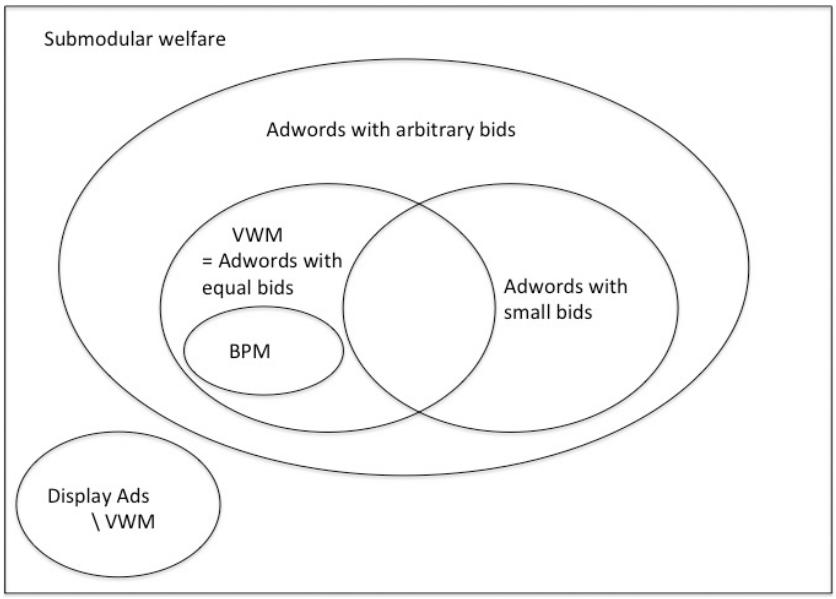

Fig. 2.1 Landscape of problems.

maximization. In this problem, vertices $u \in U$ have budgets B_u and edges have costs as well as weights. The goal is to maximize the total weight of the allocated edges subject to the constraint that the sum of costs of the edges allocated to every $u \in U$ is at most its budget.

Adwords is the special case of GAP in which the cost of an edge is equal to its weight. Display Ads is the special case in which the cost of every edge is 1 and the budget is the capacity. We will not study the GAP problem in as much detail as the others, but highlight the cases in which algorithms for the special cases generalize to GAP.

The relationships between these problems are shown in Figure 2.1.

2.2 Input Models and Competitive Ratio

The online allocation problems described above have been studied in different online input models. These differ from each other in how much information the algorithm has about the arriving vertices in V. We call the set of vertices in V, ordered by arrival time, the *query sequence*

(it is sometimes also called the *request sequence*). Let us start with the simplest model:

(1) **Adversarial order:** In this model we assume no knowledge of the query sequence, that is, no knowledge of V, E, and the arrival order of V. The algorithm begins with only the knowledge of U, and, at any point in time, it knows only the vertices in V which have already arrived, and the edges incident on them.

For the adversarial order, we may imagine an adversary, who knows the code of the algorithm, and can generate the worst possible graph and input order to make the algorithm perform as badly as possible. In the case of randomized algorithms, we will deal with a *non-adaptive adversary*, that is, the adversary will not have access to the outcomes of the random choices made by the algorithm (equivalently, of the random coins of the algorithm), but will have to fix the input graph in advance.

The adversarial order is a good model if the traffic (that is, the query sequence) is completely unpredictable. It also has the advantage that the algorithm is robust to any change in traffic from day to day, precisely because it does not use any estimates. However, in real problems, one may have some reasonably good estimate of the traffic. To capture such scenarios, we introduce **stochastic input models**, which assume less adversarial input. The objective in defining these models is to model reality better, while at the same time be able to provide quantifiable guarantees on the performance. We list these input models below, in sequence of the progressively stronger assumptions they make on the information available to the algorithm.

(2) **Random order:** Here we assume that although the input $G(U,V,E)$ is adversarial, the query sequence V arrives in a random order. Thus, the adversary can choose the worst graph (after seeing the code of the algorithm), but not the arrival order of V; vertices of V arrive in a uniform random permutation.

(3) **Unknown IID:** In this model we start with the abstraction of a *vertex type*, which encodes the set of neighbors in U,

and bids or weights on the incident edges (as relevant to the problem). There is a collection $\mathcal{T}$ of types of vertices, and a distribution $\mathcal{D}$ on $\mathcal{T}$. The query sequence is picked in the following manner: at each time a type $t \in \mathcal{T}$ is drawn from $\mathcal{D}$, IID, and a vertex $v \in V$ of that type t is instantiated, and arrives at that time. $\mathcal{D}$ is not known to the algorithm before-hand.

(4) **Known IID:** This is identical to the Unknown IID model, except that $\mathcal{D}$ is provided to the algorithm in advance.

Note that the input models are ordered in order of how much information about the arriving query sequence is available to the algorithm in advance. We formalize this next, by describing how we quantify the performance of an online algorithm on each of these models.

2.2.1 Competitive Ratio

We will use the standard notion of **competitive ratio** to measure the performance of our online algorithms. For the adversarial model, the competitive ratio is defined by the ratio of the value of the objective function attained by the algorithm ALG, to OPT, defined as the maximum objective value attained offline, that is, given the entire graph $G(U,V,E)$, and without any computational constraints. The competitive ratio is at least α if, for every graph $G(U,V,E)$ and every input order of V, $\frac{\text{ALG}(G)}{\text{OPT}(G)} \geq \alpha$. Formally, (using C.R. for the competitive ratio):

$$\text{C.R.} = \min_{G(U,V,E),\ \text{order of } V} \frac{\text{ALG}(G)}{\text{OPT}(G)}$$

If the algorithm is randomized, then we take the expected value of the objective function in the numerator.

$$\text{C.R.} = \min_{G(U,V,E),\ \text{order of } V} \frac{\text{E}[\text{ALG}(G)]}{\text{OPT}(G)}$$

where the expectation is over the random coin flips of the algorithm. Note that the standard definition of competitive ratio allows for the additional loss of an additive constant, so an algorithm is called

c-competitive if $\mathrm{ALG}(G) \geq c\ \mathrm{OPT}(G) - b$, for some constant b. This makes a difference only when OPT itself is a constant, and we do not include it in our definitions.

For the stochastic models the competitive ratio is computed in expectation over the randomness in the input. For the Random Order model, this is a simple change in the definition:

$$\text{C.R.} = \min_{G(U,V,E)} \frac{\mathrm{E}[\mathrm{ALG}(G)]}{\mathrm{OPT}(G)}$$

where the expectation of ALG is over the random arrival order of V. If the algorithm is itself randomized, then the expectation is also taken over its random coin flips. Note that the denominator does not have an expectation, since, given G, the value of OPT is independent of the arrival order.

For the known and unknown IID models, G is itself random (since V is constructed online by drawing from the distribution on types), so OPT is itself a random quantity. Then the definition becomes

$$\text{C.R.} = \min_{\mathcal{D}} \frac{E\left[\mathrm{ALG}(G)\right]}{E\left[\mathrm{OPT}(G)\right]}$$

where the minimum is over the input instances, which in this case is the distribution on types. The expectations are over the instantiations of $G(U,V,E)$ as V and E are drawn IID from $\mathcal{D}$ (and a further expectation in the numerator over the randomness of the algorithm, if it is randomized). Note how the value of OPT itself depends on the random process which instantiates the graph.

In the IID models, we may define the competitive ratio a little differently:

$$\text{C.R.} = \min_{\mathcal{D}} E\left[\frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)}\right]$$

which is the expectation over the random graphs of the ratio achieved by the algorithm and the optimum for that graph. A different, stronger definition is to achieve a good ratio on almost every instantiation: the C.R. is at least α if for all distributions $\mathcal{D}$, $\frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)} \geq \alpha$, with high probability over the instantiations of $G(U,V,E)$, as drawn from $\mathcal{D}$. We

will mostly restrict to the definition which uses the ratio of the expected values, and we will explicitly mention if a stronger definition is used.

We note that as we go down this sequence of models, the information available to the algorithm increases, and the task of the algorithm designer becomes simpler: consider an algorithm which can run in any input model. If we denote the competitive ratios it achieves in the four models as C.R.(Adv), etc., then we have:

Theorem 2.1.

$$\text{C.R.(Adv)} \leq \text{C.R.(RO)} \leq \text{C.R.(K-IID)} \leq \text{C.R.(UnK-IID)}$$

That is, an algorithm that achieves a ratio of α in a stronger model, achieves a ratio of at least α in a weaker model. The first inequality is obvious, since the algorithm achieves at least C.R.(Adv) for any input order of a given graph, hence it does so even in expectation over a random order. The last inequality is obvious since the algorithm can ignore the knowledge it has about the distribution in the Known IID model. The only non-trivial step in the proof of this is the one from the Random Order to the Unknown IID model: for this, first note that the events in the probability space of the latter represent the instantiation of the graph and the arrival order. Due to the IID nature of the process, each arrival order for a given instantiation G has the same probability. We can therefore partition this space into regions, each with $|V|!$ events, where each region corresponds to events with the same graph G and all possible arrival orders, with conditional distribution over the region being uniform. An algorithm which gives a ratio of α in the Random Order model, when run on an input which comes from the Unknown IID model, will achieve a ratio of α conditional on each region, and therefore globally.

Table 2.2 gives a preview summary of the main results we will describe in this survey. We note that despite their apparent differences there is no known separation result differentiating the Unknown IID model from the Random Order model. Thus, they share a single column in Table 2.2. There are a few other input models in the literature, but we will not study them in detail here. We mention two such models

Table 2.2. Summary of results for online matching problems in different arrival models. Rows correspond to problems and columns to arrival models. The entries are the best known ratios, and the corresponding upper bounds (in parentheses). Question marks correspond to open questions where no bound is known besides that which follows from some other input model or problem. Citations are provided in the text. Recall that $1 - \frac{1}{e} \simeq 0.63$.

	Adversarial Order	Random Order/ unknown IID	Known IID
Bipartite matching	$1 - \frac{1}{e}$ (optimal)	0.696 (?)	0.702 (0.823)
Vertex-weighted bipartite matching	$1 - \frac{1}{e}$ (optimal)	$1 - \frac{1}{e}$ (?)	$1 - \frac{1}{e}$ (?)
Adwords (small bids)	$1 - \frac{1}{e}$ (optimal)	$1 - \epsilon$ (optimal)	$1 - \epsilon$ (optimal)
Adwords (general bids)	$\frac{1}{2}$ (?)	$1 - \frac{1}{e}$ (?)	$1 - \frac{1}{e}$ (?)
Display Ads with free-disposal (large capacities)	$1 - \frac{1}{e}$ (?)	$1 - \frac{1}{e}$ (IID) (?)	$1 - \frac{1}{e}$ (?)
Display Ads with free-disposal (general capacities)	$\frac{1}{2}$ (?)	$1 - \frac{1}{e}$ (IID) (?)	$1 - \frac{1}{e}$ (?)
Display Ads no free-disposal (general capacities)	0 (0)	$\frac{1}{e}$ (optimal)	? (?)
Submodular welfare	$\frac{1}{2}$ (optimal)	$1 - \frac{1}{e}$ (IID) (?)	$1 - \frac{1}{e}$ (?)

(the non-identical distributions model, and the distribution-of-metrics model) in Sections 9 and 10.

2.3 Offline Versions

We briefly mention the complexity of the offline versions of the problems we study here; recall that, in the offline versions, the entire

Matching

$$Max \quad \sum x_{uv}$$
$$\sum_v x_{uv} \leq 1 \quad \forall u$$
$$\sum_u x_{uv} \leq 1 \quad \forall v$$
$$x_{uv} \in \{0,1\}$$

Weighted-Vertex Matching

$$Max \quad \sum w_u x_{uv}$$
$$\sum_v x_{uv} \leq 1 \quad \forall u$$
$$\sum_u x_{uv} \leq 1 \quad \forall v$$
$$x_{uv} \in \{0,1\}$$

Adwords

$$Max \quad \sum bid_{uv} x_{uv}$$
$$\sum_v bid_{uv} x_{uv} \leq B_u \quad \forall u$$
$$\sum_u x_{uv} \leq 1 \quad \forall v$$
$$x_{uv} \in \{0,1\}$$

Display Ads

$$Max \quad \sum w_{uv} x_{uv}$$
$$\sum_v x_{uv} \leq c_u \quad \forall u$$
$$\sum_u x_{uv} \leq 1 \quad \forall v$$
$$x_{uv} \in \{0,1\}$$

Fig. 2.2 IPs for the offline versions.

graph is known in advance. The offline versions of the bipartite matching, vertex-weighted bipartite matching, Adwords and Display Ads problems can be written as integer programs. Figure 2.2 shows the IPs for the different problems.

Besides the Adwords problem, the other three problems can be solved exactly in polynomial time, in fact, via combinatorial algorithms (see Section 1.2). The Adwords problem with the small bids assumption can be solved with a $(1 - 4\epsilon)$ approximation if the bid to budget ratio is at most ϵ [23].

The offline Adwords problem without the small bids assumption (also known as the Maximum Budgeted Allocation problem) is NP-hard to approximate to a factor better than 15/16 [23], and the best known offline algorithm is 3/4 [23, 89]. A sequence of previous results for this problem gave increasingly improved approximation factors: the first known was by Garg et al. [49], which had a factor of $\frac{2}{1+\sqrt{5}} \simeq 0.618$. Andelman and Mansour [8] gave an algorithm with an approximation

factor of $1 - \frac{1}{e}$ (and 0.717 when all budgets are equal). Azar et al. [14] improved this to 2/3, which first showed that the offline version is strictly easier than the online version (for which $1 - \frac{1}{e}$ was known to be an upper bound even for the special case of bipartite matching [63]).

The Generalized Assignment Problem (GAP) is a generalization of all these problems, and is captured by the following IP (where w_{uv} is the weight of the edge (u, v), and c_{uv} is its cost):

GAP

$$\text{Max} \sum w_{uv} x_{uv}$$

$$\sum_v c_{uv} x_{uv} \leq B_u, \quad \forall u$$

$$\sum_u x_{uv} \leq 1, \quad \forall v$$

$$x_{uv} \in \{0, 1\}$$

For GAP, a $\frac{1}{2}$ approximation follows from Shmoys and Tardos [88] (see [27]). This was improved to a $1 - \frac{1}{e}$ factor by Fleischer et al. [47], and by Feige and Vondrak [43] to $1 - \frac{1}{e} + \epsilon$ for a very small constant $\epsilon < 10^{-5}$. The best known hardness for GAP is $\frac{10}{11}$ due to Chakrabarty and Goel [23].

The Submodular Welfare Maximization problem is NP-hard to approximate to better than $1 - \frac{1}{e}$ [65]. An optimal algorithm in the value oracle model was provided by Vondrak in [93] with an approximation factor of $1 - \frac{1}{e}$. Mirrokni et al. [80] prove that beating $1 - \frac{1}{e}$ in the value oracle model needs exponential communication in the value oracle model.

3

Online Bipartite Matching

Online bipartite matching was introduced by Karp et al. [63]. It is interesting to note how a problem first studied purely out of academic interest has led to variants which have found core practical application a decade and a half later, in a domain which did not exist at the time. In this problem, there is a bipartite graph $G(U, V, E)$ where one side U is known to us in advance and the other side V arrives online, one vertex at a time. When a vertex $v \in V$ arrives, its neighbors in U are revealed. The arriving vertex can be matched to some available neighbor (if any). A match once made cannot be revoked. The objective is to maximize the size of the matching.

3.1 Adversarial Order

We first study this question in the adversarial model. This section thus covers the simplest of the problems in this survey in the classic online model.

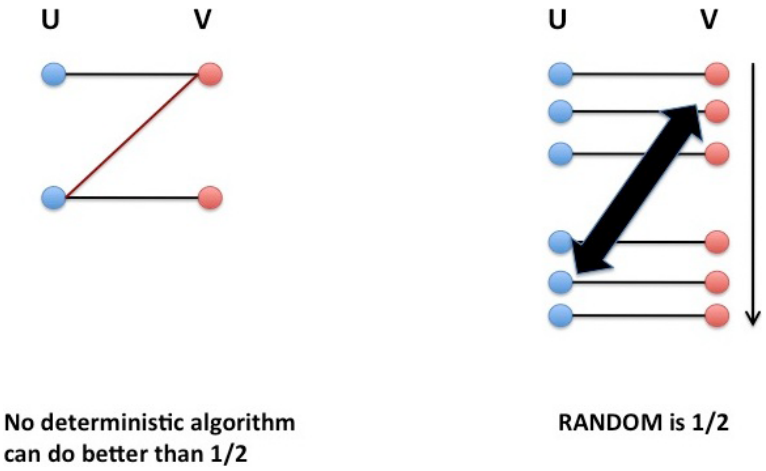

Fig. 3.1 The core difficulty for online bipartite matching.

3.1.1 Two Suboptimal Algorithms

We first describe two schemes which are natural first attempts at solving this problem. Their performance is not optimal, but provide us a baseline to improve upon.

Deterministic Algorithms and Greedy. The core difficulty in the problem is captured by a simple example (see the left side of Figure 3.1). In this example, $U = \{u_1, u_2\}$, and the first vertex v_1 to arrive has edges to both u_1 and u_2. At this point, u_1 and u_2 are indistinguishable. Suppose we match v_1 to u_2. This choice turns out to be sub-optimal since the next vertex to arrive, v_2, has only u_2 as a neighbor, and we would end up not being able to match v_2. The optimal matching, in hindsight, is to match v_1 to u_1 and v_2 to u_2. In fact this example shows that no deterministic online algorithm can achieve a ratio better than $\frac{1}{2}$: connect v_2 to precisely that vertex to which the algorithm matched v_1. The following algorithm, called Greedy, will appear in different variants throughout this survey.

Algorithm 1: Greedy

When the next vertex $v \in V$ arrives:
Match v to any available neighbor (if any).

Note that GREEDY will never leave an edge with both end points unmatched. Thus it constructs a maximal matching, and by Theorem 1.2, it achieves a ratio of $\frac{1}{2}$. Thus we have:

Theorem 3.1. No deterministic algorithm can achieve a ratio of better than $\frac{1}{2}$. GREEDY achieves a ratio of $\frac{1}{2}$.

RANDOM, a simple randomized algorithm. One might be tempted to think that a simple application of randomization will overcome this obstacle faced by deterministic algorithms. Consider the algorithm RANDOM, which lets each arriving vertex pick one of its available neighbors uniformly at random.

Algorithm 2: RANDOM

When the next vertex $v \in V$ arrives:
Match v to a neighbor picked uniformly at random from the set of available neighbors (if any).

RANDOM achieves a ratio of 3/4 in the example of Figure 3.1 (left). Note that it also constructs a maximal matching, in fact, in every outcome of its coin tosses. Thus its ratio is no worse than $\frac{1}{2}$, and one may hope for a better ratio in expectation over its randomization. However, this strategy also does not do any better than $\frac{1}{2}$ in general, as the graph in Figure 3.1 (right) shows. This is a "blown-up" version of the simple 2×2 example on the left. Each side of the bipartition has n vertices divided into two parts of size $n/2$ each ($U = U_1 \cup U_2$ and $V = V_1 \cup V_2$) There is a perfect matching between U and V (the ith vertex in U and V have an edge between them). There is also a bipartite clique between V_1 and U_2. It can be shown that RANDOM achieves a ratio of $1/2 + o(1)$ for this instance, essentially because almost all the vertices in V_1 (which arrive first) match to some vertex in U_2 (since most of their neighbors are in U_2), and so when the vertices in V_2 arrive later, do not have any available neighbors.

Theorem 3.2. RANDOM achieves a ratio of $\frac{1}{2}$. This analysis is tight.

3.1.2 RANKING: An Optimal Algorithm

The tight example for RANDOM is a bit worrying, since it seems to suggest that, to beat $\frac{1}{2}$, the algorithm has to get the correct neighbor for a constant fraction of the vertices in V_1, out of a set of $\frac{n}{2}$ neighbors. Next we present an algorithm that is able to overcome this difficulty and significantly improve the competitive ratio, by using *correlated randomness*. This algorithm, called RANKING, was introduced by Karp et al. [63] and it achieves an optimal ratio of $1 - \frac{1}{e} \simeq 0.63$. In this algorithm we begin by permuting the vertices on the left side in a random permutation π. We match each incoming vertex in V to that available neighbor U which is the highest according to π (that is, with the smallest value of $\pi(u)$). The vertex remains unmatched if none of its neighbors are available.

