Theory Research at Google

Gagan Aggarwal, Nir Ailon, Florin Constantin, Eyal Even-Dar, Jon Feldman
Gereon Frahling Monika R. Henzinger, S. Muthukrishnan, Noam Nisan
Martin Pál, Mark Sandler, Anastasios Sidiropoulos

Introduction

Through the history of Computer Science, new technologies have emerged and generated fundamental problems of interest to theoretical computer scientists. From the era of telecommunications to computing and now, the Internet and the web, there are many such examples. This article is derived from the emergence of web search and associated technologies, and focuses on the problems of research interest to theoretical computer scientists that arise, in particular at Google.

Google presents a unique work environment for research.

- Google's business comprises making accessible the information on the web, in whatever form. Hence, there is a need for researchers to understand individual elements of text, speech, images, video and other media, as well as translate and transform between them for a unified search, while avoiding agents that attempt to manipulate the process, for example, by spam. Google's business relies on revenue from placing advertisements in response to search at its site, typically via online auctions; increasingly, they place advertisements in other media including, other online sites, TV, Radio and videos. This calls for research in economic and game theory of auctions, solving large scale optimization problems, as well as, algorithms for statistical estimation of a number of parameters that are needed to run the auctions. Also, Google's business has evolved to include several web-based hosted services which are constantly growing. Taken together, thus, Google's business presents rich research opportunities.
- Engineers at Google build a lot of the systems they need, rather than buy them from external vendors. For example, they have built their own file systems (GFS), distributed computing platform (MapReduce), parallel data analysis language (Sawzall), and data storage system (BigTable). Thus computer scientists find many research challenges in the systems where they can immediately impact Google's business.
- The scale of the operations is astounding from the number of web searches, video downloads to ad clicks and map searches in a day, and the scale keeps growing. Thus, there is a constant need for innovative and efficient solutions; also, one may indeed care about certain asymptotics.
- Finally, the environment at Google encourages researchers to work closely with the engineering teams. Therefore, real needs motivate many of the research projects and solutions can have an impact in a short duration.

Given this context, theoretical computer scientists like us at Google tend to think about how to apply emerging ideas to the problems at Google, as well as study the fundamental challenges in algorithmic research generated by the technologies at Google. This article gives a selection of problems we study. Like typical of our engineering environment that gets products out as quickly it can, the results here are also being shared rapidly. Many of them represent snippets of ongoing research, some under submission, and most will appear in a detailed form only in the future.

The article has sections written by different researchers. They have been roughly grouped into three sections:

- *Distributed Systems.* Google relies on a distributed approach for nearly all tasks associated with web search as well as data analysis and others. Section 1.1 discusses load balancing challenges in web search engines, and Section 1.2 discusses a model for MapReduce, the distributed computing platform at Google.

- *Algorithms.* Researchers design algorithmic results for specific graph, text, geometric and learning problems. Section 2.1 discusses a problem of inferring labels on graph nodes; Section 2.2 discusses the connection between ranking problems and machine learning; Section 2.3 discusses hierarchical mixture models; Section 2.4 discusses hierarchical clustering.

- *Game theory.* There is significant research effort on the economic and game theory of auctions for placing advertisements on online properties and elsewhere. Section 3.1 discusses the auction for sponsored search and its analyses; Section 3.2 and Section 3.3 discuss a more detailed view of sponsored search that involves user modeling and the effect of budgets respectively; Section 3.4 discussions game theory of ad reservations in advance.

Unfortunately, due to space constraints, only a selection of results could be presented here and we had to omit several other examples of theoretical computer science research at Google. Also, the business changes rapidly and as such new, exciting problems emerge all the time; hence, theory research will continue to change and grow over time, perhaps differently from the snapshot represented here.

1 Distributed Systems

1.1 Loadbalancing Challenges in Web Search Engines

M.R. Henzinger

Search engines use a data structure called *(inverted) index* to answer user queries. For web search engines this index is too large to fit on a single machine and has to be broken into *subindices* that are stored on many machines. Consequently, each user query is converted into many requests, each requiring access to a single subindex. The search engine has complete freedom in building these subindices and in assigning them to machines. Usually the subindices are created to fulfill the following conditions: (A) All subindices need the same amount of space and about 4-10 subindices fit on a machine. (B) Over a long enough period of time (like a day) the computational load on the subindices generated by a typical sequence of requests is roughly balanced. (C) Individual requests have time-outs of a few seconds, i.e., if a request cannot be answered in a few seconds it is terminated. Thus the computational load of an individual request is a tiny constant.

