Fast Algorithms for Finding Matchings in Lopsided Bipartite Graphs with Applications to Display Ads

Denis Charles
Microsoft Corp.
One Microsoft Way
Redmond, WA
cdx@microsoft.com

Max Chickering
Microsoft Corp.
One Microsoft Way
Redmond, WA
max.chickering@microsoft.com

Nikhil R. Devanur
Microsoft Corp.
One Microsoft Way
Redmond, WA
nikdev@microsoft.com

Kamal Jain
Microsoft Corp.
One Microsoft Way
Redmond, WA
kamalj@microsoft.com

Manan Sanghi
Microsoft Corp.
One Microsoft Way
Redmond, WA
manansa@microsoft.com

ABSTRACT

We derive efficient algorithms for both detecting and representing matchings in lopsided bipartite graphs; such graphs have so many nodes on one side that it is infeasible to represent them in memory or to identify matchings using standard approaches. Detecting and representing matchings in lopsided bipartite graphs is important for allocating and delivering guaranteed-placement display ads, where the corresponding bipartite graph of interest has nodes representing advertisers on one side and nodes representing web-page impressions on the other; real-world instances of such graphs can have billions of impression nodes. We provide theoretical guarantees for our algorithms, and in a real-world advertising application, we demonstrate the feasibility of our detection algorithms.

Categories and Subject Descriptors

F.1.2 [**Modes of Computation**]: Probabilistic computation; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.2.2 [**Discrete Mathematics**]: Graph Theory; K.4.4 [**Computers and Society**]: Electronic commerce

General Terms

Algorithms,Theory

Keywords

Bipartite Matching, Selling Display Advertisements, Online Algorithms, Linear Programming

EC'10, June 7–11, 2010, Cambridge, Massachusetts, USA.

1. INTRODUCTION

In a popular form of online advertising, advertisers buy *display ads* (i.e., graphical "banner" ads) to be shown to users viewing online content, and the publishers of that content are compensated a fixed amount for each web-page *impression* (i.e., user view of a page) for which an advertisement is shown. This form of advertising is particularly appropriate for branding campaigns where, unlike most paid-search campaigns, the advertiser values impressions even if the user does not click. Display ads are typically sold with guaranteed impression goals; the publisher (or ad network) sells a fixed number of impressions and is responsible for making sure that these campaign goals are met. In addition, publishers typically allow advertisers to buy display ads that target both content type and particular attributes of the user who is shown the impression. For example, an advertiser might purchase a million impressions to be shown on sports-related web pages, but may only want to show the ads to users who are thought to be female[1].

The assignment of an advertiser to each impression on a publisher site can be understood as a matching problem on a bipartite graph. In particular, let $G = (A, I, E)$ be a bipartite graph with two sides A and I and the edge set E. (A for advertisers and I for impressions.) Let $\delta(i)$ denote the set of edges incident on vertex i in graph G. Given *sizes* B_a for all $a \in A$, a *complete matching* $M \subseteq E$ is such that

- for all $a \in A$, $|M \cap \delta(a)| \geq B_a$, and
- for all $i \in I$, $|M \cap \delta(i)| = 1$.

In this paper, we present matching algorithms for graphs that are *lopsided*; that is, graphs for which $|I| \gg |A|$ and for which I cannot be represented explicitly. In particular, we solve two distinct matching problems: (1) the *existence problem*, in which we want to know whether there exists a complete matching and (2) the *representation problem*, which is to succinctly represent a complete matching if one exists. The existence problem is important to solve in order for a publisher to decide whether or not to accept a

[1]Large publishers and ad networks can use both user-provided profile information and browsing history of users to derive thousands of targetable attributes.

candidate campaign from an advertiser; given a set of advertiser campaigns for which there exists a complete matching, the publisher will want to know if there remains a complete matching after adding the new campaign. The representation problem is important to solve in order for a publisher to implement an on-line delivery algorithm; the publisher needs to know *how* to assign each impression to an advertiser, not just whether or not such an assignment exists.

In the algorithms we present, the method by which I is (implicitly) represented differs depending on which problem we are solving. To solve the existence problem, we assume that we can draw samples from I (with replacement) from the distribution over both the targetable attributes of interest and across time. The sampled impressions are not labeled, our algorithms require only the set of neighbors in A for each such sample. For the representation problem, we assume that we can compute, for any set of advertiser nodes, the number of impressions matching the targeting criteria (which includes the time period of interest) of at least one advertiser in that set. Although we do not go into the details of these representations, in practice we use a Bayesian-network model for each time interval of interest to represent the joint distribution over the non-temporal attributes; these models are estimated from data and then used both to produce random samples for the existence problem and to compute the "union" computations for the representation problem.

An important property for any algorithm solving the existence problem is that the algorithm must be able to complete quickly enough that a sales representative for a publisher can accept or reject candidate campaigns in "real time" (for example, while on the phone). For the representation problem, on the other hand, the algorithm can be run periodically offline (e.g., once a day).

Given our representation of I for the existence problem, it is easy to see that one needs to generate at least $|I|$ samples if one wants to solve the problem exactly, and hence getting a running time that is independent of I is impossible. Hence we consider an approximate version of the problem, which we call the Gap-Existence problem with gap ϵ, a parameter. The problem is to distinguish between the following two cases.

YES There exists a complete matching with sizes B_a, and

NO There does not exist a complete matching with sizes $B_a(1-\epsilon)$,

for some given $\epsilon > 0$. The gap version is still a reasonable problem to solve for the publisher. This is because in practice a certain level of under-delivery (roughly about 5%) is tolerated. Even if no under-delivery is tolerated, one can consider the gap version where the YES case corresponds to sizes $B_a(1+\epsilon)$, and the NO case corresponds to the actual sizes B_a. In this version, the publisher is fine if he misses an advertiser that would push him close to capacity. In other words, he is happy to accept only those advertisers who he could still serve if he only had a $1/(1+\epsilon)$ fraction of the actual traffic.