Algorithm 3: RANKING (KVV)

Offline: Pick a permutation π of U uniformly at random
When the next vertex $v \in V$ arrives:
If v has no available neighbors, continue.
Match v to the neighbor u with the smallest value of $\pi(u)$.

A different way to describe the algorithm is that we first pick a random priority number for each $u \in U$ IID (from some distribution, say uniform on $[0,1]$), and match the arriving vertex to the available neighbor with the highest priority. Note how the randomness in the algorithm is correlated; every vertex $v \in V$ uses the same permutation (or priority) over the $u \in U$, as compared to RANDOM, in which each vertex $v \in V$ picks its own random choice, independent of the other $v \in V$.

3.1.2.1 Analysis

We now analyze the performance of this algorithm. The ideas in the original proof in [63] have been distilled in a series of simplifications (starting with [51]), and we present here a proof similar (although not identical) to a proof by Birnbaum and Mathieu [18]. Later, in Section 6, we will see how RANKING can be analyzed using a very different technique, in the Primal–Dual framework.

Theorem 3.3([63]). RANKING achieves a ratio of $1 - \frac{1}{e}$ for the online bipartite matching problem in the adversarial arrival model.

Proof. For simplicity of exposition, we will assume that there is a perfect matching in G, so that $\text{OPT} = n$. We will fix one such optimal perfect matching, and refer to it throughout. Fix a permutation π of U and a vertex $u \in U$, and let $t = \pi(u)$. Consider the run of the algorithm on permutation π. One of the following two cases can occur: either u is matched to some vertex in V, or it is left unmatched. In the former case, we will call the event (π, u) a *Match event* at position t, and in the latter case we will call it a *Miss event* at position t. Note that the gain of the algorithm is the total probability of all Match events. Further, since π is chosen uniformly at random, we just need to count the fraction of events that are Match events.

A warm up: As a warm-up we first prove a simple bound of $\frac{1}{2}$. Consider a Miss event (π, u^*). Then we know that u^*'s partner in the optimal perfect matching (call it v^*) does get matched during the run on π (since u^* itself was available to be matched to v^* when it arrived). Let u' be the vertex to which v^* was matched. Thus, for every Miss event (π, u^*), there is a Match event (π, u'). That is, there are as many Match events as there are Miss events, so the competitive ratio is at least $\frac{1}{2}$. In fact, this is true even for an arbitrary choice of π, rather than random (which just corresponds to the GREEDY algorithm). We already have a little more information that we did not use so far: $\pi(u') < \pi(u^*)$, that is, v^* is matched to a vertex u' which is placed higher than u^* in π. Next, we use this observation and the randomness of the permutation to amplify the argument.

A map from Misses to Matches: Consider again, the Miss event (π, u^*), with $\pi(u^*) = t$. Let u' be the vertex to which v^* is matched. Now consider the n permutations $\{\pi^{(i)},\ i \in [n]\}$ produced by moving u^* to position i, keeping the relative order of all other vertices fixed (note that $\pi^{(t)}$ is the same as π). We claim that v^* continues to be matched in each of these permutations, to some vertex $u'' \in U$ with $\pi(u'') \leq t$.

To see this, note that when u^* moves lower in the permutation ($i > t$), then the run on $\pi^{(i)}$ is unchanged from the run on π until the time v^* arrives. So v^* continues to be matched to u'. Now consider a case in which u^* moves up in the permutation ($i < t$). Intuitively, things only improve for v^*, in that it will have a larger set of available neighbors when it arrives. One can formalize this intuition by following an alternating path argument starting at the match of u^* in $\pi^{(i)}$, if any, proving the claim that v^* is still matched to a vertex with position at most t.

This observation gives a 1-to-n map from a Miss event (π, u^*) to n Match events $(\pi^{(i)}, u_i)$, (for some $u_i \in U$), such that u_i is matched to v^*, and $\pi^{(i)}(u_i) \leq t$.

No double-counting: Now we fix $t \in [n]$ and restrict attention to Miss events (π, u) with $\pi(u) = t$. We claim that every Match event is mapped to at most once in this restricted map. To see this, suppose both the Miss events (π_1, u_1) and (π_2, u_2) (with $\pi_1(u_1) = \pi_2(u_2) = t$) map to a Match event $(\widehat{\pi}, \widehat{u})$. Let v^* be the vertex to which $\widehat{u}$ is matched in $(\widehat{\pi}, \widehat{u})$. Now, since our map is defined via the optimal partner of v^* (call it u^*), we know that $u_1 = u_2 = u^*$. And since the map only changes the position of u_1 in π_1 and u_2 in π_2 (from t to $1, \ldots, n$), leaving the order of the rest of the vertices unchanged, we infer that $\pi_1 = \pi_2$, thus proving the claim that each Match event is mapped to at most once in the restricted map. Thus we proved that for every Miss event at position t there are n unique Match events at some position less than or equal to t. This yields the following set of equations:

$$\forall\, t \in [n]\colon\ n \cdot \Pr[\text{Miss event at position } t] \leq \sum_{s \leq t} \Pr[\text{Match event at position } s]$$

Thus, by leveraging the randomness of the permutation, we have managed to amplify the map from Miss events to Match events, thereby limiting the total probability of Miss events.

Setting $x_t = \Pr[\text{Match event at position } t]$, we get the following set of inequalities:

$$\forall t\colon 1 - x_t \ \leq\ \frac{1}{n} \sum_{s \leq t} x_s$$

These inequalities define a polytope in the space of the x_t, such that no matter what the input graph and input order is, the vector of x_ts lies in the polytope. Minimizing the objective function $\sum_t x_t$ over this polytope, we get a lower bound on the performance of the algorithm on the worst possible input for the algorithm; this technique is known as a *Factor-Revealing LP* technique (introduced in [52, 57, 75]): ALG $= \sum_t x_t \geq \left(1 - \frac{1}{e}\right) n$. Since OPT $= n$, this proves the required ratio. □

In Section 4, we will provide a generalization and a different interpretation of this algorithm, based on random perturbations. In Section 6.3, we will provide a different proof for the same algorithm, based on the online Primal–Dual scheme.

A tight example. In [63] the authors also gave an example to show that their bound is essentially tight. They considered the graph shown in Figure 3.2 (the left side shows the adjacency matrix, and the right depicts the graph itself) and showed that if the vertices arrive from right to left in the adjacency matrix as shown, then no online algorithm can hope to achieve a ratio better than $1 - \frac{1}{e}$ for this problem. Intuitively, the first few vertices to arrive have lots of alternatives and most of them do not end up matching their partner in the optimal solution (which is the diagonal of the adjacency matrix). This ends up exhausting the options of a large fraction of the vertices arriving towards the end which are left unmatched.

Optimality. RANKING is optimal among all online algorithms. In [63] the authors gave a construction based on the tight example above

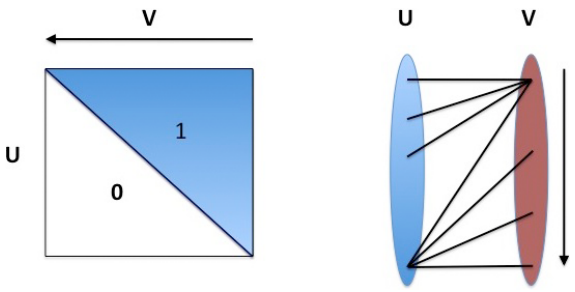

Fig. 3.2 Tight example for RANKING.

which showed, using Yao's lemma, that no randomized algorithm can achieve a ratio better than $1 - \frac{1}{e}$. This upper bound therefore holds for all the generalizations of the Online Bipartite Matching problem that we will study in the following chapters.

3.2 Random Order

Recall that in this model we assume that the graph $G(U, V, E)$ is adversarial, but we are given the guarantee that the vertices in V arrive online in a random order, that is, there is a uniformly random permutation π of V, and the vertex $\pi(t)$ arrives at time t. The model was introduced in this context by Goel and Mehta [51], who showed that the Greedy algorithm, which matches each incoming vertex to an arbitrary neighbor (breaking ties consistently across different input permutations), achieves a ratio of $1 - \frac{1}{e}$. In fact this result follows directly from the analysis of Ranking in the adversarial model, since it can be seen that Greedy with random order input simulates Ranking with adversarial input, with the roles of the two sides switched. In [51], the authors also provided upper bounds for algorithms in this model.

Theorem 3.4 ([51, 63]). Greedy achieves a ratio of $1 - \frac{1}{e}$ in the Random Order model. No deterministic algorithm can achieve a ratio better than 3/4 in this model, and no randomized algorithm better than 5/6.

Thus $1 - \frac{1}{e}$ is the baseline to beat in this model. Recall that this was in fact the upper bound in the adversarial model. An intriguing open question in [51] was whether it was possible to beat this bound in the Random Order model (and hence in the IID model) by a more intelligent algorithm, for example, Ranking. Interestingly, simulations showed that Ranking itself achieves a ratio very close to 1 for the tight example in the adversarial model (Figure 3.2). This question was answered in the affirmative by Karande et al. [61] and by Mahdian and Yan [73] where they showed that Ranking achieves a ratio strictly greater than $1 - \frac{1}{e}$ in the Random Order model. These results are summarized in the following theorem.

Theorem 3.5 ([61, 73]). RANKING achieves a ratio of at least 0.696 [73] (0.656 [61]) for the online bipartite matching problem in the Random Order model. If $G(U,V,E)$ has k disjoint perfect matchings then RANKING achieves a ratio of at least $1 - 1/\sqrt{k}$ [61].

The proof in [73] uses a computer aided technique to solve a family of *strongly factor revealing LPs*, a technique that was introduced in [73], and shown to be useful for other problems also. This LP is an augmented version of the one in the proof of Theorem 3.3. The technique in [61] is more combinatorial and builds on the proof of Theorem 3.3. Recall that that proof proceeded by finding a map from a Miss in some permutation π to Matches in related permutations $\pi^{(1)}, \ldots, \pi^{(n)}$, further limiting the total probability of possible Misses. The proof in [61] uses the random order of the input to further amplify this effect, by finding additional maps from Misses to Matches, and from a certain type of Matches to a different type of Matches. This further reduces the fraction of Misses that can exist for any input.

Intuitively, the last statement of the result states that if the given graph is internally robust with respect to perfect matchings (that is, removing a perfect matching still keeps OPT unchanged) then it is easy to find a large matching online. This explains the empirical observation that the performance of RANKING goes to 1 for the tight example for the Adversarial model shown in Figure 3.2 — note that this graph has a large number of nearly perfect matchings.

Open Question 1. Close the gap between upper and lower bounds in this model.

Open Question 2. Find examples of other problems in which making a robustness assumption on OPT gives significantly better performance.

In terms of tight examples for RANKING in this model, Karande et al. [61] provided two examples for which the algorithm achieves a

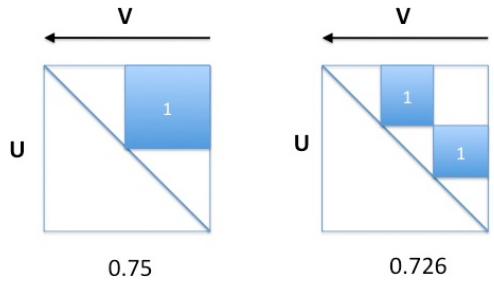

Fig. 3.3 Difficult examples for RANKING in the Random Order model. RANKING achieves 0.75 for the example on the left, and 0.726 for the example on the right.

ratio of 0.75 and 0.726 respectively. These graphs are shown in Figure 3.3. For an upper bound, a result by Manshadi et al. [74] (described in Section 3.3) shows that no online algorithm can achieve a competitive ratio better than 0.83 even in the known IID model. Since the known distribution model is strictly easier than the Random Order model, the upper bound carries over to this setting.

3.3 Known IID

This model was first introduced by Feldman et al. [46], to model the fact that often the algorithm knows the distribution of queries to arrive. In this model we assume that the algorithm knows U as well as a distribution on possible *types* of vertices in V, where a type encodes which are the neighbors in U. The arriving vertices in V are drawn independently from the distribution on types and have to be matched irrevocably upon arrival. More formally, we are given a *base graph* $\widehat{G}(U, \widehat{V}, \widehat{E})$ in advance (where $\widehat{V}$ is the set of types), and also a distribution $\mathcal{D}$ on $\widehat{V}$. We have $|U| = n$ and $|\widehat{V}| = m$, which is assumed to be polynomial in n. The outcome graph, $G(U,V,E)$, is obtained via the following process: at each time $i = 1,\ldots,n$, we pick the next type $\widehat{v_i} \in \widehat{V}$ according to $\mathcal{D}$, independently of the previous draws. The arriving vertex $v_i \in V$ is taken to be a copy of $\widehat{v_i}$, that is, with the same neighbors in U.

Intuitively, this model seems much easier for designing an algorithm than the adversarial or Random Order models, since we know the entire base graph in advance, and the arriving graph is a sample of it. In other words, we have an estimate of the graph beforehand. Let us begin by

considering a simple algorithm for this problem that is based on the idea of using the offline estimate to guide the online decisions. We will assume for simplicity that $m = n$, that there is a perfect matching in $\widehat{G}$ and that $\mathcal{D}$ is the uniform distribution on $\widehat{V}$. The algorithm Suggested Matching was introduced in [46]:

Algorithm 4: Suggested Matching (SM)

Offline Step: Find an optimal matching $\mathcal{M}$ in the base graph $\widehat{G}$
When the next vertex $v \in V$ **arrives:**
Let $\widehat{v}$ be the type of v.
If $u = \mathcal{M}(\widehat{v})$ is available, then match v to u.
Else continue.

The algorithm uses the perfect matching of the types to guide the online matching. An arriving vertex is given exactly one predetermined choice of neighbor to try to match to, namely, the perfect match of its type in the base graph. Thus this algorithm tries to build a matching as close as possible to the perfect matching in the base graph. This is also precisely the limitation of this algorithm: a type can repeat across different samples drawn from the distribution, that is, we may get several draws $v_1, v_2, \ldots, v_k \in V$ for the same base type $\widehat{v} \in \widehat{V}$. Since the algorithm tries to match each such v_i to the same vertex in U (namely, $\mathcal{M}(\widehat{v})$), this will result in all but the first arriving vertex, v_1, remaining unmatched. In fact using a standard *balls-in-bins* argument we can show that this algorithm achieves a ratio of $1 - \frac{1}{e}$, when D is the uniform distribution: consider the process in which there are n bins, corresponding to the types, and n balls, corresponding to the vertices in V. The process of generating G corresponds to throwing each ball into a random bin. We know that w.h.p. $1 - \frac{1}{e}$ bins get at least one ball at the end of this process. For each of these filled bins, Suggested Matching gets exactly one match, thus giving the following theorem.

Theorem 3.6 ([46]). Suggested Matching achieves a ratio of $1 - \frac{1}{e}$ in the known IID model, and this analysis is tight.

So, in this model again, the ratio of $1 - \frac{1}{e}$ is the baseline to beat. We will now discuss an algorithm that does better than $1 - \frac{1}{e}$ in this model. The algorithm, which we call the TWO SUGGESTED MATCHINGS (TSM) algorithm (introduced in [46]), is based on the idea of the *power of two choices*. Instead of focusing on one suggested matching in the base graph to guide in decision making, it uses two matchings for this purpose. Suppose we have two large disjoint matchings (say $\mathcal{M}_1$ and $\mathcal{M}_2$) in the base graph $\widehat{G}$. Let the next arriving vertex v be of type $\widehat{v}$. The TSM algorithm tries to match v to $\mathcal{M}_1(\widehat{v})$; if it fails (because $\mathcal{M}_1(\widehat{v})$ was already matched) then it attempts to match to $\mathcal{M}_2(\widehat{v})$. If both these attempts fail then v remains unmatched.

Algorithm 5: TWO SUGGESTED MATCHINGS (TSM)

Offline Step: Find two "large" matchings $\mathcal{M}_1$ and $\mathcal{M}_2$ in the base graph $\widehat{G}$.

When the next vertex $v \in V$ arrives:

Let $\widehat{v}$ be the type of v.
If $u_1 = \mathcal{M}_1(\widehat{v})$ is available, then match v to u_1.
Else If $u_2 = \mathcal{M}_2(\widehat{v})$ is available, then match v to u_2.
Else continue.

Thus each arriving vertex now has two attempts at finding a match, which could, in principle, lead to a better ratio than that of SUGGESTED MATCHING. However, the new difficulty introduced is that two vertices of different types $\widehat{v_1}, \widehat{v_2}$ now interact, since $\mathcal{M}_1(\widehat{v_1})$ and $\mathcal{M}_2(\widehat{v_2})$ may be the same vertex.

In fact, a weaker variant of this algorithm was defined in [46]: The first arrival of a vertex type will try only the neighbor in $\mathcal{M}_1$ (and drop out if that neighbor is already matched), and the second arrival will try the neighbor in $\mathcal{M}_2$. Thus the algorithm in [46] is *non-adaptive* and creates some loss as compared to the form defined above, since the first arrival of a vertex drops out even if the neighbor in $\mathcal{M}_2$ is available. However, this makes the algorithm easier to analyze (and the analysis holds for the adaptive version as well).

Algorithm 6: TSM (non-adaptive)

Offline Step: Find two "large" matchings $\mathcal{M}_1$ and $\mathcal{M}_2$ in the base graph $\widehat{G}$.

When the next vertex $v \in V$ arrives:

Let $\widehat{v}$ be the type of v.

If v is the first arrival of type $\widehat{v}$ and $u_1 = \mathcal{M}_1(\widehat{v})$ is available, then match v to u_1.

If v is the second arrival of type $\widehat{v}$ and $u_2 = \mathcal{M}_2(\widehat{v})$ is available, then match v to u_2.

Else continue.

We have not yet stated how the two matchings $\mathcal{M}_1$ and $\mathcal{M}_2$ are obtained. Depending on how this is done, we get several variants. The first variant was introduced in [46]: We first find a maximum flow in a derived flow graph derived from $\widehat{G}$ as follows. Connect a source to all vertices in U with capacity 2 on the edges, and connect a sink from all vertices in $\widehat{V}$, again with capacity 2. Direct all edges in $\widehat{E}$ from U to $\widehat{V}$ with capacity 1. Now decompose the maximum flow into paths and cycles. The two matchings are constructed by taking alternate edges of each path and cycle. The larger one is taken as $\mathcal{M}_1$, and the smaller as $\mathcal{M}_2$. The following is proved in [46], which we present without proof.

Theorem 3.7 ([46]). Two Suggested Matchings (non-adaptive) achieves a ratio of $\frac{1-\frac{2}{e^2}}{\frac{4}{3}-\frac{2}{3e}} \simeq 0.67$ with high probability for the online bipartite matching problem in the Known IID model. This ratio is tight for the algorithm. No algorithm can do better than 0.98.

The above theorem holds for the case of integral rates only, that is, when the distribution $\mathcal{D}$ is such that the expected number of arrivals of each type is integral. These results were later improved by Bahmani and Kapralov [15] to a 0.699 ratio algorithm (by a clever modification in the construction of the two matchings from the max flow), and an upper bound was improved to 0.902. Subsequently, Manshadi et al. [74] studied the problem in some more detail. They first provided a different

algorithm, based on TSM, but through a better method of finding the two matchings in $\widehat{G}$: The algorithm first finds a fractional matching f in $\widehat{G}$, where each edge has the same weight as its probability of being in the optimal matching of the outcome graph G. It then constructs a distribution μ on matchings so that every edge has the same probability of being in a matching as its value in the fractional matching, so $\sum_{M,e\in M} \mu_M = f_e,\ \forall e \in \widehat{E}$. The algorithm now picks $\mathcal{M}_1$ and $\mathcal{M}_2$ independently from μ, and runs Two Suggested Matchings (non-adaptive). This method of choosing the two matchings makes the analysis much simpler since it is easier to bound OPT. The authors prove that this algorithm achieves a ratio of 0.684, slightly more that in [46], although lower than in [15].

They then note that all three variants above are non-adaptive, that is, the two matchings are fixed, and the first arrival of a vertex type only tries the neighbor from the first matching. They prove that these algorithms crucially use the fact that the arrival rates of the types are integral by showing a family of examples in which the rates are arbitrarily small ($o(1)$), such that no non-adaptive algorithm can achieve a factor better than $1 - \frac{1}{e}$. They modify the algorithm to make it adaptive: (a) first, the two choices for each vertex are picked non-independently, in order to minimize the probability of the two choices being equal. (b) Second, the first arrival of a vertex type is allowed to match to the neighbor according to the second matching, if the first is already matched (as in Algorithm 5). They show that this algorithm achieves a ratio of 0.702 (even for the case when the rates are not integral). They also improved the upper bound for the Known IID model to 0.83.