This problem is different from the load balancing problems studied in the past for two reasons.

(1) Prior work assumed either that every file is stored on every machine (identical machine model) or that the *adversary* can control for *each* individual request on which machine the request can be placed (restricted assignment model.) In our model the file assignment is under the control of the *algorithm*, but has to be fixed once at the beginning of the request sequence.

(2) Many of the lower bounds in prior work depend on the fact that individual requests can be large, i.e., they can significantly increase the machine load of a machine. In our setting this cannot happen.

We model this situation as a problem with two phases: The input for Phase (1) is the number n of files that need to be assigned and the number m of machines that are available. For each machine m_i the number s_i of slots of the machine is specified. Assigning a file to a machine requires exactly one slot. No further information is given. The output of Phase (1) is an assignment of files to machine such that (a) every file is assigned to at least one machine and (b) the number of files assigned to a machine does not exceed its slots. To fulfill these conditions $\sum_i s_i$ needs to be at least n. A file can be assigned to more than one machine if there is a sufficient number of slots.

The input to Phase (2) is the assignment of files to machines computed in Phase (1) and a sequence R of requests that arrives in an online manner. Each request r_k is a pair (f, l), where f is the file that the request needs to access and l is the load of the request. As soon as it arrives the request needs to be placed on a machine to which file f was assigned. It increases the load of this machine by l. This implicitely assumed that all machines have identical speed. An interesting variante is the setting where the amount of load a request places on a machine depends on the machine, i.e., it can be different for different machines.

In pratice the goal is to maximize throughput. In online loadbalancing this is often modeled as spreading the load as equally as possible over the set of machines. Let the *machine load* ML_i of m_i be the sum of the load of the requests placed on m_i. One popular metric to achieve "well-spread" load distribution is to try to minimize the maximum machine load. However, other metrics can be used as well. If we adopt the maximum machine load as the metric of choice, then our goal is to design an offline file assignment algorithm for Phase (1) and a online request assignment algorithm for Phase (2) such as to minimize the maximum machine load. This maximum machine load can then be compared with the maximum machine load achieved by an optimal offline algorithm, i.e., an algorithm that knew the full sequence of requests *before* Phase (1). Thus, we are comparing against an offline algorithm that might use a *different* file assignment than the one produced in our Phase (1).

As we show below the competitive ratio that can be achieved against such an optimal algorithm depends on the number of slots available. Thus, the goal is to study the trade-off between the number of slots used in Phase (1) and the competitive ratio that can be achieved in Phase (2).

We studied so far the case where all machines have the same number of slots and where the goal is to minimize the maximum machine load. Let the *file load* FL_j be the sum of the loads of the requests for file f_j. In the search engine setting the file loads are roughly balanced. Thus we assumed that there is a constant $\alpha > 0$ such that for all j, $FL_j \leq (1+\alpha)min_jFL_j$. Furthermore, in web search engines the load of individual requests is a tiny constant. Thus we assumed that $\beta << min_jFL_j$ and $\beta << min_iML_i$, where β is the largest load of any request. As a result when reporting competitive ratios we ignore additive β factors, i.e., when we report a competitive ratio of r, the actually maximum machine load is at most $rOPT + O(\beta)$.

When every file is stored on every machine, i.e,. nm slots are used, a competitive ratio of 1 can be achieved by the greedy request assignment algorithm. On the other side, with with a linear number of slots, more specifically with $n + m - 1$ slots, there exists a file assignment algorithm so that the greedy request assignment algorithm is $1 + \alpha$-competitive. More generally, the following lemma shows the trade-off between the number of slots and the competitive ratio that can be achieved by the greedy request assignment algorithm.

Lemma 1. *Let g be a function of m such that for all values of m, $g(m)$ is a natural number that divides n and m. With $nm/g(m)$ slots a competitive ratio of $1 + (1 - (1 + \alpha)/(g(m) + \alpha)\alpha$ can be achieved.*

An interesting question is how close to 1 the competitive ratio can be pushed with just a linear number of slots.