For the representation problem, we allow *fractional* matchings: a fractional matching is given by $\{x_{ai}\}_{(a,i)\in E}$ with x_{ai} being the fraction of edge (a,i) included in the matching. A complete fractional matching is such that for all i, $\sum_a x_{ai} = 1$ and for all a, $\sum_i x_{ai} \geq B_a$. As described above, we also assume that I is represented in such a way that for all $S \subseteq A$, one can get the total number of impressions that could be assigned to some advertiser in S, that is $|\Gamma(S)|$, where $\Gamma(S)$ is the set of neighbors of S in the graph G. The relaxation to allow fractional matchings is justified by the observation that one could use the fractional matching to assign impressions probabilistically. A simple Chernoff bound argument shows that this is a reasonable solution given that the B_a's are fairly large.

For the remainder of the paper, we use n to denote $|A|$. The following two theorems summarize the main results of our paper.

THEOREM 1. *There exists an algorithm that solves the Gap-Existence problem with gap ϵ in time $O(\frac{n \log n}{\epsilon^2} \frac{|I|}{\min_a B_a})$ and using space $O(n)$.*

Our experimental results (see Section 3.1) suggest that in practice one actually needs far fewer samples than suggested by this bound. The running times in the experiments suggest that this algorithm could indeed be of practical use.

THEOREM 2. *There exists a representation of a complete (possibly fractional) matching of size $O(n^2)$ (given one exists). Such a representation can be found in time polynomial in n, given oracle calls to answer $|\Gamma(S)|$ for any $S \subseteq [n]$. Given such a representation and a given $i \in I$, one can compute $\{x_{ai}\}_{a:(a,i)\in E}$ in time $O(n^2)$.*

The algorithm to compute a representation uses the ellipsoid algorithm and as a result it is not clear if it is practical. However, one only needs to compute these representations periodically and hence a longer running time might be tolerated. Given the representation however, the allocation can be computed really fast, even in a matter of milliseconds.

Related work

Bipartite matching is a classic problem in combinatorial optimization that has been much studied. Matching problems with capacities > 1 are typically formulated as max-flow problems [3] and fairly fast and sophisticated algorithms are known to solve such problems [5]. The key new feature of the problems we consider is the lopsidedness of the graph. Ahuja et al [1] give an algorithm for max flow on bipartite graphs with (one term of) the running time depending on the smaller side. However, all these algorithms require the entire graph to be stored in memory (thus requiring time and space that scaled at least linearly with the number of edges). Such algorithms are impractical in our setting.

Several aspects of the display ad system have been studied recently [4, 6, 8]. However, as far as we know, the existence problem has not been studied before. A special case of our problem is when all the B_a's are the same. In this special case, our algorithm for the existence problem and its analysis appears in a paper by Motwani et al [7] as a "balls and bins" problem (in a different context). There is a natural generalization of this algorithm that seems to also work well. However, the proof of Motwani et al [7] does not generalize. In fact, it is open if this natural generalization actually solves the existence problem. We discuss this in more detail in Section 5.

The representation problem has been considered by Vee et al [8] and by Devanur and Hayes [2]; our results do not seem directly comparable to theirs.

2. EXISTENCE PROBLEM: ALGORITHM

Since bipartite matching is a fairly well understood problem, one could try to use some of the well known algorithms for bipartite matching for our problem. One natural approach is to generate a sufficient number of samples from I and solve a matching problem on the sampled graph. But this approach is still too slow for our purposes, especially since the resulting graph is still too large. So we propose a streaming-like algorithm. The following is a framework for such an algorithm, with the details of how to implement certain steps left out.

The algorithm picks a certain number of sample impressions to generate, $\dot{m}$, and a certain number of target impressions, $\dot{k}_a$ for each advertiser a. It then generates random samples from I and assigns the impressions online (as soon as they are generated). If after all the $\dot{m}$ samples have been generated, each advertiser a has received at least $\dot{k}_a$ impressions, then the algorithm outputs YES, otherwise it outputs NO. The idea is that if there is a complete matching, then the algorithm should be able to achieve the given targets whereas if there is no near-complete matching, then no matter what the algorithm does, it should not be able to achieve the targets. The algorithm is complete by specifying the choice of

1. number of samples $\dot{m}$,
2. the target $\dot{\mathbf{k}}$ ($\dot{\mathbf{k}}$ denotes the vector of k_a's) and
3. the assignment function, which given the set of neighbors of an impression, and the remaining number of samples and targets, specifies the advertiser to whom that impression is assigned. We call such an assignment function as one *step* of the algorithm.

Generic Streaming algorithm

- Initialize: $\mathbf{k} = \dot{\mathbf{k}}$, $m = \dot{m}$.
- While $m > 0$
  - Get a random $i \in I$ and $\Gamma(i)$.
  - $a :=$ GENERIC-STEP$(\Gamma(i), \mathbf{k}, m)$.
  - Update $\mathbf{k} := \mathbf{k} - \mathbf{e}_a$ and $m := m - 1$. ($\mathbf{e}_a$ is the unit vector with 1 in the a^{th} position and 0's elsewhere.)
- If $\mathbf{k} \leq \mathbf{0}$, output YES
  Else output NO

We present a general class of single steps that we call WATER-LEVEL-STEP. The resulting algorithms are called WATER-LEVEL algorithms. (The standard waterlevel algorithm, when all B_a's are the same, is to always assign i to that a that has the highest remaining target.) A WATER-LEVEL-STEPstep is parameterized by a potential function, f that takes as input, a vector of targets, $\mathbf{k}$, and the number of remaining samples, m. WATER-LEVEL-STEP **with potential function** $f(\mathbf{k}, m)$

- Given: current $\mathbf{k}$, m and $\Gamma(i)$.
- Assign i to $a = \arg\min_{a' \in \Gamma(i)}\{f(\mathbf{k} - \mathbf{e}_{a'}, m-1)\}$, that is, assign i to that a in its neighborhood such that the function f on the resulting target is minimized.