Theorem 3.8 ([74]). Two Suggested Matchings (in its adaptive form, with non-independent sampling) achieves a ratio of 0.702 in expectation for the online bipartite matching problem in the Known IID model, even with non-integral rates. No algorithm can do better than 0.823. No non-adaptive algorithm can do better than $1 - \frac{1}{e}$ when the rates are non-integral.

One may try to improve the competitive ratio by using more than two offline matchings. However, using three matchings already leads to

difficulty in the analysis. In [53], Haeupler et al. describe an algorithm which uses two new ideas: using a "discounted" version of the offline LP, and using a third "pseudo-matching" to guide the online choices (see details in [53]). This algorithm obtains an incrementally improved ratio of 0.703, although only with integral arrival rates.

The best ratios are provided by Jaillet and Lu in [56], where they provide an algorithm related to, but somewhat more general than, the algorithms above (see the paper for details).

Theorem 3.9. With integral arrival rates, the algorithm in [56] achieves a ratio of $1 - \frac{2}{e^2} \simeq 0.729$. Without integral rates, the algorithm achieves a ratio of 0.706.

Open Question 3. Find a better algorithm or analysis that can capture the power of more than two matchings. In general, close the gap between the upper and lower bounds. In some sense, the ratio of $1 - \frac{2}{e^2}$ achieved in [56] for the integral case, is a nice "round" number, and one may suspect that it is the correct answer.

4

Online Vertex-Weighted Bipartite Matching

The online vertex-weighted bipartite matching problem was introduced by Aggarwal et al. [5]. It is a strict generalization of the online bipartite matching problem: We have a graph $G(U, V, E)$ where U is known in advance and vertices in V arrive online and reveal their neighbors in U, as before. Each vertex $u \in U$ has a non-negative weight w_u, which is also known in advance. The goal is to maximize the sum of weights of vertices in U that get matched. In Section 4.1, we will study this problem in the Adversarial Model. We will describe the results in the Known IID model in Section 4.2. The problem is still open in the Random Order model.

4.1 Adversarial Order

4.1.1 Intuition

We have seen two algorithmic strategies for online matching: Greedy and Ranking. For the vertex-weighted bipartite matching problem, Greedy means match the arriving vertex v to that available neighbor $u \in U$ with the highest weight. By Ranking we mean ignore the weights and do Ranking on the unweighted graph.

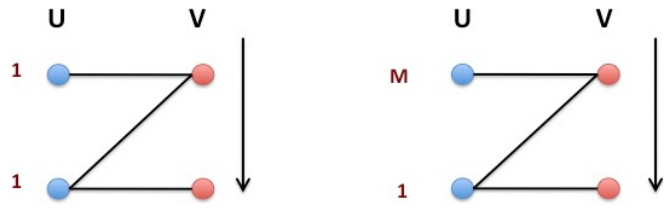

Fig. 4.1 Two extreme examples for vertex-weighted bipartite matching ($M \gg 1$). GREEDY achieves $\frac{1}{2}$ for the left instance and 1 on the right. RANKING achieves 3/4 on the left instance and close to $\frac{1}{2}$ on the right. These examples can be generalized to make GREEDY achieve $\frac{1}{2}$ and 1, and RANKING achieve $1 - \frac{1}{e}$ and 0 respectively.

We first see that neither of these two techniques performs well. First, consider the graph shown on the left in Figure 4.1. Here the two vertices in U have the same weight. For this instance RANKING achieves a ratio of $\frac{3}{4}$. From the results in Section 3.1, we know that it achieves $1 - \frac{1}{e}$ for any instance in which the weights are all equal. However, GREEDY only achieves a ratio of $\frac{1}{2}$ for such instances.

On the other hand, consider the graph on the right in Figure 4.1. Here the weight of one vertex is much greater than that of the other ($M \gg 1$). For this graph, *Greedy* is almost optimal, since it obtains the large reward at the first opportunity. However, RANKING achieves a ratio of only $\frac{1}{2}$ since it ignores the weights (one can construct larger examples in which the ratio of RANKING goes to 0).

To summarize, while RANKING works optimally for uniformly or near-uniformly weighted graphs, it can fail badly when the vertex weights are highly skewed. On the other hand GREEDY does well for highly skewed weights but achieves only $\frac{1}{2}$ when the weights are equal.

These examples reveal the fact that this problem has two different aspects to it, which need two very different approaches. This suggests that the correct method may be to find a hybrid of the two strategies. There are two candidate strategies one may explore:

(1) RANKING **with non-uniform permutations:** Instead of using a random permutation for RANKING, we can draw a permutation from a distribution that depends on the weights on the vertices (with vertices of high weight more probable to be in higher positions).

(2) **A perturbed version of** Greedy**:** Run Greedy with some kind of a randomized "soft-max" which allows weights to change order with some probability.

4.1.2 A Generalization of Ranking

In [5], the authors presented the following algorithm, which simultaneously implements both the hybridization strategies described above. To be precise, the algorithm is defined as Greedy on perturbed weights, and is also a strict generalization of Ranking.

Define

$$\psi(x) = 1 - e^{x-1}$$

This function is graphed in Figure 4.2. Let $\mathcal{U}[0,1]$ denote the uniform distribution on $[0,1]$.

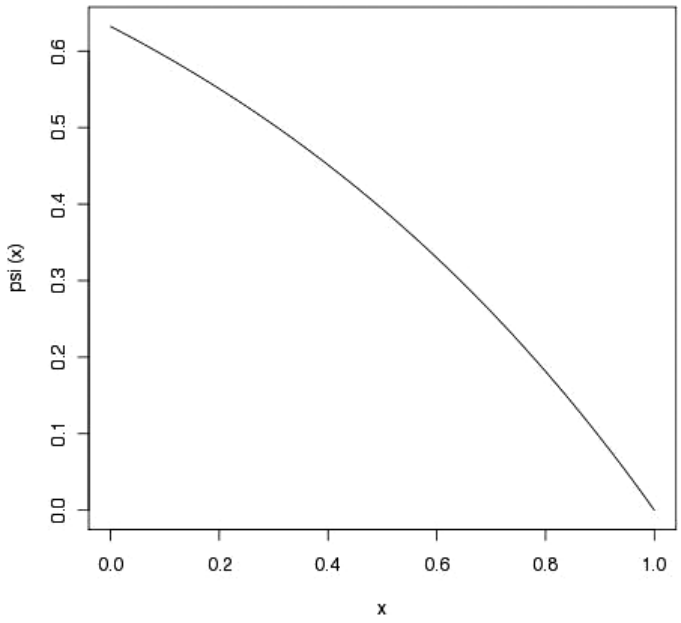

Fig. 4.2 The trade-off function ψ.

Algorithm 7: Perturbed Greedy/ Generalized Ranking (AGKM11)

Offline:
For every $u \in U$:
Pick r_u IID from $\mathcal{U}[0,1]$.
Define its perturbed weight as $\widetilde{w}_u := w_u \psi(r_u)$.
When the next vertex $v \in V$ arrives:
If v has no available neighbors, continue.
Else match v to that available neighbor u which has the maximum value of $\widetilde{w}_u$.

The algorithm has two interpretations:

Perturbed Greedy: This is how we have explicitly defined the algorithm. Instead of picking the highest weighted neighbor as in Greedy, the algorithm first perturbs each weight randomly according to a specific distribution (i.i.d.), and then picks the neighbor with the highest perturbed weight. Note that when the weights are highly skewed (say, if we have exponentially increasing weights) then the algorithms performs very similarly to Greedy w.h.p., since the perturbations will not change the order between very different weights, w.h.p.

Generalized Ranking: Note that if all the weights were equal, then the algorithm is precisely Ranking: the process of picking the r_u from $\mathcal{U}[0,1]$ and choosing the vertex with the highest $\psi(r_u)$ is identical to the process of choosing a uniformly random permutation and picking the highest vertex according to it. In the case of general weights, one can interpret the algorithm as running Ranking on a non-uniform distribution on permutations, in which permutations which have vertices with higher weights in higher positions are given a higher probability of being chosen. This distribution on permutations cannot be concisely written down, but is implicit in the formulation.

In [5] the authors proved the following theorem, which we present here without proof.

Theorem 4.1 ([5]). PERTURBED GREEDY/GENERALIZED RANKING achieves a ratio of $1 - \frac{1}{e}$ for the vertex-weighted bipartite matching problem, for any set of vertex weights. This is optimal (follows from the unweighted case).

Remark. One important question which we did not address is: where did the form of the function ψ come from? We will later describe (in Section 5.2.2) how the proof for this theorem follows the structure of the proof for a very different algorithm (for the Adwords problem with small bids). That algorithm also uses the same function to scale bids, and we will briefly mention the origin of the function when studying that problem.

Remark. The motivation for studying this problem comes from a more general problem, that of Adwords (with general bids), which is still an open problem. Vertex-weighted bipartite matching is equivalent to the special case of Adwords with equal bids, that is, all bids placed by a bidder are equal (see Section 5.4 for the simple reduction).

4.2 Known IID

Two papers have considered the vertex-weighted bipartite matching problem in the Known IID input model: Haeupler et al. [53] provide an algorithm for an even more general problem, the weighted matching problem, in which the weights w_e are on the edges (we will study this problem in detail in Section 7). Their algorithm uses the framework of the algorithm TWO SUGGESTED MATCHINGS, which was introduced in Section 3.3 for bipartite matching in the Known IID model, achieves a competitive ratio of 0.667. In the second paper, Jaillet and Lu [56] study online matching problems in the Known IID setting, and provide a general algorithm template related to the algorithms in [46, 74]. For online vertex-weighted matching, their algorithm provides a ratio of 0.725. Both these results assume integral arrival rates, as defined in Section 3.3 — that the expected number of arrivals of any type of vertex is integral.

Theorem 4.2([53, 56]). There exist algorithms for the online vertex-weighted matching problem in the Known IID model with integral arrival rates, which achieve competitive ratios of 0.667 ([53]) and 0.725 ([56]).

Open Question 4. Improve upon the $1 - \frac{1}{e}$ ratio in the Random Order or Unknown IID setting. Improve upon 0.725 ratio in the Known IID setting, and find a good competitive ratio when the arrival rates are not integral. In particular, can we achieve $1 - \frac{2}{e^2} \simeq 0.729$ (which is the ratio achieved in [56] for the unweighted problem with integral arrivals)?

5

Adwords

The Adwords problem is the following generalization of online bipartite matching: Vertices in U have budgets B_u, and edges $(u,v) \in E$ have bids bid_{uv}. When a vertex $v \in V$ arrives, we have to match it to some neighbor $u \in U$ who has not yet spent all its budget. Once we match v to u, then u depletes bid_{uv} amount of budget. If a vertex u finishes its entire budget, it becomes unavailable. The goal is to maximize the total money spent.

The motivation for the Adwords problem comes directly from online advertising. The ad platform — which is the search engine in the case of sponsored search — is in contract with a set of advertisers U (also called bidders). Each bidder puts in a set of bid values for different keywords they would like to show their ad against. Each bidder can also put in a daily budget value B_u representing the maximum they can pay in a day. The search engine knows these bids in advance. It gets a sequence of online ad-requests $v \in V$ (also called queries and ad-slots) as users search during the day. When a request v arrives, then for each bidder $u \in U$, the search engine sees a bid of bid_{uv} as entered by u. Each ad slot v can be allocated to one advertiser and the winning advertiser

is charged its bid for v.[1] Once an advertiser's budget is exhausted, it cannot be allocated any more ad slots. The objective is to maximize the total efficiency of the matching, which, in this model, is equivalent to maximizing the total amount of money spent by the advertisers.

5.1 The Small-Bids Assumption

We will identify a special case of the Adwords problem by making the following assumption.

The small-bids assumption: For every edge $(u, v) \in E$, bid_{uv} is very small compared to the corresponding B_u.

This means that each advertiser puts in bids that are small compared to their budget, which is a very realistic assumption in this setting, since advertisers typically want a large number of ad clicks per day. We will postpone the precise definition of bids being small compared to the budgets; for the present purposes we may take this to mean infinitesimally small. When relevant, we will present the exact dependence on the bid to budget ratio parameter. For the rest of this Section we will make the small-bids assumption throughout, unless we explicitly state otherwise.

The Adwords problem and the small-bids assumption were introduced by Mehta et al. in [78]. Besides the classic results in [58, 63], this is chronologically the first among the results surveyed here.

5.2 Adversarial Order

In this section, we will study the Adwords problem with the small bids assumption in the adversarial model.

Intuition: In order to build some intuition towards what kind of algorithm would work for the problem, we first look at two simpler algorithms — Greedy and Balance, defined below.

[1] This is a simplifying assumption — in search engine ad pages there are multiple slots on a page, and the pricing scheme is second price, pay-per-click. Some of these features can be modeled theoretically, and others heuristically. We will return back to these assumptions in Sections 9 and 10.

Algorithm 8: GREEDY For Adwords

When the next vertex $v \in V$ arrives:
If all neighbors of v are unavailable (that is, have spent their budgets), continue.
Else match v to that available neighbor u which has the maximum value of bid_{uv}.

GREEDY attempts to get the maximum gain at every step by assigning each slot to its highest available bidder. One technical point to note is that all the algorithms presented here need to cap each bid by the remaining budget. So, if a vertex u has a budget of B_u and has spent $S_u \leq B_u$ amount of its budget, then we take the modified bid $\widehat{\text{bid}_{uv}} = \min\{\text{bid}_{uv}, B_u - S_u\}$. With the small bids assumption, however, one can ignore this bid-capping and take the bid to be the original bid itself. In doing this, a vertex u may spend an amount $S_u > B_u$ (by overspending for the last allocated vertex) but, in that case, $S_u - B_u < \max_v \text{bid}_{uv}$, which is a small quantity by assumption. This makes the algorithms and analysis simpler to describe.

For GREEDY, we can prove the following:

Theorem 5.1. GREEDY achieves a ratio of $\frac{1}{2}$ for the Adwords problem (even without the small bids assumption).

Proof. For $u \in U$, let B_u be its budget and let S_u be its spend in the algorithm. For $v \in V$, suppose the optimal allocation OPT allocated v to u, obtaining a value of opt_v (opt_v could be less than bid_{uv} if u exhausted its budget in OPT). Let alg_v be the value obtained by the algorithm in allocating v. Let $V' \subseteq V$ be the set of vertices v for which $\text{alg}_v < \text{opt}_v$. Then the loss of the algorithm is $\text{Loss} = \sum_{v \in V'} (\text{opt}_v - \text{alg}_v)$. We partition this loss according to the vertex $u \in U$ to which v was allocated in OPT. Let V'_u be the set of vertices in V' which were allocated to u in OPT. Then, $\text{Loss} = \sum_{u \in U} \text{Loss}_u$, where

$$\text{Loss}_u := \sum_{v \in V'_u} (\text{opt}_v - \text{alg}_v) \leq B_u - \sum_{v \in V'_u} \text{alg}_v \tag{5.1}$$

Now, if $V'_u \neq \emptyset$, then consider a $v \in V'_u$. When v arrived, it was allocated to some vertex u' yielding a value $\text{alg}_v < \text{opt}_v$. This means that, at that time, u had at most alg_v budget unspent. Therefore,

$$S_u \geq B_u - \text{alg}_v, \quad \forall\ v \in V'_u$$

Plugging this into Equation (5.1), we get

$$\text{Loss}_u \leq S_u + \text{alg}_{v^*} - \sum_{v \in V'_u} \text{alg}_v, \quad \forall\ v^* \in V'_u$$

Therefore $\text{Loss}_u \leq S_u$ for all u (if $V'_u = \emptyset$ then $\text{Loss}_u = 0$). Now, $\text{OPT} - \text{ALG} = \sum_u \text{Loss}_u \leq \sum_u S_u = \text{ALG}$, proving a ratio of $\frac{1}{2}$. □

On the other hand, BALANCE tries to keep the spend of all the bidders as equal as it can, irrespective of bid, so as to keep all bidders available for as long as possible:

Algorithm 9: BALANCE

When the next vertex $v \in V$ arrives:

If all neighbors of v are unavailable (that is, have spent their budgets), continue.

Else match v to that available neighbor u which has spent the least fraction of its budget so far.

We will now see that each technique works well on some set of inputs, and fails on other types of inputs. Consider the example in Figure 5.1. There are two bidders in U, A and B, each with a budget of \$100, and

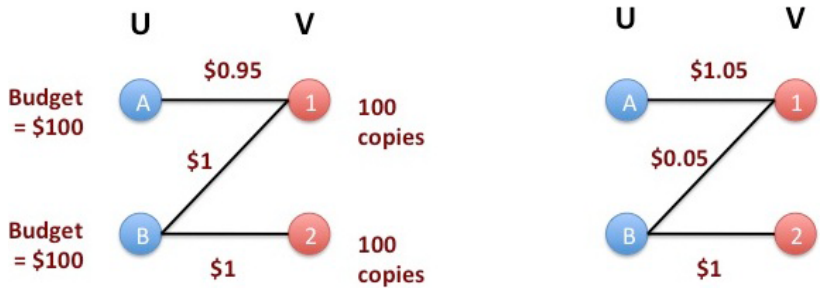

Fig. 5.1 Two extreme examples for the Adwords problem.

two types of vertices in V, 1 and 2. First a 100 copies of the type 1 vertex arrive, and then a 100 copies of type 2. The examples on the left and right differ in the bid values on the edges. In both examples, though, the optimal offline allocation is to allocate vertices of type 1 to A and type 2 to B, giving $\text{OPT} = 200$.

Let us first look at the example on the left. GREEDY will assign the 100 copies of 1 to B, which would then exhaust its budget. So the 100 copies of type 2 would remain unsold, since A does not bid for them. On the other hand, BALANCE would achieve a ratio of 3/4 on the left example by equally distributing the 100 copies of type 1. Indeed, let us distinguish a special case of the Adwords problem:

Online *b*-Matching: This is the special case of the Adwords problem when all budgets are equal to b (a positive integer), and all non-zero bids are equal to 1.

For this special case, it was shown by Kalyanasundaram and Pruhs [58] that BALANCE is an optimal algorithm (we will prove this theorem later in this section, as a special case of the result for the Adwords problem):

Theorem 5.2 ([58]). BALANCE achieves a ratio of $1 - \frac{1}{(1+\frac{1}{b})^b}$ for the Online b-Matching problem, which goes to $1 - \frac{1}{e}$ as $b \to \infty$. This is optimal.

Thus, for examples like the one on the left of Figure 5.1, in which all bids are equal (or close to equal), BALANCE performs well, while GREEDY could perform as bad as $\frac{1}{2}$. We note that a different proof of this theorem was also provided by Azar and Litichevskey [13] (by relating b-matching to the problem of fractional bipartite matching in which a vertex can be fractionally allocated).

Now let us look at the example on the right of Figure 5.1, where the bids by A and B for the vertices of type 1 are very different. It can be seen that GREEDY does optimally achieving a ratio of 1, whereas BALANCE gets a ratio of $\frac{1}{2}$ since it ignores the bids.

The main intuition in solving the general problem is to find a hybrid algorithm that combines these two algorithmic ideas and performs well

on all instances. Such an algorithm was provided in [78]. The algorithm scales the bid of an advertiser as a function of the fraction of its budget spent, and then runs GREEDY on the scaled bids.

At any point during the run of the algorithm, let x_u be the fraction of advertiser u's budget that has been spent up to this point. Let v be the next query to arrive. Define the *scaled bid* of u for v as the original bid bid_{uv} scaled down by a multiplicative factor

$$\psi(x_u) = 1 - e^{x_u - 1}$$

The algorithm allocates v to the advertiser with the highest scaled bid. Note that the winning advertiser still pays its original bid; the scaling is used only for winner determination. The algorithm is defined formally below:

Algorithm 10: MSVV

When the next vertex $v \in V$ arrives:
Allocate v to the bidder u which maximizes $\text{bid}_{uv}\psi(x_u)$
(where x_u is the fraction of u's budget spent by this time).

Remark. Note that when the bids on the edges are are all equal (thus reducing to the b-matching problem) then the algorithm becomes precisely BALANCE, since the function ψ is a monotonically decreasing function. If the bids for a query are very well separated, then the algorithm, in most cases, would not flip the order of the bids, and go with the higher bidder like GREEDY (unless the higher bidder has spent a very large fraction of its budget, as compared to the lower bidder). When the budgets are all infinity, then the algorithm becomes GREEDY (which is optimal in this case) since all the bid scaling numbers remain $\psi(0) = 1 - \frac{1}{e}$ throughout.

Remark. Note the similarity with the algorithm for vertex-weighted bipartite matching. We use the same function ψ to modify the input bids and then run the GREEDY algorithm. However the bid-scaling function used here is deterministic and dynamic, that is, changes over time as the fraction of budget spent changes. In fact, there is more than a cosmetic similarity between these two algorithms; we will describe the connections in some more depth in Sections 5.2.2 and 6.

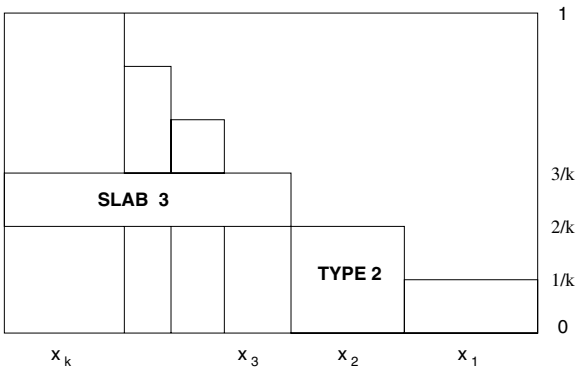

Fig. 5.2 Slabs and types in the final state of the algorithm.

In [78], the authors showed that this algorithm achieves a ratio of $1 - \frac{1}{e}$ and it is optimal, even among randomized algorithms. We provide a proof sketch here.

Theorem 5.3 ([78]). MSVV achieves a ratio of $1 - \frac{1}{e}$ for the Adwords problem. This is optimal even among randomized algorithms.

Proof. For simplicity, we will assume that all bidders in U have a budget of 1. We choose a large discretization constant k, and assume that every bid is much smaller than $\frac{1}{k^2}$, say. Figure 5.2 shows the state of the bidders in U at the end of the algorithm. In this figure, the bidders in U are placed along the x-axis. The y-axis represents the money spent by each (capped at 1, which is the budget), and they are ordered according to their money spent. The spend is discretized in steps of $1/k$.

For $i \in [1, k]$, let α_i be the number of bidders who spent between $\frac{i-1}{k}$ and $\frac{i}{k}$ of their budget at the end of the algorithm (so, $\sum_{i=1}^{k} \alpha_i = n$). These are known as bidders of *type* i. Let *slab* i be the (set of the) money spent by bidders from the $[\frac{i-1}{k}, \frac{i}{k})$ portion of their budgets, and let β_i be the amount of money in slab i. The first observation is a relation between the two sets of variables α and β, which can be seen directly from the geometry of the picture:

$$\forall i : \beta_i = \frac{n - \sum_{j<i} \alpha_j}{k} \tag{5.2}$$

For simplicity, we will assume that the instance is such that OPT spends the entire budget of all bidders in U, so $\text{OPT} = n$. To build intuition, we will first warm-up with the analysis for BALANCE in the special case of b-matching, and then show the analysis for MSVV in the general case.

Special Case: BALANCE for b-matching

Recall that we have scaled the numbers, so that every budget is 1, and every non-zero bid is $\frac{1}{b}$, and note that we have assume that b is very large (say, $\frac{1}{b} \leq \frac{1}{k^2}$). Fix an $i < k$. At the end of the algorithm, we know which bidders have type at most i. Consider all the vertices in V which OPT allocated to these bidders — call this set $V_{\leq i}$. Let $u(v)$ be the bidder to which OPT allocates a $v \in V_{\leq i}$. The total spend by OPT on $V_{\leq i}$ is $\sum_{j \leq i} \alpha_j$ (since we assumed that OPT spends all the budget of all bidders, and all the budgets are 1).

Now, let us look at the run of the algorithm. When a query $v \in V_{\leq i}$ arrives, BALANCE will surely allocate it, since $u(v)$ itself has not spent all its budget even at the end of the algorithm (since $i < k$). In fact, let s be the spend of $u(v)$ at the time v arrives. Then we know: (a) $s \leq \frac{i}{k}$, and (b) By the rule of the algorithm, v is allocated to some bidder whose is at most s. That is, the algorithm gets the spend for v from slabs 1 through i. Now we use the fact that all non-zero bids are equal, so both OPT and the algorithm get the same spend (equal to $\frac{1}{b}$) for each vertex in $V_{\leq i}$. This gives the following set of inequalities:

$$\forall i < k\text{: } \sum_{j \leq i} \alpha_j \leq \sum_{j \leq i} \beta_j \tag{5.3}$$

Here the left-hand side is OPT's spend on queries in $V_{\leq i}$, and the right-hand side is an upper bound on ALG's spend on the same queries. Note that at this point there is no explicit reference to OPT anymore; this is simply a set of constraints on the final state of BALANCE, on any input. We can use Equations (5.2) to write the β_i in terms of the α_i, to get:

$$\forall i < k\text{: } \sum_{j \leq i} \alpha_j \left(1 + \frac{i-j}{k}\right) \leq \frac{i}{k} n \tag{5.4}$$

Also note that the spend of the algorithm is

$$\text{ALG} = \sum_{i \le k} \frac{i}{k} \alpha_i \tag{5.5}$$

We now use the technique of *Factor-revealing LPs* (introduced in [52, 57, 75]): Our constraints show that the final state cannot have any arbitrary set of values for the α_i. Our goal is to show that the spend of the algorithm is not too low. To do this, we minimize the total spend of the algorithm (5.5) under the constraints (5.4). This is a lower bound on the performance of the algorithm on worst possible input. If our constraints capture enough of the structure of the algorithm, then this lower bound would be useful. Indeed, we can write the dual of this LP (which turns out to be identical in form) and analytically solve the primal–dual pair to get the following optimal values of the variables:

$$\alpha_i^* = \frac{n}{k}\left(1 - \frac{1}{k}\right)^{i-1}, \quad (i = 1, \ldots, k-1)$$

which gives that the optimal objective function of the LP is $\left(1 - \frac{1}{e}\right) n$ (as $k \to \infty$). This proves that the ratio of Balance is $1 - \frac{1}{e}$ for b-matching.

General case: MSVV **for Adwords with Small Bids** — For the general small bids problem, Equation (5.2) still holds, but the argument for Balance fails for the following reason. The algorithm will still allocate vertices $v \in V_{\le i}$, but (a), it may get a much smaller spend for v than OPT does, since the bids are no longer all the same, and (b), there is no longer a guarantee to get the spend from the lowest possible slab, since we have lost the form of Balance. So Equation (5.3) no longer holds. Indeed, we seem to have lost all the structure we had in Balance, which is a worrying fact. However, it turns out that we can now make a different argument based on the tradeoff that MSVV makes between bid and spend, which we describe next.

When a query $v \in V$ arrives, consider its OPT bidder $u(v)$ (we will call it u^* henceforth for brevity). Suppose that u^* is of type i and that it has spent $\frac{\widehat{i}}{k}$ fraction of its budget when v arrives ($\widehat{i} \le i$). Suppose MSVV allocates v to u', who has spent $\frac{j}{k}$ fraction of its budget at the

time v arrives (j could be any number in $[0, k]$). Then we know from the form of the algorithm (since it allocated v to u' and not to u^*):

$$\text{bid}_{u^*v} \cdot \psi\left(\frac{\widehat{i}}{k}\right) \leq \text{bid}_{u'v} \cdot \psi\left(\frac{j}{k}\right) \tag{5.6}$$

Since $\widehat{i} \leq i$, and using the fact that ψ is monotonically decreasing, we also know that $\psi(i) \leq \psi(\widehat{i})$, which gives, from (5.6):

$$\text{bid}_{u^*v} \cdot \psi\left(\frac{i}{k}\right) \leq \text{bid}_{u'v} \cdot \psi\left(\frac{j}{k}\right) \tag{5.7}$$

Recall that bid_{u^*v} is the spend of OPT on v, and $\text{bid}_{u'v}$ is the spend of ALG on v. We can write this inequality for every $v \in V$ and add them all up. If we collect the terms for $\psi(\frac{i}{k})$ on the left, then the sum of the coefficients bid_{u^*v} becomes the total spend of OPT from bidders of type i, which is α_i. Similarly, collecting the terms for $\psi(\frac{j}{k})$ on the right, we get the total spend of the algorithm from slab j, that is, β_j. So we get:

$$\sum_{i=1}^{k} \psi\left(\frac{i}{k}\right) \alpha_i \leq \sum_{i=1}^{k} \psi\left(\frac{i}{k}\right) \beta_i \tag{5.8}$$

Note that, unlike the case for BALANCE, we get a single constraint, and not a set of constraints. But this single constraint captures all the structure of the algorithm. Now we can use (5.2) in (5.8) to write the β_is in terms of the α_is, to get a constraint purely in the latter. Finally, using the explicit form of the tradeoff function $\psi(x) = 1 - e^{x-1}$, we get:

$$\sum_{i=1}^{k} \frac{i}{k} \alpha_i \geq n\left(1 - \frac{1}{e}\right), \quad (\text{as } k \to \infty)$$

But the left-hand side is precisely ALG, thus proving the theorem. □

5.2.1 Where did the Tradeoff Function ψ Come From?

In the proof above, we crucially used the form of the function $\psi(x) = 1 - e^{x-1}$. The obvious question is how did we come up with this function in the first place. In [78] the authors provided a combinatorial

approach to find this tradeoff function, as well as the ratio of the algorithm. They start with the factor revealing LP for Balance for b-matching:

$$\text{Minimize (5.5), subject to (5.4)}$$

As we have seen, this LP does not hold in the general case. However, they showed how the LP can be modified to capture the general case by a vector of new variables, representing the deviation from Balance. From this modification, they obtained the function ψ as a prefix-sum of the optimal dual variables of the original LP for Balance. These dual variables can be explicitly written down (for a fixed k), and their prefix sum yields precisely the function $1 - e^{x-1}$. They called their technique a tradeoff-revealing LP, since the LP reveals not only the ratio of the algorithm (as in the technique of factor-revealing LPs), but also the form of the algorithm itself, in terms of the optimum tradeoff to use between bid and spend fraction. In Section 6 we will see a different approach for this problem which yields the same function. This approach, called the Online Primal–Dual technique, is based on a different LP, the allocation LP for the problem.

5.2.2 Relationship between MSVV and Perturbed Greedy

The algorithms Perturbed Greedy (for vertex-weighted bipartite matching) and MSVV (for Adwords with small bids) are very similar in form. The former uses a static randomized scaling function to scale the weights, while the latter uses a dynamic deterministic scaling function to scale the bids, but both use the same tradeoff function $\psi(x)$. We have not provided the proof for Perturbed Greedy in this survey, but there is, in fact, a close relationship between the two proof techniques as well. The slabs in the proof for MSVV correspond to the random seeds r_u of the vertices in U for Perturbed Greedy. As lower slabs get filled earlier during a run of MSVV, so also, vertices with higher random seeds get matched with higher probability in Perturbed Greedy. Eventually the two proofs end up with the same structure and inequalities, although for very different reasons. This suggests that a unified algorithm may be found for common generalizations

of these two problems. In Section 6 we will describe the online primal–dual approach to both these problems, which also yields very similar algorithms and proof structure for these two problems. Thus we have a very strong indication that there is a common generalization of these two algorithms to one which works for a common generalized problem.

Open Question 5. Find a common generalization of MSVV and Perturbed Greedy, for some problem which generalizes both Adwords with small bids and vertex-weighted bipartite matching.

5.3 Random Order and IID

We now shift our focus to the stochastic input models. In [51], Goel and Mehta showed that Greedy achieves a ratio of $1 - \frac{1}{e}$ for Adwords with small bids in the random order model, and hence in the Unknown IID model as well. The question remained open as to how well does MSVV itself perform in the IID model. For the special case of the b-matching problem (with equal budgets, and large b), Motwani et al. [81] proved that Balance is near-optimal $(1 - o(1))$. This was generalized to an algorithm called Water Level by Charles et al. [26] for the unequal budgets case. This is a non-trivial generalization of Balance, in which the next bin chosen is the one which minimizes the failure probability of the rest of the algorithm (see [26, 36, 34]). In [79], Mirrokni et al. proved that for the Adwords with small bids problem, MSVV provides a ratio of 0.76 (better than $1 - \frac{1}{e} \simeq 0.63$) in the Unknown IID model.

Thus, Balance and MSVV provide a good simultaneous approximation in the worst case and IID models. If the input sequence is IID, then the guarantees are much better than the ones provided in Theorems 5.2 and 5.3 for the adversarial order. Note that the algorithms were designed for the worst case, and do not actually use the distribution as an input parameter — the distribution is unknown in advance. In fact, it is interesting that the algorithms do not even adapt themselves as they learn the distribution over time. This kind of dynamic adaptation does not seem necessary for the b-matching problem, since Balance and Water Level are optimal. However, MSVV still has a ratio of 0.76, which could be suboptimal in this

model. It turns out that for Adwords (with small bids), one does need to be dynamic, and adapt the bid-scaling factors to the incoming distribution. An algorithm due to Devanur and Hayes [35] achieves a $1 - \epsilon$ ratio for this problem in the Random Order model, where ϵ is very small under the small bids assumption. The algorithm uses the initial prefix of the input sequence to compute optimal bid-scaling numbers (as compared to MSVV, which uses a fixed bid-scaling function designed for the worst case). The algorithm and its analysis is based on the online Primal–Dual framework, and we will describe it in Section 6 (Theorem 6.3). These results are summarized in the following theorem.

Theorem 5.4 ([26, 35, 51, 79, 81]). Greedy achieves a ratio of $1 - \frac{1}{e}$ for the Adwords problem with small bids in the Random Order Model. Balance achieves a $1 - o(1)$ ratio in the Unknown IID model for the b-matching problem (and Water Level for unequal budgets), with large budgets. MSVV achieves a ratio of 0.76 for the Adwords problem with small bids in the Unknown IID model. The algorithm in [35] achieves a $1 - \epsilon$ ratio in the Random Order Model, where ϵ depends on the bid to budget ratio.

See Devanur [34] for details on some of these algorithms in the stochastic input models. A related result along these lines is due to Mahdian et al. [72], who provided a hybrid algorithm based on MSVV (which performs well in the adversarial case), and the offline LP (which performs optimally if the queries are known). The hybrid performs well when the estimates of the queries are accurate, as well as when they are completely incorrect. Note that this algorithm works in the Known IID model (to be precise, in a slightly different, stronger model), since it needs to know the distribution in advance, so as to construct the offline LP.

5.4 General Bids

The Adwords problem without the assumption of small bids remains an interesting open question at this time. Firstly, we know from Theorem 5.1 that Greedy achieves a ratio of $\frac{1}{2}$ in the adversarial model.

Besides, this, for the adversarial model, the only other result known is for a special case of the problem, called the Equal Bids case.

Equal Bids Case: For each $u \in U, v \in V$, $\text{bid}_{uv} = \text{bid}_u$, for some values bid_u. That is, each bidder u makes the same bid across its incident edges; different bidders can make different bids for the same query.

One can reduce this problem to the vertex-weighted bipartite matching problem by splitting each vertex u into $\lfloor B_u/\text{bid}_u \rfloor$ vertices, each with a weight of bid_u. Thus we have a $1 - \frac{1}{e}$ ratio algorithm for this case.

In the Unknown IID model, a result due to Devanur et al. [36], gives an improved competitive ratio for the general problem:

Theorem 5.5 ([36]). GREEDY achieves a ratio of $1 - \frac{1}{e}$ for the Adwords problem (without any restriction on bids) in the IID arrival model.

As we shall see in Section 8, this result can be extended all the way to the submodular welfare problem.

In the same paper, the authors also proved that if the bid to budget ratio is at most $\frac{\epsilon^2}{\log n}$, then there is an algorithm achieving a ratio of $1 - O(\epsilon)$. Subsequently, Devanur et al. [37] considered the intermediate case between small bids and general bids, and improved the result in [36]. They parameterize the problem by a parameter k, s.t. every bid is at most $\frac{1}{k}$ of the corresponding budget. They also reduce the bid to budget ratio bound required to achieve $1 - O(\epsilon)$, and also prove a parameterized impossibility result.

Theorem 5.6 ([36]). There exists an algorithm which achieves a ratio of $1 - \frac{1}{\sqrt{2\pi k}}$ for the Adwords problem in the Unknown IID arrival model (with knowledge of the optimal budget utilization), when the bid to budget ratios are at most $\frac{1}{k}$. If the bid to budget ratios are at most ϵ^2, then the algorithm achieves a competitive ratio of $1 - O(\epsilon)$. No algorithm can achieve a competitive ratio better than $1 - o(1/\sqrt{k})$ when the bid to budget ratios are as large as $\frac{1}{k}$.

This algorithm does need a little more information, which is the budget utilization in the optimal solution on the distributional instance (see the paper for details). The algorithm works even in the case when the input distribution changes over time.

Open Question 6. Find an algorithm which beats the ratio of $\frac{1}{2}$ in the adversarial model.

We can imagine two possible candidate algorithms, which, along with some variants, seem to be promising approaches.

The first is a common generalization of MSVV and Perturbed Greedy (as asked in Open Question 5). Pick a random seed r_u from $\mathcal{U}[0,1]$, for every $u \in U$ IID. Let x_u be the spend of u at the time $v \in V$ arrives. Allocate v to that neighbor u who maximizes $\psi(x_u + r_u \text{bid}'_{uv})$, where $\text{bid}'_{uv} = \min\{\text{bid}_{uv}, u\text{'s remaining budget}\}$.

The second idea ignores the fraction of budget spent as a parameter: Pick r_u from $\mathcal{U}[0,1]$, for every $u \in U$ IID, and allocate the arriving v to the u which maximizes $\psi(r_u)\text{bid}'_{uv}$, where again, bid'_{uv} is the capped bid. This is basically Perturbed Greedy itself, and the first step would be to prove that it provides a good ratio for Adwords with the small bids assumption, in expectation.

For the IID model, while the results in [37] show that we can achieve a ratio strictly better than $1 - \frac{1}{e}$ when the bid to budget ratio is at most $\frac{1}{2}$ (with additional optimal budget utilization information), the general case is still open.

Open Question 7. Can we obtain a ratio better than $1 - \frac{1}{e}$ in either IID model, for general bids?

6

The Online Primal–Dual View

The algorithms that we saw in the earlier sections — Ranking for the online bipartite matching problem, Perturbed Greedy for the vertex-weighted version, and MSVV for the Adwords problem with small bids — were all combinatorial in nature, as was their analysis. In a parallel line of work, the same problems have been solved using the Online Primal–Dual framework (introduced by Buchbinder and Naor [22]). In this section we will show how this approach yields the same algorithms and proofs.

6.1 Adwords with Adversarial Order

Buchbinder et al. [21] studied the Adwords problem with the small bids assumption using the Primal–Dual approach. They begin with the Primal–Dual pair of LPs for the (fractional) allocation problem, given in Figure 6.1.

If the problem were offline, we would know the entire LP in advance and therefore could solve the primal LP to get the optimal allocation x_{uv}, as we saw in Section 2. (This would be a fractional solution, but we can show that rounding a basic feasible solution would lose very little in objective value.) In the online problem, since we do not know V, we

$$\begin{aligned}
&\max \sum_{u,v} x_{uv} bid_{uv} && \min \sum_{u} B_u \alpha_u + \sum_{v} \beta_v \\
&s.t.\ \forall\ u\ :\ \sum_{v} x_{uv} bid_{uv} \le B_u && s.t.\ \forall\ u,v:\ \ bid_{uv}\ \alpha_u + \beta_v\ \ge\ bid_{uv} \\
&\forall\ v:\ \ \sum_{u} x_{uv} \le 1 && \alpha_u \ge 0\ \ \forall u \\
&x_{uv} \ge 0 && \beta_v \ge 0\ \ \forall v
\end{aligned}$$

Fig. 6.1 The Primal–Dual pair of LPs for the fractional version of the allocation problem in Adwords.

know very little of the LP in advance. As vertices $v \in V$ arrive, we get to see new columns of the LP matrix, and more terms in the objective function. Let us investigate the structure of the offline solution more closely, via the complementary slackness conditions:

$$x_{uv} > 0 \Rightarrow \text{bid}_{uv}(1 - \alpha_u) = \beta_v \tag{6.1}$$

$$\alpha_u > 0 \Rightarrow \sum_{v} x_{uv} \text{bid}_{uv} = B_u \tag{6.2}$$

$$\beta_v > 0 \Rightarrow \sum_{u} x_{uv} = 1 \tag{6.3}$$

The interpretation is that a feasible pair of primal and dual solutions $\{x_{uv}, \alpha_u, \beta_v\}$ are optimal if (a) v is allocated to the bidder who maximizes the scaled bid $\text{bid}_{uv}(1 - \alpha_u)$ (6.1). (b) The dual variable α_u is positive (that is, the scaling factor is less than 1) only for bidders u who have finished their budget (6.2). Note that (a) implies that given only the optimal dual variables α_u, one can reconstruct the optimal primal solution by allocating each vertex v to the bidder u who maximizes $\text{bid}_{uv}(1 - \alpha_u)$.