A WATER-LEVEL algorithm is completely specified by the choice of $f, \dot{m}$ and $\dot{\mathbf{k}}$. We show that there exists a choice of $f, \dot{m}$ and $\dot{\mathbf{k}}$ that solves the gap-existence problem, but not specify those choices right away. We will simultaneously give the proof of the following theorem and present the corresponding choices, thus revealing how these choices are dictated by the proof.

THEOREM 3. *For all $\epsilon, \delta > 0$, there is a certain choice of $f, \dot{\mathbf{k}}$ and $\dot{m}$ such that the resulting streaming algorithm with probability[2] $1 - \delta$ solves the Gap-Existence problem with gap ϵ.*

3. EXISTENCE PROBLEM: ANALYSIS

Choice of f

For the purpose of analysis and the choice of f, we consider a hypothetical algorithm, which we call PURE-RANDOM. PURE-RANDOM is defined only in the YES case, and it requires a complete matching M. It is a streaming algorithm and hence we only need to specify the generic step. (We still leave $\mathbf{k}$ and m unspecified.) By abuse of notation we let $M(i)$ denote the node in A that i is matched to in M. This is well defined since $|M \cap \delta(i)| = 1$.
PURE-RANDOM-STEP (with a complete matching M)

- Given: i.
- Assign i to $a = M(i)$

The idea is to design a WATER-LEVEL-STEP such that in the YES case, the probability that the WATER-LEVEL algorithm achieves the given targets is at least as much as the probability that the PURE-RANDOM algorithm achieves the targets (for all m and $\mathbf{k}$). Then, we pick the targets such that PURE-RANDOM algorithm is guaranteed to achieve it. Finally, the number of samples is picked in such a way that if we are in the NO case, then any algorithm would fail to achieve the targets. All these are probabilistic events that happen with probability $1 - \delta$.

In fact, we don't actually show that the probability of achieving the targets is higher for WATER-LEVEL. Instead, we show that a suitable bound on the failure probability of PURE-RANDOM also holds for WATER-LEVEL. This is achieved through a "Hybrid Argument"[3].
Hybrid Argument: We consider a sequence of algorithms, $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{\dot{m}}$, where $\mathcal{H}_0$ is the PURE-RANDOM algorithm, $\mathcal{H}_{\dot{m}}$ is the WATER-LEVEL algorithm, and $\mathcal{H}_t$ is the algorithm that does WATER-LEVEL-STEP for the first t steps and then does PURE-RANDOM-STEP for steps $t+1$ to $\dot{m}$. We also define a sequence of potential functions $\phi_0 \geq \phi_1 \geq \cdots, \phi_{\dot{m}}$ where ϕ_t depends on $\mathcal{H}_t$ such that $\phi_{\dot{m}}$ is equal to the failure probability of WATER-LEVEL and ϕ_0 is a Chernoff-type bound on the failure probability of PURE-RANDOM. Ideally, we would have liked ϕ_0 to be exactly the failure probability of PURE-RANDOM, but the exact failure probabilities do not have a succinct closed form expression (they can be expressed as summations). Hence we have to work with

[2]We use δ for the failure probability, which is not to be confused with $\delta(i)$ being the edge set incident on i. The context should clarify the meaning.
[3] The name is derived from a similar argument used in cryptography which is also called a Hybrid argument.

bounds on the probability that we get when we prove concentration inequalities such as Chernoff bounds.

Consider the algorithm $\mathcal{H}_t$ at step t. The natural candidate for the WATER-LEVEL-STEP would be to put the ball in that bin that minimizes the failure probability for the rest of the algorithm, which is PURE-RANDOM. But as mentioned before, this failure probability is hard to work with. Therefore the WATER-LEVEL-STEP instead uses a bound on the failure probability that is nicer, and this is what $f(\mathbf{k}, m)$ will be. f will also be a function of $p_a = B_a/|I|$, that is the probability that PURE-RANDOM-STEP assigns an impression to advertiser a, but this dependence will remain implicit. If $m = 0$, then we define $f(\mathbf{k}, m) = 0$ if $\mathbf{k}$ is the all zeroes vector, and 1 otherwise. Otherwise $f = \min\{\sum_a f_a(k_a, m), 1\}$ where f_a is a bound on the failure probability of the a^{th} bin with target k_a and m remaining balls. In other words the failure probability for the entire algorithm is obtained by taking a union bound over the failure probabilities for each bin. Let $\mathbf{k}_t(\mathcal{H}_t)$ be the random variable denoting the vector of remaining targets after step t of algorithm $\mathcal{H}_t$. We are now ready to define the potential function ϕ.

$$\phi_t := \mathbf{E}_{\text{ first } t \text{ steps of } \mathcal{H}_t} \left[f(\mathbf{k}_t(\mathcal{H}_t), \dot{m} - t) \right].$$

Note that $\phi_0 = f(\mathbf{k}, m)$, where $\mathbf{k}$ is the initial target, is simply a bound on the failure probability of PURE-RANDOM, and $\phi_m =$ the failure probability of WATER-LEVEL, as required.

Recall that we wish to show that $\phi_0 \geq \phi_1 \geq \cdots, \phi_{\dot{m}}$, or equivalently, $\phi_t \geq \phi_{t+1}$ for all t. We do this in two steps, by showing that $\phi_t \geq \mu_t \geq \phi_{t+1}$, where μ is defined as follows.

$$\mu_t := \mathbf{E}_{\text{ first } t+1 \text{ steps of } \mathcal{H}_t} \left[f(\mathbf{k}_{t+1}(\mathcal{H}_t), \dot{m} - t - 1) \right].$$

The easier of the two inequalities is that $\mu_t \geq \phi_{t+1}$ and we show that now. Note that $\mathcal{H}_t$ and $\mathcal{H}_{t+1}$ only differ in the $t + 1^{\text{st}}$ step. μ_t and ϕ_{t+1} are the expected values of f after $t+1$ steps of $\mathcal{H}_t$ and $\mathcal{H}_{t+1}$ respectively. While the $t+1^{\text{st}}$ step of $\mathcal{H}_t$ is PURE-RANDOM, that of $\mathcal{H}_{t+1}$ is defined to minimize f, and the resulting value is ϕ_{t+1} which is therefore smaller than μ_t.

The other inequality, $\phi_t \geq \mu_t$, follows essentially from what we call the Monotone Quality Property (MQP) of f_a's. This property is as follows.

DEFINITION 4. *$f_a(k, m)$ is said to satisfy the MQP if*

$$f_a(k, m) \geq p_a f_a(k - 1, m - 1) + (1 - p_a) f_a(k, m - 1).$$

If f_a were the exact failure probabilities, then they would satisfy the MQP with equality. And vice versa, if we solved the above with equality as a recurrence, then the solution is the exact failure probability. We now show that MQP implies $\mu_{t,a} \leq \phi_{t,a}$, where $\mu_{t,a}$ (and respectively $\phi_{t,a}$) is the i^{th} component of μ_t (respectively ϕ_t):

$$\begin{aligned}
\mu_{t,a} &= \mathbf{E}_{\text{ first } t+1 \text{ steps of } \mathcal{H}_t} \left[f_a(k_{t+1,a}(\mathcal{H}_t), m - t - 1) \right] \\
&= \mathbf{E}_{\text{ first } t \text{ steps of } \mathcal{H}_t} [p_a f_a(k_{t,a}(\mathcal{H}_t) - 1, m - t - 1) + \\
&\quad (1 - p_a) f_a(k_{t,a}(\mathcal{H}_t), m - t - 1)] \\
&\leq \mathbf{E}_{\text{ first } t \text{ steps of } \mathcal{H}_t} \left[f_a(k_{t,a}(\mathcal{H}_t), m - t) \right] \text{ (Due to MQP)} \\
&= \phi_{t,a}.
\end{aligned}$$

We have thus shown that it is sufficient that the functions f_a bound the failure probabilities for the a^{th} advertiser in PURE-RANDOM, and that they satisfy the MQP. It remains to pick such functions. We give one such family of functions here. Let

$$f_a(k, m) = \inf_h \{ e^{hk} (1 - p_a + p_a e^{-h})^m \}.$$

LEMMA 5. *f_a satisfies the MQP.*

PROOF. For all h,

$$\begin{aligned}
& p_a f_a(k - 1, m - 1) + (1 - p_a) f_a(k, m - 1) \\
\leq\ & p_a e^{h(k-1)} (1 - p_a + p_a e^{-h})^{m-1} + \\
& (1 - p_a) e^{hk} (1 - p_a + p_a e^{-h})^{m-1} \\
=\ & e^{hk} (1 - p_a + p_a e^{-h})^{m-1} [p_a e^{-h} + (1 - p_a)] \\
=\ & e^{hk} (1 - p_a + p_a e^{-h})^m.
\end{aligned}$$

By picking h to be the one that minimizes $f_a(k, m)$, the last quantity becomes equal to $f_a(k, m)$. □

LEMMA 6. *f_a bounds the failure probability.*

PROOF. The first few steps of the proof of Chernoff bounds gives this. □

LEMMA 7.

$$f_a < \exp\left(\frac{-(p_a m - k)^2}{2 p_a m} \right).$$

PROOF. This follows from the remaining steps. □

The bound in Lemma 7 is the same as that obtained by applying Chernoff bounds to bound the probability of failure of PURE-RANDOM. We now state the Chernoff bounds (Lemma 8, the multiplicative versions) and an equivalent form (Corollary 9) that we find more suitable.

Chernoff bounds: Let $X = \sum_a X_a$ where $X_a \in [0, 1]$ are independent random variables. Let $\mu = E[X]$.

LEMMA 8. *For all $\epsilon > 0$,*

$$\mathbf{Pr}[X < \mu(1 - \epsilon)] \leq \exp\left(\frac{-\epsilon^2 \mu}{2} \right),$$

$$\mathbf{Pr}[X > \mu(1 + \epsilon)] \leq \exp\left(\frac{-\epsilon^2 \mu}{2 + \epsilon} \right).$$

COROLLARY 9. *For all $\delta > 0$, with probability at least $1 - \delta$, $-\mathcal{D}_l(\mu, \delta) \leq X - \mu \leq \mathcal{D}_h(\mu, \delta)$ where*

$$\mathcal{D}_l(\mu, \delta) = \sqrt{2\mu \ln\left(\frac{1}{\delta}\right)} \quad \textit{and}$$

$$\mathcal{D}_h(\mu, \delta) = \sqrt{2\mu \ln\left(\frac{1}{\delta}\right)} + \frac{1}{2} \ln\left(\frac{1}{\delta}\right).$$

PROOF. Plug in $\epsilon = \sqrt{\frac{2}{\mu} \ln\left(\frac{1}{\delta}\right)}$ and $\epsilon = \sqrt{\frac{2}{\mu} \ln\left(\frac{1}{\delta}\right)} + \frac{1}{2\mu} \ln\left(\frac{1}{\delta}\right)$ in the Chernoff bounds above. □

Choice of Initial Targets

Suppose we are in the YES case. If we run the PURE-RANDOM algorithm and X_a is the random variable indicating the number of impressions assigned to advertiser a by the algorithm, then after m steps, with probability $1 - \delta$,

$$\text{for all } a, X_a \geq \dot{m} p_a - \mathcal{D}_l(\dot{m} p_a, \delta/n). \tag{1}$$

As we showed using the Hybrid argument, the same statement is also true for the WATER-LEVEL algorithm. Thus we set the initial target to be $\dot{k}_a = \dot{m} p_a - \mathcal{D}_l(\dot{m} p_a, \delta/n)$.