In the online problem, the entire LP is not known before time, so we do not have access to the optimal dual variables. The idea in the online Primal–Dual approach is that one can maintain a best estimate for the optimal dual variables α_u, and continue to use (6.1) as a guide in making the allocation and get an allocation with good competitive ratio. As a warm-up, let us analyze Greedy and prove that it has

ratio $\frac{1}{2}$. We will do this only with the small-bids assumption (although we know from Theorem 5.1 that the ratio is the same even without it).

A Primal–Dual proof for GREEDY:

Theorem 6.1. GREEDY achieves a ratio of $\frac{1}{2}$ for the Adwords problem with the small-bids assumption.

Proof. We will construct a primal and a dual solution online, as we run the algorithm. The primal solution will give exactly the value of the solution obtained by the algorithm, while the dual will give us a lower bound on the competitive ratio. We will prove that at the end of the algorithm: (1) the solutions are feasible for both primal and dual, and that (2) the primal objective is at least $\frac{1}{2}$ of the dual objective. This will suffice to prove the theorem, as the following sequence of inequalities shows:

$$\text{ALG} = \text{Primal} \geq \frac{1}{2}\text{Dual} \geq \frac{1}{2}\text{Dual}^* \geq \frac{1}{2}\text{OPT} \tag{6.4}$$

where Primal and Dual are the values of the solutions we constructed, Dual^* is the value of the optimal dual solution, and OPT is the value of the optimal integral primal solution. The first inequality is the condition (2) above, the second is true because the dual solution we constructed is feasible (condition (1)), and the third is true since OPT is no greater than the fractional optimal primal solution (which is equal to Dual^*, by strong duality). This proves that GREEDY has a ratio of $\frac{1}{2}$.

It remains to describe how we set the primal and dual variables during the run of GREEDY, and to prove that (1) and (2) hold.

We will use the following setting of the dual variables. We begin with all variables, $x_{uv}, \alpha_u, \beta_v$ initialized to 0. At any time, if a bidder u finishes its budget, then we will update its variable $\alpha_u = 1$. Thus,

$$\alpha_u = \begin{cases} 0, & \text{if } u \text{ has not yet finished its budget} \\ 1, & \text{if } u \text{ has finished its budget} \end{cases}$$

When a vertex $v \in V$ arrives, we allocate it to that available neighbor u with the maximum value of bid_{uv}. Note that since every available vertex

u has $\alpha_u = 0$, this is the same as allocating to the bidder with the highest value of $\text{bid}_{uv}(1 - \alpha_u)$ (we will use this form in later algorithms). If we allocate v to u, then we will update $\beta_v = \text{bid}_{u'v} \geq \text{bid}_{uv}$, and $x_{uv} = 1$. We can now show that the two conditions hold:

(1) Feasibility. The primal solution is feasible, since we do not allocate to a bidder once its budget is finished. For every pair u, v, consider the corresponding dual constraint once the algorithm ends. If u has finished its budget, then $\alpha_u = 1$, so the constraint holds. If u has not finished its budget, then $\alpha_u = 0$, so feasibility reduces to proving $\beta_v \geq \text{bid}_{uv}$. Consider the time that v was allocated. We know that u was available at that time, and that we allocated v to that available bidder u' with the highest bid, and updated β_v to that $\text{bid}_{u'v}$. Therefore $\beta_v \geq \text{bid}_{uv}$.

(2) Primal-Dual ratio. Every time we allocate a vertex v to a bidder u, the primal objective function increases by bid_{uv}. The dual solution also increases by $\beta_v = \text{bid}_{uv}$. In addition, the dual objective increases by B_u whenever a bidder u becomes full (and α_u is increased from 0 to 1). For every full bidder u, B_u is simply the sum of the bids bid_{uv} for vertices v which were allocated to it, and hence already counted in the primal objective. Thus, the dual objective value is at most twice the primal objective value at the end of the algorithm.

This concludes the proof that the competitive ratio is at least $\frac{1}{2}$. Note that we have crucially used the small-bids assumption in this proof: When we allocated v to u we counted a gain of bid_{uv} even if the remaining budget of u was less than that. We would therefore have to subtract a value of $\sum_u \max_v \text{bid}_{uv}$ to account for this over-counting. With the small-bids assumption, this value is very small and does not reduce the ratio. □

Just like we interpreted Greedy as setting the duals α_u to 0 or 1, we can interpret MSVV as finding the best online dual variables α_u as a function of the fraction of budget spent. In [21], the authors provide an algorithm based on this idea; the algorithm looks different in form, but turns out to be identical to MSVV.

For simplicity, we will take all budgets $B_u = 1, \forall\ u$. We make the following definitions:

$$\rho := \frac{1}{1 - \frac{1}{e}}$$

For $x \in [0,1]$:

$$a(x) := \frac{e^x - 1}{e - 1}$$

Note that

$$a'(x) := \frac{\mathrm{d}a(x)}{\mathrm{d}x} = \rho\ e^{x-1}$$

Define

$$\Delta_{uv}(x) := a'(x)\mathrm{bid}_{uv} = \rho e^{x-1}\mathrm{bid}_{uv}$$

At any point of time, let x_u be the fraction of budget of u which has been spent so far.

Algorithm 11: Primal–Dual Adwords (BJN07)

Initialize: $\alpha_u = 0\ \forall\ u, \beta_v = 0\ \forall\ v$.
When the next vertex $v \in V$ **arrives:**
If v has no available neighbors, continue.
Match v to that available neighbor u
which maximizes $\rho\ \mathrm{bid}_{uv} - \Delta_{uv}(x_u)$
Update: $\alpha_u = \alpha_u + \Delta_{uv}(x_u)$
$\beta_v = \rho\ \mathrm{bid}_{uv} - \Delta_{uv}(x_u)$

Remark. While the allocation rule above looks very different from anything seen in the earlier sections, we note that this algorithm is, in fact, identical to MSVV, since

$$\rho\ \mathrm{bid}_{uv} - \Delta_{uv}(x_u) = \rho\ \mathrm{bid}_{uv}(1 - e^{x_u - 1})$$

Theorem 6.2 ([21]). Primal–Dual Adwords (BJN07) achieves ratio $1 - \frac{1}{e}$ for the Adwords problem with the small bids assumption.

Proof. We initialize with a feasible primal and an infeasible dual solution. As in the proof of Theorem 6.1, we will prove that at the end of the algorithm (1) the solution we build online is feasible for both primal and dual, and that (2) the primal objective is at least ρ times the dual objective. This will prove:

$$\text{ALG} = \text{Primal} \geq \frac{1}{\rho}\text{Dual} \geq \frac{1}{\rho}\text{Dual}^* \geq \frac{1}{\rho}\text{OPT}$$

proving a competitive ratio of $1/\rho = 1 - \frac{1}{e}$.

(1) Feasibility. The primal solution is feasible since we do not allocate to any bidder who has finished its budget. To prove dual feasibility, let us first prove an identity for the value of the dual variable α_u as it evolves during the algorithm. At any time, α_u is equal to the sum of the cumulative increments it got during the algorithm, whenever a query was allocated to u. Suppose $v_1, \ldots, v_k$ were allocated to u so far (in that order), and let $x_1 = 0, \ldots, x_k$ be the spends of u at the times when $v_1, \ldots, v_k$ arrive. Then α_u at the time v_k has been allocated is $\sum_{i=1}^{k} \Delta_{uv_i}(x_i)$. But $x_i = \sum_{j=1}^{i-1} \text{bid}_{uv_j}$. So $\alpha_u = \sum_{i=1}^{k} \Delta_{uv_i}(\sum_{j=1}^{i-1} \text{bid}_{uv_j})$. Since we assume that every bid_{uv} is infinitesimally small, we can replace this by an integral:

$$\alpha_u = \int_{x=0}^{x_u} \rho e^{x-1}\,\mathrm{d}x = \rho(e^{x_u - 1} - e^{-1}) = \frac{e^{x_u} - 1}{e - 1} = a(x_u) \tag{6.5}$$

Thus, at any point in time, $\alpha_u = a(x_u)$, $\forall u \in U$. To prove dual feasibility, we need to prove that, for every (u, v), $\beta_v \geq \text{bid}_{uv}(1 - \alpha_u)$, where these are the values of the variables at the end of the algorithm. Suppose we allocated v to the bidder u'. When v arrives, let $\widehat{x}_{u'}$ and $\widehat{x}_u$ be the fraction of budgets spent by u' and u respectively. Let x_u be the fraction of budget spent by u at the end of the algorithm ($\widehat{x}_u \leq x_u$). We have the following sequence of inequalities.

$$\begin{aligned}
\beta_v &= \rho\ \text{bid}_{u'v} - \Delta_{u'v}(\widehat{x}_{u'}) \geq \rho\ \text{bid}_{uv} - \Delta_{uv}(\widehat{x}_u) \\
&= \text{bid}_{uv}(1 - a(\widehat{x}_u)) \geq \text{bid}_{uv}(1 - a(x_u)) \\
&= \text{bid}_{uv}(1 - \alpha_u)
\end{aligned}$$

The first equality is from the way the algorithm updates the β variables. The first inequality is because v was allocated to u' so it had the maximum value of the expression. The second equality is simply from the definition of $\Delta(\cdot)$ and $a(\cdot)$. The last inequality is because $a(\cdot)$ is an increasing function, and the last equality is from (6.5).

(2) Primal–Dual ratio. Suppose we allocate v to u. Then the primal objective goes up by bid_{uv}. The dual objective goes up by $\Delta_{uv}(x_u) + \beta_v = \rho\ \text{bid}_{uv}$. That is, for every $v \in V$, we split ρ times the increase in the primal objective function between the two dual variables. Therefore at the end of the algorithm, Primal $\geq \frac{1}{\rho}$Dual. □

6.2 Adwords with Random Order

In [78], the authors asked whether MSVV could be adapted to work optimally for Adwords with small bids in the IID model. They proposed "a concrete algorithm that works as follows: each bidder is assigned a weight, and his effective bid for a keyword is defined to be the product of the actual bid and his weight. The main question then is whether for any fixed distribution on queries there is always a set of weights such that the algorithm achieves $1 - o(1)$ expected competitive ratio." In [35], Devanur and Hayes answered this open question by showing that there always exists such a set of weights, in the random order model. They presented a $1 - \epsilon$ competitive algorithm that is also based on the Primal–Dual method (where the ϵ depends on the quantifying the small bids assumption).

The main idea behind the algorithm is to solve the allocation LP from Figure 6.1 on a sample of the queries. Note that we cannot expect to get a representative sample of all types of vertices from a small sample, that is, we cannot expect to estimate the distribution of bids. The insight provided in [35] is to use only the dual variables α_u and β_v from the sampled LP, which suffice to guide the allocation problem for the rest of the query stream (using the complementary slackness conditions). The authors draw an analogy to PAC learning, in that the initial query stream is used to learn from a small hypothesis class: not the distribution of bids, but only the set of dual variables of the bidders.

More formally, let $\widehat{D}$ be the sampled version of the dual program D of Figure 6.1, in the following sense: pick an $\epsilon > 0$. Let $\widehat{V} \subseteq V$ be the first ϵ fraction of the query stream (we assume that we know the length of the stream). Allocate $v \in \widehat{V}$ arbitrarily. For all u, replace B_u by ϵB_u. Include only the dual variables for $u \in U$ and $v \in \widehat{V}$, the constraints for (u, v) with $u \in U$, and $v \in \widehat{V}$.

Let $\widehat{\alpha}^*_u$, for $u \in U$ (and $\widehat{\beta}^*_v$ for $v \in \widehat{V}$) be the optimal solution to $\widehat{D}$. Now for the rest of the query stream $v \in V \backslash \widehat{V}$, allocate according the complementary slackness condition (6.1), using the $\widehat{\alpha}^*$.

Algorithm 12: Random Order Adwords (DH09)

Let $\widehat{V}$ be the first ϵ queries in V. Solve $\widehat{D}$.

When the next vertex $v \in V \backslash \widehat{V}$ arrives:

Match v to that available neighbor u which maximizes $\text{bid}_{uv}(1 - \widehat{\alpha}^*_u)$

Theorem 6.3 ([35]). Algorithm 12 (DH09) is $1 - \epsilon$ competitive in the case that

$$\frac{\text{OPT}}{\text{bid}_{\max}} \geq \Omega\left(\frac{n^2 \log \lambda/\epsilon}{\epsilon^3}\right)$$

where $\text{bid}_{\max}$ is the maximum bid over all edges, and λ is the ratio of the maximum to the minimum non-zero bid over all edges.

This technique of sampling the dual program according to the Random Order can be used in the practical setting by getting estimates from the data from the past. This technique was later generalized to general packing-covering LPs in [6, 44, 92].

6.3 Bipartite Matching via Randomized Primal–Dual

In Section 6.1 we saw how the MSVV algorithm can be obtained via the Primal–Dual framework, and how that gives a completely different proof. The question arises whether there is a similar technique

$$\max \sum_{(u,v)\in E} w_u x_{uv}$$

$$s.t.\ \forall\ u\ :\ \sum_{v:(u,v)\in E} x_{uv} \le 1$$

$$\forall\ v:\ \sum_{u:(u,v)\in E} x_{uv} \le 1$$

$$x_{uv} \ge 0,\ \forall\ (u,v) \in E$$

$$\min \sum_u \alpha_u + \sum_v \beta_v$$

$$s.t.\ \forall\ (u,v) \in E:\ \ \alpha_u + \beta_v\ \ge\ w_u$$

$$\alpha_u \ge 0\ \ \forall u$$

$$\beta_v \ge 0\ \ \forall v$$

Fig. 6.2 The Primal–Dual pair of LPs for the fractional version of the allocation problem in vertex-weighted bipartite matching.

to re-interpret Ranking for bipartite matching, and (its generalization) Perturbed Greedy for vertex-weighted bipartite matching? One may try to use Algorithm 11 directly, but the obstacle is that the updates to the variables would not be "smooth", since the fraction of budget spent in these problems is either 0 or 1 (for example, recall that we crucially used the small bids assumption to convert the cumulative sum into an integral in Equation (6.5)). Indeed, a more basic question is how to introduce randomization (as required by Ranking) in Primal–Dual update algorithms such as Algorithm 11. In [33], the authors provide a very elegant idea to side-step the smoothness problem and introduce randomization by interpreting Ranking and Perturbed Greedy as randomized dual update algorithms.

The Primal–Dual LP formulation for vertex-weighted bipartite matching is the same as in Figure 6.1, with the following setting of the constants: $\forall\ u,v:\ B_u = w_u,\ \forall\ (u,v) \in E : \text{bid}_{uv} = w_u$. We repeat the formulation in Figure 6.2.

The algorithm starts by picking, for each $u \in U$, a random number r_u IID from the uniform distribution on $[0,1]$.

Now, the only structural difference from Algorithm 11 is the definition of Δ_{uv}, which we define here as:

$$\Delta_u(r_u) = w_u \rho e^{r_u - 1}$$

Note that, unlike in Algorithm 11, Δ_u depends on the random variable r_u, rather than the spend parameter x, and it is not defined with respect

to any particular v. With this difference, the algorithm can be defined *identically* to Algorithm 11. We repeat it below:

Algorithm 13: Primal–Dual Vertex-Weighted Bipartite Matching (DJK11)

For each $u \in U$, pick a random number r_u IID from the uniform distribution on $[0,1]$. Define $\Delta_u(r_u) = w_u \rho e^{r_u - 1}$.
Initialize: $\alpha_u = 0 \ \forall \ u, \beta_v = 0 \ \forall \ v$
When the next vertex $v \in V$ arrives:
  If v has no available neighbors, continue.
  Match v to that available neighbor u
   which maximizes $\rho \ w_u - \Delta_u(r_u)$
  Update: $\alpha_u = \Delta_u(r_u)$
    $\beta_v = \rho \ w_u - \Delta_u(r_u)$

Remark. Note that this algorithm is identical to Perturbed Greedy (and to Ranking for equal weights), since the vertex is matched to the available neighbor u maximizing $\rho w_u - \Delta_u(r_u) = \rho w_u(1 - e^{r_u - 1}) \ \propto \ w_u \psi(r_u)$.

As before, to prove that this algorithm achieves $1 - \frac{1}{e}$, we need to prove the competitive ratio and feasibility.

Primal-Dual ratio. This remains easy, since the primal increment on allocating v to u is w_u, while the dual increment is $\alpha_u + \beta_v = \Delta_u(r_u) + (\rho w_u - \Delta_u(r_u)) = \rho w_u$. Again, ρ times the primal increase is split between the two dual variables, this time according to a random split.

Feasibility. The primal solution is feasible by construction. For dual feasibility, we need to prove, for every $(u,v) \in E$, that $\alpha_u + \beta_v \geq w_u$.

Fix a choice of the r_u. Recall that the algorithm matches an arriving vertex v to the available neighbor u^* which maximizes the value of $\rho w_u - \Delta_u(r_u)$. It then sets β_v to $\rho w_{u^*} - \Delta_{u^*}(r_{u^*})$ and α_{u^*} to $\Delta_{u^*}(r_{u^*})$. Note that the constraint for (u^*, v) is automatically satisfied. However, a moment's thought shows that the constraint may not be satisfied for other $(u,v) \in E$. For example, consider a $u \neq u^*$ s.t. $(u,v) \in E$ and suppose that u was available when v arrived. We know that $\beta_v \geq \rho w_u -$

$\Delta_u(r_u)$. This was sufficient for the proof of Algorithm 11, but it does not suffice here. For instance, u may never be matched during the algorithm, so $\alpha_u = 0$ even at the end, and $\alpha_u + \beta_v$ could be as small as $0 + \rho w_u - \Delta_u(r_u) = \rho w_u(1 - e^{r_u - 1})$. The latter is strictly less than w_u for all $r_u > 0$. On the other hand, if u does get matched later, then $\alpha_u = \Delta_u(r_u)$, and the constraint would be satisfied.

This problem arises precisely because the lower bound on β_v of $\rho w_u - \Delta_u(r_u)$ is useful only in the case when u does get matched later and we get a corresponding increment in α_u. This problem did not occur for the Adwords problem (Algorithm 11) because in that algorithm, α_u changes very little in every step, so even if the bidder u were not to be allocated ever again after v's arrival, the current value of α_u is good enough to make the constraint satisfied together with the lower bound on β_v.

Thus the dual feasibility constraints may not actually hold for every choice of the r_u. In [33], the authors prove (using proof techniques from the original proof for RANKING [63]) that they hold *in expectation* over the choices of r_u. There are two cases to consider.

The first case is the one we considered above, in which v was matched to u^* although u was still available at that time. Let τ be the highest value of r_u such that v still prefers to match to u^* instead of u. One can prove (using an alternating patch argument, similar to the proof of Theorem 3.3), that for any value of r_u, v is still matched, so

$$\beta_v \geq \rho w_u - \Delta_u(\tau)$$

So the expected value of β_v is also at least this value. Furthermore, we can prove that for all $r_u < \tau$, u is matched, and therefore $\alpha_u = \Delta(r_u)$. Thus, in expectation,

$$\alpha_u \geq \int_{r_u=0}^{\tau} \Delta(r_u) \mathrm{d}r_u = \Delta_u(\tau) - \Delta_u(0)$$

Thus

$$E[\beta_v + \alpha_u] \geq \rho w_u + \Delta_u(0) = \rho w_u(1 - e^{-1}) = w_u$$

which proves that the constraint holds in expectation.

The second case is when u is always matched, for all r_u. In this case we simply see that

$$E[\alpha_u] = \int_{r_u=0}^{1} \Delta_u(r_u) \mathrm{d}r_u$$

which is precisely w_u.

Thus we have the Primal–Dual ratio of ρ in each run and feasibility in expectation. This suffices to prove the competitive ratio of $1 - \frac{1}{e}$ in expectation.

Open Question 8. We noted that Algorithms 11 and 13 are identical in their structure. Find a hybrid of the two algorithms and their proofs which provides a non-trivial competitive ratio for a more general problem. For example, prove a ratio greater than $\frac{1}{2}$ for the Adwords problem with general bids.

7

Display Ads

The Display Ads problem is the edge-weighted and capacitated generalization of online bipartite matching. In this problem, the edges of the graph have weights w_{uv}, and the vertices $u \in U$ have capacities c_u. As before, when a vertex in V arrives, it has to be matched to a neighbor in U, such that each $u \in U$ is matched at most c_u times. The goal is to maximize the total weight of the matched edges. This is clearly a generalization of online bipartite matching and vertex-weighted bipartite matching, but is not comparable to the Adwords problem.

The first observation for this model is that even for a simple star graph (that is, when $U = \{u\}$, $(u, v) \in E, \forall\ v \in V$, and $c_u = 1$), it is not possible to obtain any non-trivial competitive ratio. This is because the problem is identical to picking the maximum from a stream of numbers which arrive online. Consider any deterministic algorithm: as soon as the algorithm chooses a number, the adversary can create the arrival of a much bigger number. A similar argument can be made for randomized algorithms as well (see [5] for one proof).

To sidestep this impossibility we have to look beyond the adversarial order. There are two ways to do this, first in the Random Order input model, and second in a new input model called the *free disposal* model.

7.1 Random Order and Secretary Algorithms

In this section, we study the Display Ads problem in the Random Order model, for the case of matching, that is, $\forall\, u, c_u = 1$. When $|U| = 1$, this is the classic Secretary Problem [41], for which the optimal algorithm is as follows (recall that the Random Order model assumes that we know $|V|$, the number of vertices to arrive).

Algorithm 14: Classic Secretary

Phase 1: Reject the first $|V|/e$ of the sequence. Let m be the maximum of these numbers.

Phase 2: Pick the first number which arrives which is larger than m. (If no such number arrives then declare failure).

Theorem 7.1 ([70, 41]). Algorithm Classic Secretary picks the highest number with probability at least $\frac{1}{e}$, that is, its competitive ratio is $\frac{1}{e}$.

Proof. (intuition) We present the intuition for why this strategy works. First, note that the algorithm succeeds in picking the highest number if the following event occurs: the second highest number occurs in the first phase, and the highest occurs in the second phase. The probability of this event is $\frac{1}{e}\left(1 - \frac{1}{e}\right) = \frac{e-1}{e^2} \simeq 0.23$. This already proves a non-trivial and constant ratio. Note that there are other events in which the algorithm succeeds, for example, when the third highest number is in Phase 1, the first and second highest are in Phase 2, but the first highest arrives before the second highest. These events increase the probability of success to $\frac{1}{e}$. □

In [67], Korula and Pal study the matching problem, and analyze a natural generalization of Algorithm 14. This algorithm was first introduced by Dimitrov and Plaxton [38] for the secretary problem on transversal matroids, for which they proved a ratio of 16 (this is a special case of our problem, in which, for every $v \in V$, all the edges incident on v have the same weight). The generalization of the *Classic Secretary* algorithm to matching is as follows:

Algorithm 15: Secretary Matching (DP08, KP09)

Phase 1:
– Do not match any vertex in the first half of the vertex sequence (call this set V_1).
– Compute a greedy maximal weight matching $\mathcal{M}_1$ on the graph formed between U and V_1.
– $\forall\ u \in U$: Let t_u be the value of the edge in $\mathcal{M}_1$ incident on u (0 if there is no such edge).

Phase 2: For each subsequent v, match it to that neighbor u which has the highest weight among those with $w_{uv} \geq t_u$ (if any).

Theorem 7.2 ([67]). Algorithm 15 (Secretary Matching KP09) has a competitive ratio of at least $\frac{1}{8}$.

Intuitively, one can think of the algorithm losing a ratio of $\frac{1}{2}$ three times: first in discarding half the arrival sequence, second in using a greedy matching for the estimates of the threshold weights t_u, and finally, due to the online nature of the second half of the sequence.

Recently Kesselheim et al. [64] significantly improved this result, by providing a different algorithm which achieves the optimal ratio of $\frac{1}{e}$, thus extending the classic secretary result to the matching setting. Their algorithm does not explicitly compute thresholds for the vertices in U, but updates the solution using locally optimal matchings at each step.

Algorithm 16: Secretary Matching (KRTV13)

- Do not match the first $\frac{1}{e}$ fraction of the arriving vertices. Call this set V'
- Set the solution $M = \emptyset$.
- For each subsequent $v \in V \backslash V'$:
  - Set $V' = V' \cup v$
  - Find the optimal matching M' on the currently revealed graph $G(U, V', E')$
  - If v was matched to u in M', and if u is available in M, then add (u, v) to M.

Theorem 7.3 ([64]). Algorithm 16 (Secretary Matching KRTV13) has a competitive ratio of $\frac{1}{e}$ (and this is optimal).

7.2 Adversarial Order and the Free-Disposal Model

In [45], Feldman et al. introduced the free-disposal model which provided a new approach for overcoming the impossibility of a non-trivial competitive ratio in the adversarial model. In this model, a vertex $u \in U$ is allowed to be matched more times than its capacity c_u, but the algorithm makes a gain only for the c_u highest weight edges matched to u. More precisely, if the algorithm allocates the set of vertices $V_u \subseteq V$ to u, then, the contribution of a vertex u to the objective function is

$$f_u(V_u) = \max_{S \subseteq V_u,\ |S| \leq c_u} \sum_{v \in S} w_{uv}$$

and the gain of the algorithm, as usual, is $\sum_{u \in U} f_u(V_u)$.

It can be verified that this objective function is submodular. One can immediately see how this model overcomes the example of the star graph for the adversarial order described in the beginning of this section. An algorithm can now pick an edge whose weight is higher than that of the previously chosen best edge. This would achieve a ratio of 1 (indeed an algorithm can pick *all* the edges in the star graph example).

The motivation for this model comes from the Display Ads setting: in the contract between advertisers and the publisher, the publisher guarantees a certain number of ad impressions to each advertiser for a negotiated price; however, the advertiser would not be unhappy if it gets more impressions, as long as it does not pay extra. The advertiser's utility can be taken to be the best set of ads within capacity. In [45], the authors provided an algorithm for this problem, stated below; in fact, it is a family of algorithms, based on a choice of a bid-scaling rule β_u.

Algorithm 17: Free Disposal (FKMMP09)

When the next vertex $v \in V$ arrives:

For each $u \in U$, let β_u be the value of the bid-scaling rule.

Allocate v to that u which maximizes $w_{uv} - \beta_u$.

Note how this algorithm involves additive bid-scaling, as opposed to the multiplicative bid-scaling that we have seen in the previous sections (β_u is typically independent of w_{uv}).

At any time during the algorithm, let V_u be the set of vertices already allocated to u. Let $n_u = |V_u|$. Order V_u in decreasing order, and define w_j as the weight of the jth edge in this order. The following setting of β_u corresponds to the GREEDY algorithm, which maximizes the marginal gain in the current step.

$$\beta_u = \begin{cases} 0, & \text{if } n_u < c_u \\ w_{c_u}, & \text{if } n_u \geq c_u \end{cases}$$

In the case of matching ($c_u = 1,\ \forall u \in U$), β_u is simply the weight of the edge previously matched to u, which we can then dispose of if we match the new vertex to u. In [45], the authors noted that since the utility function of free-disposal model is submodular, GREEDY achieves a ratio of $\frac{1}{2}$ (Theorem 8.1).

In [45], the authors described an exponential weighing scheme for β, which achieves a ratio better than $\frac{1}{2}$ for large capacities c_u. This uses the following bid-scaling function β.

$$\beta_u := \frac{1}{n_u\left(\left(1+\frac{1}{n_u}\right)^{n_u} - 1\right)} \sum_{j=1}^{n_u} w_j \left(1 + \frac{1}{n_u}\right)^{j-1}$$

Theorem 7.4 ([45]). Algorithm 17 (Free Disposal FKMMP09) with the exponential weighting rule achieves a ratio of $1 - \frac{1}{e}$ as $c_u \to \infty$, $\forall\ u \in U$.

The proof of this theorem is a generalization of the Primal–Dual technique shown in Section 6.1 for the Adwords problem. In [45], the result was generalized to the bigger class of problems known as Generalized Assignment Problems (GAP); see Section 2 for the definition.

Open Question 9. Find an algorithm which achieves a ratio better than $\frac{1}{2}$ for the Display Ads problem in the free-disposal model with small capacities (for example, when all capacities are equal to 1), or prove an upper bound of $\frac{1}{2}$.

8

Online Submodular Welfare Maximization

As mentioned in Section 1, all the matching and allocation problems studied in this survey can be restated as online welfare maximization problems. In this setting, there is a universe of elements V, and a set U of n players. Player u has a valuation function $f_u : 2^V \to \mathbb{R}^+$. Elements in V arrive online, and have to be allocated (irrevocably) to a player in U as they arrive. At the end of the input stream, the algorithm would have allocated a set $V_u \subseteq V$ to u. The goal of the algorithm is to maximize $\sum_{u \in U} f_u(V_u)$.

A function $f: 2^V \to \mathbb{R}^+$ is called monotone if $\forall\ S \subseteq T \subseteq V,\ f(S) \leq f(T)$. The function f is called submodular if for any two sets X and Y: $f(X \cup Y) + f(X \cap Y) \leq f(X) + f(Y)$. This is also equivalent to the following: $\forall\ S \subseteq T \subseteq V,\ x \notin T:\ f(S + x) - f(S) \geq f(T + x) - f(T)$.

A generalization of all the problems we have studied is the online submodular welfare maximization problem in which the f_u are all monotone submodular functions (in Section 2, we defined the valuation functions corresponding to the other problems, which shows that they are submodular). The simplest algorithm for this problem is Greedy, which allocates the next item $v \in V$ to that $u \in U$ for which the *marginal gain* is the highest:

Algorithm 18: GREEDY for Submodular (LLN06)

When the next vertex $v \in V$ arrives:
For each $u \in U$, let V_u denote the set of vertices in V already allocated to u.
Allocate v to that u which maximizes $f_u(V_u \cup \{v\}) - f_u(V_u)$.

Lehmann et al. [69] proved the following theorem (which generalizes all the results for the greedy algorithms in the previous sections).

Theorem 8.1 ([69]). GREEDY achieves a ratio of $\frac{1}{2}$ for the online submodular welfare maximization problem in the adversarial model.

The proof of this theorem is a generalization of the proof that GREEDY achieves a ratio of $\frac{1}{2}$ for the Adwords problem, even with large bids (Theorem 5.1). We omit the proof here.

The obvious question is: can we do better than $\frac{1}{2}$, as we managed for the special cases in the earlier sections? Given that there exist bid-scaling algorithms for these special cases, one may expect a generalization to the submodular case using some sort of a randomized bid-scaling algorithm. Indeed the algorithms in [78] and [5] follow the same intuition of creating a hybrid between two extreme algorithms, by scaling the bids. Similarly, the Primal–Dual proofs in [21] and [33] also follow an identical schema. One may expect to design a hybrid of the two bid-scaling algorithms — randomized scaling for the vertex-weighted bipartite matching problem, and deterministic scaling for the Adwords problem with small bids — which can achieve a ratio better than $\frac{1}{2}$ for this problem.

However, in a recent paper [59], Kapralov et al. show that such an algorithm does not exist (under reasonable complexity assumptions). In particular they show the following:

Theorem 8.2 ([59]). There is no (randomized) algorithm which can achieve a competitive ratio of better than $\frac{1}{2}$ for the online submodular welfare maximization problem, unless NP = RP.

Open Question 10. This result eliminates the possibility of generalizing the techniques in [5, 78] (or in [21, 33]) to submodular functions. However, the possibility of having such a generalization still remains an interesting open question for the Adwords problem without the small bids assumption, for the Display Ads problem with general (small) capacities in the free-disposal model, or for other special submodular functions.

Open Question 11. Find an upper bound for the online submodular welfare maximization problem without making a complexity assumption.

In [59], the authors further study the online submodular welfare maximization problem in the Unknown IID model. For this model, one has to allow vertices in U to be allocated multiple copies of vertices in V. Thus, we need to first generalize submodular functions to multisets. In [59], the notion of valuations with diminishing returns is introduced, which is a natural generalization of submodular functions to multisets: adding a new item to a multiset S provides at least as much marginal gain as adding the item to a super-multiset. They prove the following generalization of Theorem 5.5.

Theorem 8.3 ([59]). GREEDY achieves a ratio of $1 - \frac{1}{e}$ for the online welfare maximization for valuations with diminishing returns in the Unknown IID model. Unless NP = RP, no algorithm can do better, even for coverage valuations.

9

Applications

In the previous sections, we have seen the richness of the problem landscape and of the theoretical techniques in online matching. In this section we will see a glimpse of how these algorithmic ideas can be applied in practice. We will start, in Section 9.1, by taking a unified, high-level view of all the algorithms we have studied so far, as *bid-scaling* algorithms, and note some properties of these algorithms that are useful in practice. In Section 9.2, we will note that, in practice, one needs to consider several different objective functions, corresponding to the various participants in the ad ecosystem. We will define some useful metrics to capture these objective functions. In Section 9.3, we will give an overview of results published from the Industry, all of which use some variant of the bid-scaling algorithms. In Section 9.4, we will briefly describe a feedback-loop based heuristic, inspired by the bid-scaling algorithms, to capture situations in which the distribution of the request sequence changes over time. Finally, in Section 9.5, we will describe *throttling-based* algorithms for ad allocations. This is an algorithmic framework related to, but different from bid-scaling.

9.1 A Unified View: Bid-Scaling Algorithms

At a high level, all the algorithms that we have seen so far (except for the secretary algorithms from Section 7.1) can be considered to be members of a larger class of *Bid-Scaling Algorithms*: In this class of algorithms, instead of making a greedy choice by picking the highest weight edge, the algorithm first scales down the weights by some scaling factor and then picks the highest. The form of the scaling function depends on the particular problem setting: the scaling can be deterministic (MSVV, BJN07, DH09 and variants) or randomized (Ranking and Perturbed Greedy); it can be a function of the current utilization of the budget (MSVV and BJN07) or the utilization of the capacity (FKMMP09), or a function of a carefully chosen random variable (Perturbed Greedy); it can be multiplicative (MSVV, BJN07, DH09) or additive (FKMMP09).

It is important to note that the optimal online algorithms take a very simple form, not too different from Greedy itself. This makes this class of algorithms useful in practice. While Greedy needs zero state in memory, bid-scaling algorithms also need very little state to be carried around, at the most one number per bidder. Oftentimes in practice, the parameters of the bid-scaling function, for example, the budget utilization, are already available at run-time. The algorithms are very lightweight at run time, with small memory footprint (essentially one extra number per advertiser) and CPU time (one arithmetic operation to scale the bids). They also do not require much communication between servers, only updates of the bid-scaling numbers (or equivalently, information such as the budget utilization), which can be updated at regular intervals, and are typically already available in existing systems.

Bid-scaling has become a well-understood scheme for ad allocation, and has been applied successfully in ad systems in major Internet Search, Display and Mobile ad companies, as evidenced by the published literature (surveyed in Section 9.3). We can classify bid-scaling algorithms in two categories:

(a) based on the adversarial order assumption,
(b) based on some assumption on the distribution of the query sequence.

The latter are based on the sampled LP technique described in Section 6.2 in which we compute the dual variable values for the sampled LP, and thereby obtain a near optimal solution for the entire problem, provided that the distribution of queries does not change over time. Clearly, the adversarial algorithms have two advantages: (1) they are much easier to implement, since they do not require solving any LPs, and (2) they guarantee a certain performance even if the query sequence does not follow the estimated distribution. Furthermore, some of these algorithms perform much better than their worst-case guarantees if the queries do come from a static distribution. On the other hand, the sampling based techniques provide much better guarantees if we have a good estimate of the distribution of the queries.

9.2 Objective Functions Used in Practice

In the theoretical analysis in the previous sections, our objective has always been the *efficiency* of the allocation, that is, we wish to maximize the total weight of matched edges. Efficiency captures the total value to advertisers and the platform (since the efficiency is shared between the two via payments, as determined by the pricing mechanism). In practice, we need to look at more detailed objective functions, corresponding to the three different participants in the ecosystem: the quality of ads shown to the users, the Return on Investment (ROI) to advertisers and the short-term revenue to the ad platform. Depending on the information available, we can formulate different proxies for these abstract objective functions:

- **Quality.** If we have access to a quality score q_i of an ad impression i, then an obvious metric is the average quality of ads shown, $\sum_{i \in I} q_i/|I|$, where I is the set of ad impressions shown. One may try to formulate different ways of defining the quality scores q_i. A simple method if to use the click-through rate of the impression (CTR_i) as a proxy for the quality score. CTR_i is the probability that the user will click on the ad impression i (when shown for a given query at a given position on the page), as predicted by the platform's machine learning algorithms.

- **Return on Investment (ROI).** If we have an estimate of the conversion-rate (CVR_i) which is the probability with which a user "converts", that is, makes a purchase after clicking on the ad impression i, and if we have an estimate of the value V_i of the purchase to the advertiser, then ROI can be defined as $\frac{\sum_{i \in I} \text{CTR}_i \text{CVR}_i V_i}{\sum_{i \in I} \text{CTR}_i \text{CPC}_i}$. Here I is the set of impressions shown for this advertiser, and CPC_i is the cost per click for this impression. The numerator is the expected value to the advertiser from the shown ad impressions and the denominator is the expected spend. Thus, this is the value to the advertiser per dollar spent. One may make simplifying assumptions on CVR or value in the absence of that data. Another proxy for ROI can be profit per dollar spent, $\frac{\sum_{i \in I} \text{CTR}_i \text{CVR}_i (V_i - \text{CPC}_i)}{\sum_{i \in I} \text{CTR}_i \text{CPC}_i}$.
- **Revenue.** The short-term revenue is simply $\sum_{i \in I} \text{CTR}_i \text{CPC}_i$.

There are other objective functions, such as *fairness* to different advertisers, which are also important. In the remainder of this section, we describe some of the work published from the industry which describes the use of the algorithms from this survey, or their variants. Some of these papers work with high level objective functions, such as the efficiency, while some look at the detailed objective functions mentioned above.

Open Question 12. Formulate an online matching problem using these detailed objective functions, perhaps in a multi-objective setting (see Open Question 18), and find optimal algorithms.

9.3 Published Results from the Industry

Here, we survey some of the published results which apply bid-scaling algorithms in practice. Much of the published work is for the domain of Display Ads. In this setting, advertisers and publishers sign contracts of the following form: The publisher guarantees to deliver X ad impressions for the advertiser based on the advertiser's targeting criteria

(website targeting, demographic targeting, geographic targeting, etc.). The advertiser promises to pay the publisher Y dollars in exchange. The risk in this case, is borne by the publisher, that is, if it fails to deliver the promised number of impressions, then there would typically be some re-compensation. Clearly, this is a matching problem as in the Display Ads problem of Section 7, although with *lower bound* constraints on capacity.

In a paper from Google [44], the authors describe how the bid-scaling algorithms from Section 7, as well as those based on the sampled-LP technique (introduced in [35], and generalized in [6, 36, 44]) can be applied in the Display Ads system to achieve gains over the simple greedy allocation. In particular, they simulate these algorithms over real data, where the edge weights are taken to be the "quality-scores" of an advertiser for an ad-slot. The experimental results show that the bid-scaling algorithms from Section 7 improve the efficiency of the allocation over Greedy by 10 or more percentage points, and the LP sampling based algorithms improve even further. The upper bound for comparison is provided by the offline optimum which is the solution to the corresponding allocation LP. This paper also introduces the notion of *fairness* of an allocation as another objective function. Intuitively, the idea is that maximizing the sum of utilities may end up giving a certain advertiser a very bad set of impressions. They capture this notion via a fairness metric, and provide experimental evaluation of the adversarial and sampled LP bid-scaling algorithms for the fairness metric. Refer to [44] for the detailed experimental results.