Choice of number of samples $\dot{m}$

Now how large does m have to be so that we can distinguish between the YES and the NO cases? Suppose we are in the NO case. Then there exists a set $S \subseteq A$ such that $|\Gamma(S)| < B(S)(1-\epsilon)$. Let $Y(S)$ be the random variable indicating the number of impressions drawn from $\Gamma(S)$. $Y(S)$ is an upper bound on the total number of impressions allocated to the advertisers in S, by any algorithm. Let $q(S) = |\Gamma(S)|/|I|$ be the probability that an impression is in $\Gamma(S)$. Let $p(S) = \sum_{a\in S} p_a = \sum_{i\in S} B_a/|I|$. Note that in the NO case, $q(S) \leq p(S)(1-\epsilon)$. Then applying the concentration inequality to $Y(S)$ we get that with probability $\geq 1-\delta$,

$$Y(S) \leq \dot{m}q(S) + \mathcal{D}(\dot{m}q(S), \delta) \leq \dot{m}p(S)(1-\epsilon) + \mathcal{D}(\dot{m}p(S), \delta). \quad (2)$$

We pick $\dot{m}$ such that Inequalities (1) and (2) cannot both be true. A simple choice is to pick $\dot{m}$ such that

$$\mathcal{D}(\dot{m}p_a, \delta/n) \leq \epsilon\dot{m}p_a/2 \text{ for all } i, \text{and} \quad (3)$$

$$\mathcal{D}(\dot{m}p(S), \delta) \leq \epsilon\dot{m}p(S)/2 \text{ for all } S. \quad (4)$$

In that case, $Y(S) < \dot{m}p(S)(1-\epsilon/2)$ and hence there exists an $i \in S$ such that the number of impressions assigned to a is less than $mp_a(1-\epsilon/2)$. On the other hand, (1) and (3) along with the Hybrid argument imply that for all $a \in A$, at least $\dot{m}p_a(1-\epsilon/2)$ impressions are assigned to it.

(3) and (4) are achieved (with probability $1-\delta$) by picking

$$\dot{m} = \frac{8}{min_a\{p_a\}\epsilon^2} \ln\left(\frac{2n}{\delta}\right).$$

One could optimize for $\dot{m}$ by making sure $\mathcal{D}(\dot{m}p_a, \delta/n) \leq \epsilon_1\dot{m}p_a$ and $\mathcal{D}(\dot{m}p(S), \delta) \leq \epsilon_2\dot{m}p(S)$ and $\epsilon_1 + \epsilon_2 = \epsilon$. This only saves us a constant factor, so we leave out the detailed calculations here. The simple bound corresponds to the choice of $\epsilon_1 = \epsilon_2 = 1/2$. In fact this choice is better than optimizing as mentioned above, because this choice is biased towards the YES case. That is, if we are in the YES case, then we can detect it with fewer samples than if we are in the NO case. This is desirable for us since we expect to be in the YES case more often than in the NO case.

3.1 Existence Problem: Experimental Evaluation

We evaluated the accuracy of our algorithm using known NO and known YES matching instances. To create realistic instances, we modified the impression goals of a real set of advertisers booked against a set of Microsoft properties for a particular one-month period. The algorithm to create these instances proceeded as follows. For each instance, we first chose a random subset S of the advertisers[4] to be the "bottleneck" set. Next, we generated impressions using our traffic model and assigned those impressions to orders using the following rule: if the impression was targeted by an advertiser in S, assign the impression to any advertiser in S with the fewest assigned impressions; otherwise, assign the impression to any advertiser (not in S) with the fewest assigned impressions. Let d_a denote the number of impressions assigned to advertiser a using this procedure. To create a YES instance, we simply set the impression goal B_a of each advertiser a to be $(1-\epsilon)d_a$. In this case we "under-book" the capacity of the system by the parameter ϵ. To create a NO instance, we set the impression goal B_a of each order $a \in S$ to be $d_a(1+\epsilon)$ for the given "overbooking" parameter ϵ; for the advertisers $a \notin S$, we set the impression goal B_a to be $d_a(1-\epsilon')$, where ϵ' was chosen so that the number of impressions booked is equal to the number of expected impressions[5].

Using the above procedure we created 50 YES instances and 50 NO instances with $\epsilon = 0.03$. The number of impressions for these instances was about 320×10^6, and the number of impressions targeted by the orders was in the range of 100×10^6 and 200×10^6 impressions. We simulated 5 independent trials of our algorithm for a fixed value of δ, namely 0.001, and number of samples between 25×10^3 and 450×10^3. The number of samples were chosen such that the running time for each instance was less than 6 seconds. In all, we simulated 700 runs of our algorithm. It turns out that at around 275×10^3 samples our algorithm exhibits a transition in behavior. For smaller number of samples our algorithm accepts all the YES instances, but on the NO instances it rejects only about 17% of the cases. This indicates that for small sample size our δ value was still too small. When the number of samples exceeded 300×10^3 samples we did not see any errors in the YES case or the NO case. When the number of samples is $\approx 275 \times 10^3$, for the YES case our algorithm made an error in 10.4% of the instances; for the NO case our algorithm made an error in 15.2% of the cases.