In a paper from Microsoft [26], the authors describe an algorithm to satisfy the capacity constraints for advertisers in a Display Ads setting. As opposed to the problem described in Section 7, here the capacity constraints are lower bounds, representing the demand of an advertiser. In [26], the authors define the WaterLevel algorithm (which, as we saw in Section 6.2, generalizes Balance to the case of unequal bids) and describe how it can be used for this problem (see [26, 36, 34] for the technical details). In a different paper from Microsoft [29], the authors show how the Primal–Dual based algorithm for Adwords (Sections 6.1 and 6.2) can be adapted for performance-based display advertising.

They also show a control-loop based heuristic based on the algorithm, and provide experimental evaluation, which show significant gains for these algorithms.

In a paper from Yahoo! [92], the authors describe an algorithmic framework for the Display Ads problem. The main idea is along the lines of the Primal–Dual algorithms of Section 6.2, although there are some differences. The main components of this system include: (a) a convex programming formulation for the allocation problem with a general convex objective function, (b) showing how KKT conditions provide a way to use the dual solution to obtain the optimal primal solution, and (c) a method to sample the input in a biased manner so that small advertisers are not forgotten in the optimization. (a) and (b) generalize the techniques of the previous sections to convex programs (see also [32] for another convex program setting). They name their solution a Compact Allocation Plan, since (as with all bid-scaling algorithms) it only needs to store one dual variable per advertiser.

9.4 Adaptive Bid-scaling Heuristics

As we noted in Section 9.1, the two types of bid-scaling algorithms that we have seen, adversarial and distribution based, make different assumptions on the statistics of the query sequence. The former make no assumptions and provide worst case guarantees. The latter assume that the distribution is static, so an initial sample of the sequence (or a sample from previous days' logs) is representative of the entire sequence. However, the query stream over a day may not be uniform, that is, the distribution may change over time. For example, there may be bursty, unpredictable traffic, due to unforeseeable events. More routinely, there are a large number of time based effects, namely time-of-day, day-of-week, seasonality, holidays, etc. Each of these can be captured in theory, but it is difficult to capture them all.

One approach is to use the distribution based techniques, but resample regularly and recompute the sampled LP's dual variables. Another approach would be to simply use the algorithms designed for the adversarial model (recall that these often provide hybrid guarantees, in the worst and IID case [26, 79]), or hybrid algorithms which use sampled

variables as well as the worst-case bid-scaling formulae (for example, [72]). A third approach may be to introduce a time-varying distributional input model (see [7, 37] for some approaches along this direction).

From a practical viewpoint we may ask if there are any simple heuristics that we can extract from the intuition of the algorithms which could work well in dynamically changing environments. In fact, there is a simple and practical control scheme which updates the bid-scaling numbers via a feedback loop, which was introduced in the paper on Adwords [78], which we quote below. The authors first ask if the guarantee of the MSVV algorithm can be improved upon if there is a static distribution of queries:

"...suppose the queries are chosen from an arbitrary but fixed probability distribution. Is there a variant of the basic algorithm given in this paper that achieves a $1 - o(1)$ expected competitive ratio in this setting, while still providing the $1 - \frac{1}{e}$ worst case guarantee? Such an algorithm would be very desirable, since it is not unreasonable to expect the queries to be drawn from a fixed distribution for some time window, and for this distribution to either drift over time or to make sudden jumps (triggered by external events/time of day). We propose a concrete algorithm that works as follows: each bidder is assigned a weight, and his effective bid for a keyword is defined to be the product of the actual bid and his weight. The main question then is whether for any fixed distribution on queries there is always a set of weights such that the algorithm achieves $1 - o(1)$ expected competitive ratio." Note that for a fixed distribution, the algorithm of [35] (and the generalizations in [6, 44]) provides precisely such a guarantee.

Continuing further, the authors consider the case when the distribution is not static, and provide a simple feedback loop based control scheme to update the weights as the distributions change: "The following online heuristic might provide a quick way of computing such weights: consider the allocation of queries for some window of time under the current weights. Adjust the weight of a bidder upwards if that bidder spends less than his fair share of his budget during this time window, and downwards if he spends more than his fair share of the budget. Repeat this process after each such window of time." This is a simple heuristic that captures the intuition behind the MSVV and related

algorithms, and adapts naturally to changing distributions. Note that a control algorithm along these lines has been reported in the experimental literature in [29], which was described in Section 9.3.

9.5 Throttling

Throttling is a scheme orthogonal to bid-scaling, described in two papers, out of Google [60] and Microsoft [25]. We will first follow the former setup, before describing the results from the latter.

In the throttling setting, as usual, there is a bipartite graph $G(U,V,E)$, with advertisers U, advertiser budgets B_u for $u \in U$, and a query sequence of the vertices V. The allocation problem is to determine, for each $v \in V$, the advertisers whose ads are to be shown for v. In the previous sections, we determined this by scaling the bids and then choosing the ads with the highest scaled bids — that is, the ads participate in the auction for v with their scaled bids. In the throttling setting, we do not modify the bids, but explicitly allow or disallow participation in the auction[1] for v. Thus for each v, throttling determines, for each $u \in U$, whether it can participate in the auction for v or not (in which case it is said to be throttled).

Throttling was first introduced as a simple engineering scheme to ensure two goals: (1) each advertiser spends at most its budget, and (2) it spends its budget in a smooth manner. This was achieved by throttling each advertiser u perfectly randomly: for each v, throttle u independently with probability p_u, which is carefully chosen so that u spends exactly its budget at the end of the day. In [60], the authors introduce the notion of *optimized throttling*, which throttles advertisers in a non-uniform manner so as to maximize a chosen objective function.

A new input model: In [60], the authors introduce a new stochastic input model for the Adwords problem. Instead of assuming that queries are picked IID from a distribution, this model makes the following weaker assumption. Choose an auction-time per-advertiser metric $\mu_u(v)$ (different choices of μ will result in the optimization of different

[1] This is equivalent to allowing bid-scaling by 0 or 1 only.

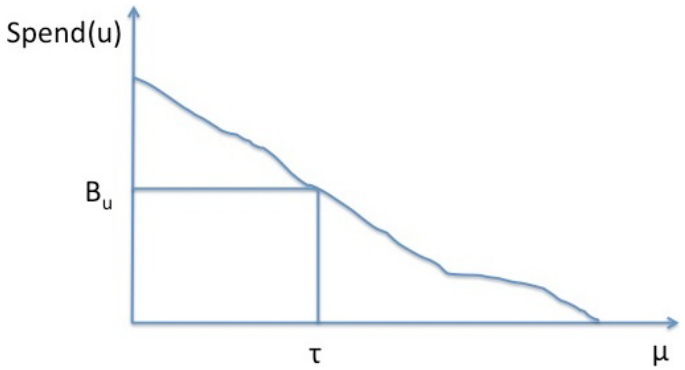

Fig. 9.1 The estimated distribution of spend according to the metric μ. τ is the threshold for the algorithm.

objectives; for concreteness, take this to be the click-through rate for an impression of u's ad on the query v). The stochastic assumption is that, for each $u \in U$, the algorithm knows the distribution of μ_u. More precisely, for $x \in \mathbb{R}$, let $S_u(x)$ be the expected spend of advertiser u on queries $v \in V$, s.t. $\mu_u(v) \geq x$. The assumption is that the algorithm knows S_u. For example, taking μ to be the CTR, this says the algorithm knows the expected spend of u on queries in which its CTR is at least x. In other words, it has an estimate of u's spend sliced by CTR. Figure 9.1 depicts the estimated distribution.

The allocation problem is to determine the subset $V_u \subseteq V$ of queries for which we allow the advertiser's ad to participate in the auction (that is, not throttle), so as to (1) satisfy the budget constraints and (2) optimize the chosen objective function.[2]

Online Knapsack Problem with estimates: The main observation is that with the above distributional assumption, we can formulate optimize throttling for an advertiser as an *Online Knapsack Problem* in which we have an estimate on the value and size of each item. To see this, let us fix an objective function: maximize the value that u gets

[2] By taking a hybrid of such an optimized throttling scheme with the random throttling scheme (for example, by randomly choosing, for each query, which throttling scheme to follow), we can achieve an improvement in the objective function while spending the budgets at a reasonably smooth rate as well.

(that is, its ROI). This becomes the following knapsack problem:

$$\max_{V_u \subseteq V} \sum_{v \in V_u} \nu_u \mathrm{CTR}(u,v)$$

$$\text{s.t.} \quad \sum_{v \in V_u} \mathrm{CTR}(u,v)\mathrm{CPC}(u,v) \leq B_u$$

Here, ν_u is u's value-per-click,[3] $\mathrm{CTR}(u,v)$ is the click-through rate, and $\mathrm{CPC}(u,v)$ is the cost-per-click for u's ad on v.

The offline greedy approximate solution for the knapsack problem is to allow u to participate for those v which maximize the ratio of value to size, that is, $\nu_u/\mathrm{CPC}(u,v)$ which is proportional to $1/\mathrm{CPC}(u,v)$ (since the value-per-conversion ν_u is assumed to be fixed for a given advertiser u). This is equivalent to picking a threshold τ and throttle u whenever $\mathrm{CPC}(u,v) > \tau$; the threshold τ should be chosen so that this scheme would spend the entire budget B_u.

Now note that, with our distributional assumption, this offline algorithm can be implemented online as follows:

Algorithm 19: Optimized Throttling for advertiser value

Offline Step:
– Take $\mu_u(v) = \mathrm{CPC}(u,v)$.
– Estimate the spend distribution of u according to μ.
– Estimate the threshold τ from the distribution.
When the next vertex $v \in V$ arrives:
– If $\mathrm{CPC}(u,v) \leq \tau$, allow u to participate in the auction for v.
Else throttle u.

Figure 9.1 shows how we choose τ. Note that the objective function of maximizing clicks for u led us to use $\mu = \mathrm{CPC}$; another objective function, for example, maximizing the quality of ads shown to the users will naturally lead to other metrics.

Optimizing for all advertisers simultaneously: We saw how a single advertiser's budget throttling problem is a knapsack problem.

[3] The advertiser really gets a value on a sale. This can be incorporated into the knapsack problem by estimating the conversion-rate of the ad, $\mathrm{CVR}(u,v)$ (that is, probability of a sale given a click)

However, when we try to apply this to optimize for all advertisers together, then a new problem gets introduced. Because advertisers participate in the same auctions, their distributions are not really independent. In fact, any throttling decisions made for one advertiser u will necessarily affect the distribution of another advertiser u' who targets the same auctions. This is due to two different types of interactions: if we show an impression for u, then this can push an ad for u' further down the page, thus reducing its CTR; secondly, in a second-price auction, an ad for u can set the CPC of the ad for u'. In [60], the authors suggest a heuristic to incorporate these interactions in the decision: the idea is to iterate over the advertisers in each auction, making the decision whether to throttle an ad or not based on the decisions for the other advertisers. The same heuristic is used when estimating the distributions. Another issue, when optimizing for multiple objectives is fairness among the advertisers, which is described below.

In [60], the authors compare different optimized throttling schemes to a global optimum obtained by solving a large configuration LP, as described in [2]. They simulate these schemes on experimental data, and show how throttling can be used to maximize ROI for advertisers as well as the average quality of ads.

In [25], the authors also study optimized throttling for the ROI objective. They simulate an optimized throttling scheme and demonstrate significant gains in the objective function. The main theoretical result in the paper is on fairness while optimizing for multiple advertisers.

Best-effort/regret-free throttling: One effect which arises when optimizing for multiple advertisers is that a global optimum (say, to maximize the total number of clicks) may not be fair, in the sense that it may maximize the total number of clicks by increasing the clicks for one set of advertisers at the expense of another. Both [60] and [25] introduce a game-theoretic notion of fairness, called best-effort optimization in [60] and regret-free optimization in [25]. A throttling based allocation is called best-effort or regret-free, if, for each $u \in U$, given the throttling decisions for all $u' \neq u$, the allocation makes u

participate in the best subset V_u, while keeping the spend of u at most B_u. Such an allocation would essentially be an equilibrium in that we optimize for each advertiser, given the choices we made for the others. In [25], the authors prove that a regret-free randomized throttling-based allocation exists.

10

Related Models and Future Directions

In the final section, we briefly describe some related models, without going into the details. The aim is to provide a sense of the richness of the problem domain even outside the landscape described in the previous sections. We end with a (necessarily partial) list of open problems of interest.

10.1 Related Models

10.1.1 Matching with Stochastic Rewards

In many applications, including Internet Advertising and Crowdsourcing, our goal is not just to maximize the number of matched edges, but to maximize the number of matched edges which eventually become successful (that is, get a click, finish the task, etc.). We typically have an estimate of the probability of a matched edge becoming successful. This was formalized by Mehta and Panigrahi [77] in the following model: when a vertex in V arrives, we see its incident edges; each edge e now has a (known) probability p_e of becoming successful if matched. The algorithm has to decide which neighbor u to match v to. After the decision, a coin is flipped with probability p_{uv} and we get a reward if the edge is successful. If not, then v is lost without a successful match,

while u can be tried again (each vertex can be matched at most once). The goal is to maximize the number of successful matches.

This now becomes a stochastic optimization problem: the algorithm gets a gain only after the edge is chosen, based on its probability of success. This model captures the notion of *pay-per-click* (the advertiser only pays upon a click) and the fact that the system knows the *click-through rate* of an ad on a ad-slot. In the crowdsourcing application, the probabilities represent the successful handling of a task by a worker.

In [77], the authors proved that this online, stochastic problem is strictly harder than the (non-stochastic) online bipartite matching problem from Section 3 (in which $p_e = 1 \ \forall e \in E$). They prove that no algorithm can do better than 0.62 (recall that $1 - \frac{1}{e} \simeq 0.63$). Greedy, which allocates the vertex v to that available neighbor u with the highest value of p_{uv}, still achieves $\frac{1}{2}$, but no better. They provided two algorithms, Stochastic Balance and Ranking, which achieve 0.56 and 0.53 when the probabilities on the edges are all equal and very small (and a curve of competitive ratios as the probabilities range from 0 to 1). An interesting technical observation made in this paper was to show the equivalence of the stochastic rewards matching problem to the Adwords problem in which the budgets are picked from an exponential distribution, and remain unknown until exhausted.

Open Question 13. Solve the problem in the general case when the p_e can be different. An algorithm for this is proposed in [77], but the analysis has technical difficulties. Either analyze this algorithm, or find a different algorithm.

Open Question 14. Consider the variant of this problem in which each edge has a weight as well as a probability of success; this is interesting in the free-disposal model or in the random arrival model, as it would generalize the problems in Section 7 (Note that in variants of problems such as Adwords or Display Ads with assumptions such as small bids or large capacities, Chernoff bounds can show that maximizing achieved value reduces to maximizing expected value).

10.1.2 Matching on General Graphs

All the problems in this survey are on bipartite graphs. This fits the motivating applications, where there are always two sides to the market with well-defined demand and supply, and only one side entering online. However, in other applications, most notably the *kidney exchange markets* (see [87] for details), there is only one type of an agent. This results in a matching problem on general graphs. The online version of matching on general graphs is a bit tricky to define: an arriving vertex can either be matched to a previously arrived neighbor, or kept alive for later (in other words, edges can only be matched when their second end-point arrives). It can be easily seen that no deterministic algorithm can do better than $\frac{1}{2}$ in the online setting (which is achieved by Greedy): for example, the second arriving vertex has an edge to the first, and if the algorithm matches that edge, then two more vertices will arrive with an edge to the first two respectively; if the algorithm does not match it, then the sequence ends. We can modify this example to prove that no randomized algorithm can do better than $\frac{2}{3}$.

The online version on general graphs may not be very well motivated, but there is still value in finding a fast simple offline approximation algorithm as opposed to the optimum algorithm, especially when the data is very big. This question was first asked by Dyer and Frieze [40], and the first non-trivial result was due to Aronson et al. [9] who showed that the algorithm which picks a random unmatched vertex and matches it to a random unmatched neighbor achieves a factor, incrementally greater than $\frac{1}{2}$, of 0.50000025.

Open Question 15. Improve the analysis, or find a better algorithm. In particular, can we analyze the following algorithm motivated by Ranking: Process the vertices in a Random Order, matching the next vertex to the highest available neighbor in the same order (which would necessarily be lower in the order than itself). A second variant (motivated by the analysis of Ranking in the Random Order model) would be to process vertices in one Random Order, and choose the neighbor according to another Random Order.

10.1.3 Matching with Concave Utilities

In [32], Devanur and Jain generalize the Adwords problem with small bids by introducing the problem of online matching with concave returns. Here the budget B_u of bidder u is replaced by a concave function M_u: if u is allocated an amount of x, then it receives a return of $M_u(x)$. Note that the Adwords problem uses the concave function $M_u(x) = x$ if $x \leq B_u$ and $M(x) = B_u$ if $x > B_u$. The motivation for studying the concave returns problem comes from various settings where the utility is not simply additive with a budget bound, for example, to model the cost of under-delivery in Display Ads or to provide proportional fairness. Using the Primal–Dual paradigm and convex programming duality, the authors prove that for any given concave function M, there is a number $F(M) \leq 1$, so that (a) there exists an algorithm which attains a factor of $\min_u F(M_u)$, and (b) in the case that all $M_u = M$, no algorithm can do better than $F(M)$. Additionally, the authors prove an interesting tight connection between the analysis of the competitive ratio and the upper bound example for the algorithm, for any given concave function.

10.1.4 AdX: Ad Exchange Allocation and Selective Call-Outs

In an Ad Exchange (AdX) setting (see [83] for a survey article), there is an exchange which auctions online ad requests from many publishers to many Ad Networks (a network is an agent for a collection of advertisers). In [24], Chakraborty et al. present an ad allocation model in this setting. The new feature of the exchange setting is that the exchange does not know the bids of the advertisers for the arriving query (these are known only by the corresponding network). Thus, when a ad request arrives, the exchange has to *call-out* to each network to solicit bids for the impression. Each network that is called out to will respond with a bid for this impression (from the set of bids of the advertisers it represents). Now, if the exchange could call-out to all the networks for all arriving requests, then then problem would reduce to a problem of the style we have seen in earlier sections. However, the networks are also *rate-limited* in that they can only handle some limited

number of call-outs per time period (consider that the exchange has a very large traffic of arriving requests, but many small networks can be infrastructure-limited).

In [24], the authors model the setting in the following manner: (i) the request arrival is in the Unknown IID model, (ii) each network N_i has a rate limit ρ_i which is the query-per-second (QPS) of call-outs it can handle (iii) for each query and for each network N_i, the exchange knows the distribution of the bids of N_i's advertisers. When a query arrives, the exchange has to determine a subset of the networks to call-out to, and then choose the maximum among the returned bids. The objective is to maximize the total value of the allocation, while keeping all call-out constraints satisfied. The trade-off is between calling out to a network with a good bid-distribution for the immediate query and keeping space in its call-out constraint for future call-outs.

Under these assumptions, the authors give an algorithm which achieves a $1 - \frac{1}{e}$ ratio. Similar to the algorithm in Section 6.2, the algorithm first obtains a random sample of the query sequence from the initial queries, solves an LP on the sample, and uses parameters from the sampled LP to solve the rest of the problem online. The authors prove a theorem of the following form (here m is the length of the sequence, n is the number of networks, and L is the maximum bid value):

Theorem 10.1 ([24]). For any $\epsilon, \delta > 0$, if the offline optimum is at least δm, then given a sample of size $\tilde{O}\left(\frac{n^2 L}{\delta \epsilon}\right)$, the algorithm achieves a ratio of $1 - \frac{1}{e} - \epsilon$. The algorithm has a preprocessing time of $poly\left(\frac{n^2 L}{\delta \epsilon}\right)$, and takes $O(n \log n)$ time per request.

10.1.5 Stochastic Matching in the Edge-probe Model

Another variant of matching that has been studied is that of stochastic matching in the edge-probing model, introduced in [28]: in this problem, motivated by online dating as well as kidney exchanges, we know the vertices V of a (general, non-bipartite) graph, but we do not know the edges, only the probability $p(u, v)$ that the edge (u, v) exists. We can probe a pair of vertices u, v to see if it materializes (that is, $(u, v) \in E$),

and if it does then we are committed to choosing that edge in the matching. There is a further constraint that each vertex v can only be probed a limited number t_v of times, called its patience parameter. Note that, if all the patience parameters are infinite, and the graph is bipartite then we can use RANKING to achieve a factor of $1 - \frac{1}{e}$; for general graphs, Costello et al. [31] provide an algorithm achieving a ratio of 0.573. For the problem with general patience values, Chen et al. [28] gave a greedy algorithm with a factor of $\frac{1}{4}$, the analysis of which was improved to $\frac{1}{2}$ by Adamczyk [3]. Bansal et al. [16] study the problem on weighted graphs and provide an LP-rounding based algorithm with factor of $\frac{1}{4}$, and factor $\frac{1}{3}$ for weighted bipartite graphs. They also study an online version of the problem (which turns out to be a generalization of online matching in the Known IID model as studied in Section 3.3) providing a ratio of $\frac{1}{7.92}$. Note that, with general patience parameters, the comparison is necessarily to the best adaptive algorithm, rather than the optimal in hindsight.