4. REPRESENTATION PROBLEM

The basis for our representation is an LP for minimizing a submodular function. The representation will be a solution to the dual of this LP, with an appropriate choice of a submodular function. In the rest of this section, we first present an LP that minimizes a given submodular function, and its dual (Section 4.1), and then show how with an appropriate choice of a submodular function this gives us a compact representation of a complete matching (Section 4.2)

4.1 LP to Minimize a Submodular function

Let $[n]$ be the set $\{1, 2, \ldots, n\}$. Let $2^{[n]}$ be the power set of $[n]$, that is, the set of all subsets of $[n]$. Let $\psi : 2^{[n]} \to \mathbb{R}$ be a submodular function, that is, for all $S, T \subseteq [n]$, $\psi(S) + \psi(T) \geq \psi(S \cup T) + \psi(S \cap T)$. Given a permutation $\pi \in S_n$, let $S(\pi, a) = \{\pi(1) \ldots \pi(a)\}$. By extension, let $S(\pi, 0) = \emptyset$. The LP that minimizes ψ is as follows.

$$\begin{aligned} &\text{minimize } z \quad (5)\\ &\text{s.t. for all } \pi \in S_n,\\ &z \geq \psi(\emptyset) + \sum_{a=1}^{n} x_{\pi(a)} \left(\psi(S(\pi, a)) - \psi(S(\pi, a-1))\right),\\ &\text{for all } a, 0 \leq x_a \leq 1. \end{aligned}$$

LEMMA 10. *LP (5) captures the problem of minimizing the submodular function ψ. That is, the optimum value of the LP is equal to $\min_{S\subseteq[n]}\{\psi(S)\}$ and the optimum solution to the LP gives a set that minimizes ψ.*

PROOF. We first show that for any given value of x_a's, the inequality corresponding to a permutation that orders the x_a's non-increasingly is the most constraining inequality of all (Lemma 11). Thus z can be set to be equal to the

[4]The size of S was one third of the total number of advertisers in our experiments

[5]Whenever no such ϵ' existed, we abandoned the instance.

RHS of this inequality. Further the RHS can be interpreted as a convex combination of ψ's by rearranging terms:

$$\psi(\emptyset) + \sum_{a=1}^{n} x_{\pi(a)} \left(\psi(S(\pi, a)) - \psi(S(\pi, a-1))\right) = \quad (6)$$

$$\sum_{a=0}^{n} \psi(S(\pi, a))(x_{\pi(a)} - x_{\pi(i+1)}),$$

with $x_{\pi(0)}$ being set to 1 and $x_{\pi(n+1)}$ being 0. Hence z is minimized if and only if this convex combination only includes those sets that minimize ψ. In particular, if there is a unique set S that minimizes ψ, then the optimum solution is to set $x_a = 1$ for all $a \in S$ and 0 otherwise. In that case, the RHS of (6) is just $\psi(S)$, for any permutation that ranks all elements in S before those not in S. □

LEMMA 11. *Given $0 \leq x_a \leq 1$, the permutation that ranks x_a's in a non-increasing order maximizes*

$$\sum_{a=1}^{n} x_{\pi(a)} \left(\psi(S(\pi, a)) - \psi(S(\pi, a-1))\right).$$

PROOF. We will just show that for all $a, b \in [n]$ and $S \subseteq [n]$ such that $a, b \notin S$, $x_a > x_b$ if and only if

$$x_a \left(\psi(S \cup a) - \psi(S)\right) + x_b \left(\psi(S \cup a \cup b) - \psi(S \cup a)\right) \geq$$

$$x_b \left(\psi(S \cup b) - \psi(S)\right) + x_a \left(\psi(S \cup a \cup b) - \psi(S \cup b)\right).$$

It is easy to see that this is sufficient to prove the lemma: for a given permutation, let a and b be consecutively ranked elements (a is before b) such that $x_a < x_b$, and S is the set of all elements ranked before a. The above inequality tells us that we could swap a and b in the ranking without decreasing the quantity we wish to maximize.

The inequality we need is the same as the following, by rearranging terms:

$$x_a \left(\psi(S \cup a) - \psi(S) - \psi(S \cup a \cup b) + \psi(S \cup b)\right) \geq$$
$$x_b \left(\psi(S \cup a) - \psi(S) - \psi(S \cup a \cup b) + \psi(S \cup b)\right).$$

The lemma now follows from the fact that $\psi(S \cup a) - \psi(S) - \psi(S \cup a \cup b) + \psi(S \cup b)$ is non-negative, which is by the definition of submodularity of ψ. □

An immediate corollary of Lemma 11 is as follows.

COROLLARY 12. *Given an oracle access to ψ, there is a polynomial (in n) time separation oracle for LP (5), even though it has exponentially many constraints. Hence one can use the ellipsoid algorithm to solve LP (5) in time polynomial in n.*

The dual of LP (5) is as follows. Recall that $S(\pi, a) = \{\pi(1) \ldots \pi(a)\}$. $S(\pi, \pi^{-1}(a))$ is the set of all elements that are ranked up to (and including) a in the permutation π. $S(\pi, \pi^{-1}(a) - 1)$ is the set of all elements that are ranked before (and not including) a in the permutation π.

$$\text{maximize } \psi(\emptyset) \sum_{\pi} \beta_\pi - \sum_{a} \alpha_a \quad (7)$$

s.t. for all i,

$$\alpha_a + \sum_{\pi} \beta_\pi \left[\psi(S(\pi, \pi^{-1}(a))) - \psi(S(\pi, \pi^{-1}(a) - 1))\right] \geq 0,$$

$$\sum_{\pi} \beta_\pi \leq 1.$$

$$\alpha_a \geq 0, \beta_\pi \geq 0.$$

Note that the β_π's define a solution to LP (7) since each α_a occurs in exactly one constraint (in addition to the constraint $\alpha_a \geq 0$), and hence we either set α_a so that the constraints holds with equality, or set $\alpha_a = 0$.