10.1.6 Other Variants Studied in the Literature

The literature on online matching and allocations has grown substantially in the last several years, with several new models and problems. We mention only a few additional ones here, without details. Goel and Mehta [50] introduced the model of decreasing bids, in which the bid of a bidder for a query is dynamic: the bid reduces along a curve, as it gets more clicks for ads on that query. This captures the notion that the advertiser has an inventory capacity, and the early clicks are worth more to it than the later ones. This was also studied by Buchbinder et al. [20] in the specific practical setting of frequency capping. Azar et al. [12] formalize and study the Adwords and online bipartite matching problems in the second-price setting (in the second-price matching problem one gets a gain from allocating a vertex only if there exists one more available neighbor at that time). They show that the problem becomes more difficult, in both the online and offline settings. In particular, they prove that the offline problem is APX-hard and provide a $\frac{1}{2}$-approximation algorithm. For the online problem, they show an upper bound of $\frac{1}{2}$, and an algorithm generalizing RANKING which achieves a ratio close to $\frac{1}{5}$.

Alaei et al. [7] introduced the AdCell problem, motivated by mobile advertising, which is a generalization of the Adwords problem in the following sense: each query v belongs to a cell-phone user, and each user has a capacity on the number of ads that can be shown to it, so as to not give a bad experience. This introduces a capacity on subsets of the query side of the graph as well. This becomes a generalization also of the classic secretary problem (Section 7.1), thus they study it in a stochastic model, introducing the model of non-identical distributions to this literature.

10.2 Open Problems

We have already listed some open questions as they came up through the previous sections. Here is a list of some more open questions related to future directions rather than to specific problems that we saw before. Progress on these would be of considerable interest.

Open Question 16. Find a new problem in the landscape of problems (Figure 2.1) which is theoretically elegant and practically compelling. Likewise, introduce new input models which capture more aspects of the practical setting, and are still amenable to theoretical analysis.

Open Question 17. Online matching with edge arrivals: Edges of the graph arrive online, and an arriving edge can be selected in the matching when it arrives. Clearly, a Greedy algorithm achieves $\frac{1}{2}$ since it constructs a maximal matching. Also, it can be easily shown that no deterministic algorithm can achieve better than $\frac{1}{2}$. Find optimal randomized algorithms for this problem. Consider the same problem in the random arrival model, in which edges arrive online. Analyze the performance of Greedy in this model. Find optimal algorithms.

Open Question 18. Online Matching with multi-objective optimization: As we noted in Section 9, there are multiple objective functions, corresponding to the multiple parties involved in the ecosystem. The problem formulations in this survey have focused on

maximizing the efficiency of the matches in an abstract sense. An open direction would be to formulate a multi-objective optimization question based on the different objective functions, and provide algorithms for the same. (Work in this direction has been reported in currently unpublished results [4, 66]).

Open Question 19. Online Matching with arrivals and departures: Suppose vertices on both sides can arrive as well as depart. Two adjacent vertices can be matched only while both are present. Note that the classic problem from Section 3 is a special case, with vertices in U arriving at time 0 and departing at infinity, while the ith vertex in V arriving and departing at time i. A GREEDY algorithm again gives a ratio of $\frac{1}{2}$. The question is what arrival–departure patterns allow for algorithms that beat $\frac{1}{2}$ (can we beat $\frac{1}{2}$ irrespective of the arrival–departure pattern?).

Open Question 20. Budget Oblivious Algorithms: The algorithms in [21, 78] for Adwords with small bids need to know the budget value for each bidder. Consider instead the budget-oblivious model: The budget of a vertex $u \in U$ is unknown until precisely the time that it gets exhausted. Note that GREEDY works in the budget-oblivious model, and gives a ratio of $\frac{1}{2}$. Find an algorithm which beats $\frac{1}{2}$ in this model, or prove that one cannot beat $\frac{1}{2}$ in this model. Practically, one can imagine that an algorithm that works with lesser information may be more robust. But the motivation for this problem is mainly theoretical and comes from the problem of matching with stochastic rewards (Section 10.1.1): In [77], the authors reduced this problem to an allocation problem in which the algorithm does not know the budgets, but knows that the budgets are picked from a certain distribution. This open problem asks if the distributional assumption can be dropped.

Open Question 21. Find applications of Perturbed Greedy or Bid-Scaling algorithms, possibly outside the domain of online matching.

Open Question 22. Find more applications of the randomized Primal–Dual method, possibly outside the domain of online matching, and even possibly for offline problems. Reformulations of known algorithms in this framework would also be interesting.

Open Question 23. Note that the algorithms in the Primal–Dual framework use multiplicative updates to update the dual variables. The algorithms for Adwords and vertex-weighted matching use a trade-off function similar to the use of potential functions, and also utilize a "soft-max" choice function. Find stronger connections of bid-scaling algorithms with the multiplicative-weights update algorithms and problems (see the survey [10]). Find connections of bid-scaling algorithms to potential function based algorithms (for example, those used for makespan minimization in scheduling [11]).

Open Question 24. Find a natural restriction on graphs (for example, on the degree distribution, or on the distribution of the input query stream) which allows for simple online algorithms to perform close to optimally (one such example, from Section 3.2, is that RANKING has a ratio close to 1 in the Random Order model when the graph has many perfect matchings).

References

[1] A. Abdulkadiroğlu, P. A. Pathak, and A. E. Roth, "The New York city high school match," *American Economic Review*, pp. 364–367, 2005.

[2] Z. Abrams, O. Mendelevitch, and J. Tomlin, "Optimal delivery of sponsored search advertisements subject to budget constraints," in *ACM Conference on Electronic Commerce*, pp. 272–278, 2007.

[3] M. Adamczyk, "Improved analysis of the greedy algorithm for stochastic matching," *Information Processing Letters*, vol. 111, no. 15, pp. 731–737, 2011.

[4] G. Aggarwal, Y. Cai, A. Mehta, and G. Pierrakos, "Biobjective online bipartite matching," Manuscript.

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

[6] S. Agrawal, Z. Wang, and Y. Ye, "A dynamic near-optimal algorithm for online linear programming," *CoRR,* abs/0911.2974, 2009.

[7] S. Alaei, M. T. Hajiaghayi, V. Liaghat, D. Pei, and B. Saha, "Adcell: Ad allocation in cellular networks," in *ESA*, pp. 311–322, 2011.

[8] N. Andelman and Y. Mansour, "Auctions with budget constraints," in *Scandinavian Workshop on Algorithm Theory (SWAT)*, pp. 26–38, 2004.

[9] J. Aronson, M. Dyer, A. Frieze, and S. Suen, "Randomized greedy matching. II," *Random Structures & Algorithms*, vol. 6, no. 1, pp. 55–73, 1995.

[10] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: A meta-algorithm and applications," *Theory of Computing*, vol. 8, pp. 121–164, 2012.

[11] B. Awerbuch, Y. Azar, and S. Plotkin, "Throughput-competitive on-line routing," in *Proceedings of Annual Symposium on Foundations of Computer Science*, pp. 32–40, 1993.

[12] Y. Azar, B. Birnbaum, A. Karlin, and C. Nguyen, "On revenue maximization in second-price ad auctions," *Algorithms-ESA 2009*, pp. 155–166, 2009.

[13] Y. Azar and A. Litichevskey, "Maximizing throughput in multi-queue switches," *Algorithms–ESA 2004*, pp. 53–64, 2004.

[14] Y. Azar, B. E. Birnbaum, A. R. Karlin, C. Mathieu, and C. T. Nguyen, "Improved approximation algorithms for budgeted allocations," in *ICALP (1)*, pp. 186–197, 2008.

[15] B. Bahmani and M. Kapralov, "Improved bounds for online stochastic matching," in *ESA (1)*, pp. 170–181, 2010.

[16] N. Bansal, A. Gupta, J. Li, J. Mestre, V. Nagarajan, and A. Rudra, "When lp is the cure for your matching woes: Improved bounds for stochastic matchings — (extended abstract)," in *ESA (2)*, pp. 218–229, 2010.

[17] C. Berge, "Two theorems in graph theory," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 43, no. 9, p. 842, 1957.

[18] B. E. Birnbaum and C. Mathieu, "On-line bipartite matching made simple," *SIGACT News*, vol. 39, no. 1, pp. 80–87, 2008.

[19] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*, volume 53. Cambridge: Cambridge University Press, 1998.

[20] N. Buchbinder, M. Feldman, A. Ghosh, and J. Naor, "Frequency capping in online advertising," in *WADS*, pp. 147–158, 2011.

[21] N. Buchbinder, K. Jain, and J. Naor, "Online primal-dual algorithms for maximizing ad-auctions revenue," in *ESA*, pp. 253–264, 2007.

[22] N. Buchbinder and J. Naor, "The design of competitive online algorithms via a primal-dual approach," *Foundations and Trends in Theoretical Computer Science*, vol. 3, no. 2–3, pp. 93–263, 2009.

[23] D. Chakrabarty and G. Goel, "On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and gap," in *FOCS*, pp. 687–696, 2008.

[24] T. Chakraborty, E. Even-Dar, S. Guha, Y. Mansour, and S. Muthukrishnan, "Selective call out and real time bidding," *Internet and Network Economics*, pp. 145–157, 2010.

[25] D. Charles, D. Chakrabarty, M. Chickering, N. R. Devanur, and L. Wang, "Budget smoothing for internet ad auctions: a game theoretic approach," in *Proceedings of the ACM Conference on Electronic Commerce*, pp. 163–180, 2013.

[26] D. X. Charles, M. Chickering, N. R. Devanur, K. Jain, and M. Sanghi, "Fast algorithms for finding matchings in lopsided bipartite graphs with applications to display ads," in *ACM Conference on Electronic Commerce*, pp. 121–128, 2010.

[27] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM Journal on Computing*, vol. 35, no. 3, pp. 713–728, 2005.

[28] N. Chen, N. Immorlica, A. R. Karlin, M. Mahdian, and A. Rudra, "Approximating matches made in heaven," in *ICALP (1)*, pp. 266–278, 2009.
[29] Y. Chen, P. Berkhin, B. Anderson, and N. R. Devanur, "Real-time bidding algorithms for performance-based display ad allocation," in *KDD*, pp. 1307–1315, 2011.
[30] V. Chvátal, *Linear Programming*. WH Freeman, 1983.
[31] K. P. Costello, P. Tetali, and P. Tripathi, "Stochastic matching with commitment," in *Automata, Languages, and Programming*, pp. 822–833, Springer, 2012.
[32] N. R. Devanur and K. Jain, "Online matching with concave returns," in *Proceedings of the Symposium on Theory of Computing*, pp. 137–144, 2012.
[33] N. Devanur, K. Jain, and R. Kleinberg, "Randomized primal-dual analysis of ranking for online bipartite matching," in *SODA*, 2013.
[34] N. R. Devanur, "Online algorithms with stochastic input," *SIGecom Exchanges*, vol. 10, no. 2, pp. 40–49, 2011.
[35] N. R. Devanur and T. P. Hayes, "The adwords problem: online keyword matching with budgeted bidders under random permutations," in *ACM Conference on Electronic Commerce*, pp. 71–78, 2009.
[36] N. R. Devanur, K. Jain, B. Sivan, and C. A. Wilkens, "Near optimal online algorithms and fast approximation algorithms for resource allocation problems," in *ACM Conference on Electronic Commerce*, pp. 29–38, 2011.
[37] N. R. Devanur, B. Sivan, and Y. Azar, "Asymptotically optimal algorithm for stochastic adwords," in *Proceedings of the ACM Conference on Electronic Commerce*, pp. 388–404, 2012.
[38] N. B. Dimitrov and C. G. Plaxton, "Competitive weighted matching in transversal matroids," in *ICALP (1)*, pp. 397–408, 2008.
[39] E. A. Dinic, "Algorithm for solution of a problem of maximum flow in networks with power estimation," *In Soviet Math. Dokl*, vol. 11, no. 5, no. 5, pp. 1277–1280, 1970.
[40] M. E. Dyer and A. M. Frieze, "Randomized greedy matching," *Random Struct. Algorithms*, vol. 2, no. 1, pp. 29–46, 1991.
[41] E. B. Dynkin, "The optimum choice of the instant for stopping a markov process," *Soviet Mathematics. Doklady*, vol. 4, 1963.
[42] J. Edmonds, "Paths, trees, and flowers," *Canadian Journal of mathematics*, vol. 17, no. 3, pp. 449–467, 1965.
[43] U. Feige and J. Vondrak, "Approximation algorithms for allocation problems: Improving the factor of 1-1/e," in *Annual IEEE Symposium on Foundations of Computer Science, 2006. FOCS'06*, pp. 667–676, 2006.
[44] J. Feldman, M. Henzinger, N. Korula, V. S. Mirrokni, and C. Stein, "Online stochastic packing applied to display ad allocation," in *ESA (1)*, pp. 182–194, 2010.
[45] J. Feldman, N. Korula, V. S. Mirrokni, S. Muthukrishnan, and M. Pál, "Online ad assignment with free disposal," in *WINE*, pp. 374–385, 2009.
[46] J. Feldman, A. Mehta, V. S. Mirrokni, and S. Muthukrishnan, "Online stochastic matching: Beating 1-1/e," in *FOCS*, pp. 117–126, 2009.

[47] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight approximation algorithms for maximum general assignment problems," in *Proceedings of the Annual ACM-SIAM symposium on Discrete Algorithm*, pp. 611–620, 2006.

[48] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly*, pp. 9–15, 1962.

[49] R. Garg, V. Kumar, and V. Pandit, "Approximation algorithms for budget-constrained auctions," *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pp. 102–113, 2001.

[50] G. Goel and A. Mehta, "Adwords auctions with decreasing valuation bids," in *WINE*, pp. 335–340, 2007.

[51] G. Goel and A. Mehta, "Online budgeted matching in random input models with applications to adwords," in *SODA*, pp. 982–991, 2008.

[52] M. Goemans and J. Kleinberg, "An improved approximation ratio for the minimum latency problem," *Mathematical Programming*, vol. 82, no. 1, pp. 111–124, 1998.

[53] B. Haeupler, V. Mirrokni, and M. Zadimoghaddam, "Online stochastic weighted matching: Improved approximation algorithms," *Internet and Network Economics*, pp. 170–181, 2011.

[54] C.-J. Ho and J. W. Vaughan, "Online task assignment in crowdsourcing markets," in *AAAI*, 2012.

[55] J. E. Hopcroft and R. M. Karp, "An nˆ5/2 algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225–231, 1973.

[56] P. Jaillet and X. Lu, "Online stochastic matching: New algorithms with better bounds," Manuscript.

[57] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani, "Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp," *Journal of ACM*, vol. 50, no. 6, pp. 795–824, 2003.

[58] B. Kalyanasundaram and K. Pruhs, "An optimal deterministic algorithm for online b-matching," *Theoretical Computer Science*, vol. 233, no. 1–2, pp. 319–325, 2000.

[59] M. Kapralov, I. Post, and J. Vondrák, "Online and stochastic variants of welfare maximization," *CoRR,* abs/1204.1025, 2012.

[60] C. Karande, A. Mehta, and R. Srikant, "Optimization of budget constrained spend in search advertising," in *WSDM*, 2013.

[61] C. Karande, A. Mehta, and P. Tripathi, "Online bipartite matching with unknown distributions," in *STOC*, pp. 587–596, 2011.

[62] R. M. Karp, E. Upfal, and A. Wigderson, "Constructing a perfect matching is in random nc," *Combinatorica*, vol. 6, no. 1, pp. 35–48, 1986.

[63] R. M. Karp, U. V. Vazirani, and V. V. Vazirani, "An optimal algorithm for on-line bipartite matching," in *STOC*, pp. 352–358, 1990.

[64] T. Kesselheim, K. Radke, A. Tönnis, and B. Vöcking, "An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions," in *Algorithms–ESA 2013*, pp. 589–600, Springer, 2013.

[65] S. Khot, R. J. Lipton, E. Markakis, and A. Mehta, "Inapproximability results for combinatorial auctions with submodular utility functions," *Algorithmica*, vol. 52, no. 1, pp. 3–18, 2008.

[66] N. Korula, V. Mirrokni, and M. Zadimoghaddam, "Bicriteria online matching: Maximizing weight and cardinality," Manuscript.

[67] N. Korula and M. Pál, "Algorithms for secretary problems on graphs and hypergraphs," in *ICALP (2)*, pp. 508–520, 2009.

[68] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 2006.

[69] B. Lehmann, D. J. Lehmann, and N. Nisan, "Combinatorial auctions with decreasing marginal utilities," *Games and Economic Behavior*, vol. 55, no. 2, pp. 270–296, 2006.

[70] D. V. Lindley, "Dynamic programming and decision theory," *Applied Statistics*, pp. 39–51, 1961.

[71] L. Lovász and M. D. Plummer, *Matching Theory,* volume 367. American Mathematical Society, 2009.

[72] M. Mahdian, H. Nazerzadeh, and A. Saberi, "Allocating online advertisement space with unreliable estimates," in *ACM Conference on Electronic Commerce*, pp. 288–294, 2007.

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

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

[75] R. McEliece, E. Rodemich, H. Rumsey, and L. Welch, "New upper bounds on the rate of a code via the delsarte-macwilliams inequalities," *IEEE Transactions on Information Theory*, vol. 23, no. 2, pp. 157–166, 1977.

[76] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, 1999.

[77] A. Mehta and D. Panigrahi, "Online matching with stochastic rewards," in *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2012.

[78] A. Mehta, A. Saberi, U. V. Vazirani, and V. V. Vazirani, "Adwords and generalized online matching," *Journal of ACM*, vol. 54, no. 5, 2007.

[79] V. S. Mirrokni, S. O. Gharan, and M. Zadimoghaddam, "Simultaneous approximations for adversarial and stochastic online budgeted allocation," in *SODA*, pp. 1690–1701, 2012.

[80] V. S. Mirrokni, M. Schapira, and J. Vondrák, "Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions," in *ACM Conference on Electronic Commerce*, pp. 70–77, 2008.

[81] R. Motwani, R. Panigrahy, and Y. Xu, "Fractional matching via balls-and-bins," in *APPROX-RANDOM*, pp. 487–498, 2006.

[82] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani, "Matching is as easy as matrix inversion," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, pp. 345–354, 1987.

[83] S. Muthukrishnan, "Ad exchanges: Research issues," *Internet and Network Economics*, pp. 1–12, 2009.

[84] A. Roth, T. Sonmez, and U. Unver, "Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences," *American Economic Review*, vol. 97, no. 3, 2007.

[85] A. E. Roth, "The evolution of the labor market for medical interns and residents: a case study in game theory," *The Journal of Political Economy*, pp. 991–1016, 1984.

[86] A. E. Roth and M. A. O. Sotomayor, *Two-sided Matching: A Study in Game-theoretic Modeling and Analysis,* vol. 18. Cambridge University Press, 1992.

[87] A. E. Roth, T. Sonmez, and M. U. Unver, "Pairwise kidney exchange," *Journal of Economic Theory*, vol. 125, pp. 151–188, 2005.

[88] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Math. Program*, vol. 62, pp. 461–474, 1993.

[89] A. Srinivasan, "Budgeted allocations in the full-information setting," in *APPROX-RANDOM*, pp. 247–253, 2008.

[90] L. G. Valiant, "The complexity of computing the permanent," *Theoretical Computer Science*, vol. 8, no. 2, pp. 189–201, 1979.

[91] V. V. Vazirani, "A theory of alternating paths and blossoms for proving correctness of the general graph maximum matching algorithm," *Combinatorica*, vol. 14, no. 1, pp. 71–109, 1994.

[92] E. Vee, S. Vassilvitskii, and J. Shanmugasundaram, "Optimal online assignment with forecasts," in *ACM Conference on Electronic Commerce*, pp. 109–118, 2010.

[93] J. Vondrák, "Optimal approximation for the submodular welfare problem in the value oracle model," in *STOC*, pp. 67–74, 2008.