4.2 Using LP (7) to Represent a Matching

We now show that with a natural choice for ψ, the β_π's represent a complete matching if there exists one. In fact more generally, for any matching $M \subseteq E$ such that $|M \cap \delta(i)| = 1$ for all $i \in I$, define profit$(M) := \sum_a \min\{B_a, |M \cap \delta(a)|\}$. A complete matching is one whose profit is $\sum_a B_a$. Extend the above definition to fractional matchings as well: $\{x_{ai}\}_{(a,i) \in E}$ is a fractional matching if $\sum_a x_{ai} = 1$ for all $i \in I$ and the profit of such a matching is $\sum_a \min\{B_a, \sum_i x_{ai}\}$. Then we show that LP (7) with ψ appropriately defined finds a maximum profit matching.

Define ψ as follows: for any $S \subseteq A$,

$$\psi(S) = \sum_{a \in S^c} B_a + |\Gamma(S)|,$$

where $\Gamma(S)$ is the set of neighbors of S in the given graph G. We show the following:

1. ψ is submodular (Lemma 13).
2. The optimal value of LP (7) $= \min_S\{\psi(S)\}$ $= \max_M\{\text{profit}(M)\}$ = maximum profit of a fractional matching (Lemma 14).
3. A solution β_π to LP (7) corresponds to a fractional matching with profit equal to the objective function of the LP (Lemma 15).

From the above, it follows that LP (7) finds a maximum profit matching. In particular, if a graph has a complete matching, then it can be represented using β_π's. We will later show that one can even guarantee that there exists a representation of small size, and it can be computed in polynomial time.

LEMMA 13. *ψ is submodular.*

PROOF. Proof is standard. The terms corresponding to B_a's are "linear". So if we let $B(S) = \sum_{a \in S} B_a$, then

$$B(S) + B(T) = B(S \cup T) + B(S \cap T), \text{ and equivalently,}$$

$$\begin{aligned} B(S^c) + B(T^c) &= B(S^c \cup T^c) + B(S^c \cap T^c) \\ &= B((S \cap T)^c) + B((S \cup T)^c). \end{aligned}$$

It remains to show that the Γ terms are submodular. This follows from the observations that $\Gamma(S \cup T) = \Gamma(S) \cup \Gamma(T)$ and $\Gamma(S \cap T) \subseteq \Gamma(S) \cap \Gamma(T)$. □

LEMMA 14. *The optimal value of LP (7) $= \min_S\{\psi(S)\} = \max_M\{$profit$(M)\}$ = maximum profit of a fractional matching.*

PROOF. Lemma 10 implies that the optimum value of LP (7) is equal to $\min_S\{\psi(S)\}$. Note that ψ naturally corresponds to the weight of a vertex cover, namely that of $S^c \cup \Gamma(S)$, with weight of $a \in A$ being B_a and weight of $i \in I$ being 1. Thus minimizing ψ is equivalent to finding a minimum weight vertex cover. Using standard max-flow/min-cut technology one can show that a minimum weight of a vertex cover is equal to the maximum profit of a matching which is equal to the maximum profit of a *fractional* matching. □

LEMMA 15. *A solution β_π to LP (7) corresponds to a fractional matching with profit equal to the objective function of the LP.*

PROOF. We first define an integral matching M_π given a permutation π. Given a solution β_π to LP (7), the corresponding fractional matching is $\sum_\pi \beta_\pi M_\pi$. We also show that at the optimum, $\sum_\pi \beta_\pi$ is always equal to 1, and hence the above is a *convex* combination of integral matchings. Finally we show that the objective function for a given solution is equal to the profit of the corresponding fractional matching.

Define M_π as follows: all impressions adjacent to $\pi(1)$ are matched to $\pi(1)$. All impressions in the set $\Gamma(\{\pi(1), \pi(2)\}) \setminus \Gamma(\{\pi(1)\})$ are matched to $\pi(2)$, and so on, all impressions in the set $\Gamma(S(\pi, \pi^{-1}(a))) \setminus \Gamma(S(\pi, \pi^{-1}(a)-1))$ are assigned to a. Equivalently, an impression i is assigned to $\arg\min_{a:(a,i)\in E}\{\pi^{-1}(a)\}$, the earliest ranked advertiser adjacent to i.

For any feasible solution, multiplying all the β_π's by the same positive number maintains feasibility of all constraints except that of $\sum_\pi \beta_\pi \leq 1$. The objective function is also multiplied by the same number. Thus at the optimum $\sum_\pi \beta_\pi = 1$.

To relate the profit of the matching $\sum_\pi \beta_\pi M_\pi$ to the objective function $\psi(\emptyset)\sum_\pi \beta_\pi - \sum_a \alpha_a$, we first reformulate the constraints in LP (7). The constraint $\sum_\pi \beta_\pi[\psi(S(\pi, \pi^{-1}(a))) - \psi(S(\pi, \pi^{-1}(a)-1))] + \alpha_a \geq 0$ is equivalent to

$$\sum_\pi \beta_\pi \left[|\Gamma(S(\pi, \pi^{-1}(a)))| - |\Gamma(S(\pi, \pi^{-1}(a)-1))|\right] \geq B_a - \alpha_a.$$

The LHS of this constraint now corresponds to $\sum_i x_{ai}$ where $\{x_{ai}\}_{(a,i)\in E}$ is the fractional matching given by $\sum_\pi \beta_\pi M_\pi$. This is because $|\Gamma(S(\pi, \pi^{-1}(a)))| - |\Gamma(S(\pi, \pi^{-1}(a)-1))| = |M_\pi \cap \delta(a)|$, the number of impressions allocated to a in M_π. Thus $B_a - \alpha_a = \min\{B_a, \sum_i x_{ai}\}$. Since $\psi(\emptyset) = \sum_a B_a$, by setting $\sum_\pi \beta_\pi = 1$, we get that the objective function is equal to $\sum_a (B_a - \alpha_a) = \sum_a \min\{B_a, \sum_i x_{ai}\} = \text{profit}(\sum_\pi \beta_\pi M_\pi)$.

□

This implies that one can always represent a (fractional) maximum matching as a convex combination of M_π's. But we might need a large number of M_π's to do so (potentially all $n!$ of them). However, we now show that we only need $n+1$ of them.

LEMMA 16. *There exists a maximum fractional matching that can be represented as a convex combination of at most $n+1$ M_π's. Moreover, such a representation can be found in time polynomial in n.*

PROOF. We show that any vertex solution of LP (7) has at most $n+1$ non-zero β_π's. LP (7) has $n!+n$ variables, and thus a vertex solution to this LP has $n!+n$ tight constraints. Suppose we wish to maximize the number of non-zero β_π's, then we should minimize the number of tight constraints of the form $\beta_\pi = 0$. Hence we try to set as many of the other constraints to be tight as possible. But the number of other constraints is only $2n+1$. Hence the number of tight constraints of the form $\beta_\pi = 0$ is at least $n! - n - 1$ which implies that the number of non-zero β_π's is at most $n+1$.

We cannot use the ellipsoid algorithm directly to solve LP (7). But we can use the following stadard technique: we first solve LP (5) using the ellipsoid algorithm (Corollary 12) and consider the constraints that appear as separating hyperplanes during its run. Since the ellipsoid algorithm runs in polynomial time, there are only polynomially such constraints. We now solve LP (7) with only those $\beta'_\pi s$ that correspond to these constraints (with all others being set to 0). This does not change the optimum value of LPs (5) and (7). □

A very interesting fact is that the set of optimal fractional matchings obtained this way (as $\sum_\pi \beta_\pi M_\pi$) does not contain any integral matchings, unless there is a unique integral matching! In fact this gives us the following corollary.

COROLLARY 17. *If a bipartite graph has a unique perfect matching then it is a subgraph of the "half graph".*

5. CONCLUDING REMARKS

A natural open direction that our approach suggests is to analyze the following Waterlevel algorithm for the existence problem: assign i to the advertiser maximizing k_a/B_a. This would be the "natural generalization" of the algorithm in Motwani et. al. [7], that is, assign to the advertiser with the highest k_a. (They consider the case when the B_a's are all the same.) In fact, the proof of Motwani et. al. seems to generalize to this case also, at first sight, but on closer examination, it can be seen that it does not. The proof cosntructs a coupling (essentially a bijective mapping) between the random choices made by PURE-RANDOM and the random choices made by the Waterlevel algorithm. It is shown inductively that under this mapping, Waterlevel is always more balanced than PURE-RANDOM. To be more precise, for every choice that PURE-RANDOM makes, the coupling gives a choice for Waterlevel that keeps it more balanced. However, if the B_a's are different, then the same impression contributes to different fractions of B_a for different advertisers and as a result the inductive step fails. In particular, suppose that the distribution of the fractions is the same for both algorithms, and that there are two choices for a given impression with the same ratio k_a/B_a but different B_a's. By breaking ties accordingly, waterlevel can be made to be given to that advertiser with a larger B_a, thus making it "less balanced" than PR which gives it to the one with the smaller B_a. One can also verify that the proof of Motwani et. al. works against a stronger "adversary", that is even if at every step an impression is drawn from a different graph, as long as all graphs have a complete matching. However, if the sizes are different, then one can show that the natural generalization will fail against the stronger adversary. This indicates that a different proof is needed if this algorithm indeed works.

Also another open problem is to design a fast (possible combinatorial) algorithm to compute a representation of a matching as guaranteed by Lemma 16, instead of using the ellipsoid algorithm

6. REFERENCES

[1] Ravindra K. Ahuja, James B. Orlin, Clifford Stein, and Robert Endre Tarjan. Improved algorithms for bipartite network flow. *SIAM J. Comput.*, 23(5):906–933, 1994.

[2] Nikhil R. Devanur and Thomas P. Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In John Chuang,

Lance Fortnow, and Pearl Pu, editors, *ACM Conference on Electronic Commerce*, pages 71–78. ACM, 2009.

[3] L.R. Ford and D.R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[4] Arpita Ghosh, Preston McAfee, Kishore Papineni, and Sergei Vassilvitskii. Bidding for representative allocations for display advertising. In *WINE*, pages 208–219, 2009.

[5] Andrew V. Goldberg and Satish Rao. Beyond the flow decomposition barrier. *J. ACM*, 45(5):783–797, 1998.

[6] Sébastien Lahaie, David C. Parkes, and David M. Pennock. An expressive auction design for online display advertising. In Dieter Fox and Carla P. Gomes, editors, *AAAI*, pages 108–113. AAAI Press, 2008.

[7] Rajeev Motwani, Rina Panigrahy, and Ying Xu 0002. Fractional matching via balls-and-bins. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 487–498. Springer, 2006.

[8] Erik Vee, Sergei Vassilvitskiiy, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. Technical Report 05, Yahoo! Labs, 2009.