1.2 On Distributing Symmetric Streaming Computations

Jon Feldman

We now have truly massive data sets, many of which are generated by logging events in physical systems. For example, data sources such as IP traffic logs, web page repositories, search query logs, and retail and financial transactions, consist of billions of items per day, and are accumulated over many days. In theory, the *data stream model* facilitates the study of algorithms that process such truly massive data sets. Data stream models [2, 6] make one pass over the logs, read and process each item on the stream rapidly and use local storage of size sublinear—typically, polylogarithmic—in the input. There is now a large body of algorithms and lower bounds in data stream models (see [7] for a survey).

Yet, streaming models alone are not sufficient. For example, logs of Internet activity are so large that no single processor can make even a single pass over the data in a reasonable amount of time. Therefore to accomplish even a simple task we need to distribute the computations. This distribution poses numerous challenges, both theoretical and practical. In theory, the streaming model is highly sequential and one needs to design distributed versions of algorithms. In practice, one has to deal with data distribution, synchronization, load balancing, processor failures, etc. Distributed systems such as Google's MapReduce [4] and Apache's Hadoop [3] are successful large scale platforms that can process many terabytes of data at a time, distributed over hundreds or even thousands of machines, and process hundreds of such analyses each day. One reason for their success is that algorithms written for these platforms have a simple form that allow the machines to process the input in an arbitrary order, and combine partial computations using whatever communication pattern is convenient.

The fundamental question that arises is: Does the class of computational tasks supported by these systems differ from the class for which streaming solutions exist? That is, successful though these systems may be in practice, does using multiple machines (rather than a single streaming process) inherently limit the set of possible computations?

Massive, unordered, distributed (mud) model of computation: In [5], the authors introduce the following simple algorithmic model for massive, unordered, distributed (mud) computation, as implemented by these systems.

A *mud algorithm* is a triple $m = (\Phi, \oplus, \eta)$, where the function $\Phi : \Sigma \rightarrow Q$ maps an input item to a message, the aggregator $\oplus : Q \times Q \rightarrow Q$ maps two messages to a single message, and the post-processing operator $\eta : Q \rightarrow \Sigma$ produces the final output. Formally for an input $\mathbf{x} \in \Sigma^n$, the mud algorithm outputs

$$m(\mathbf{x}) = \eta(\Phi(x_1) \oplus \Phi(x_2) \oplus \cdots \oplus \Phi(x_n)).$$

Note that the order in which the $\oplus$ operations is applied is not specified; a mud algorithm *computes* a function $f : \Sigma^n \rightarrow \Sigma$ if for all $\mathbf{x} \in \Sigma$, we have $m(\mathbf{x}) = f(\mathbf{x})$ for *all possible topologies* of $\oplus$ operations. This topology-independent property is essential for being able to execute the algorithm in a highly distributed setting, combining partial computations using arbitrary communication patterns. In contrast, a *streaming algorithm* can be defined in the same way, except that it is only required to compute f for a sequential application of $\oplus$ operations. The communication complexity of a mud or a streaming algorithm is $\log |Q|$, the number of bits needed to represent a "message" from one component to the next. The {space, time} complexity of a mud or a streaming algorithm is the maximum {space, time} complexity of its component functions Φ, $\oplus$ and η.

Comparing mud algorithms to streaming algorithms: If a function can be computed by a mud algorithm, it can clearly also be computed by a streaming algorithm (using the same component functions) with the same {time,space,communication} complexity. The central question then is, *can any function computable by a streaming algorithm also be computed by a mud algorithm?* The immediate answer is clearly no. For example, consider a streaming algorithm that counts the number of occurrences of the *first* element x_1 in the stream: no mud algorithm can accomplish this since it cannot determine the first element in the input. Therefore, in order to be fair, since mud algorithms work on unordered data, we restrict our attention to functions $\Sigma^n \rightarrow \Sigma$ that are *symmetric* (order-invariant). With this restriction, we get the following result:

Theorem 1. *For any symmetric function $f : \Sigma^n \rightarrow \Sigma$ computed by a $g(n)$-space, $c(n)$-communication streaming algorithm, with $g(n) = \Omega(\log n)$ and $c(n) = \Omega(\log n)$, there exists a $O(c(n))$-communication, $O(g^2(n))$-space mud algorithm that also computes f.*

This result generalizes to certain approximation algorithms, and randomized algorithms with public randomness. We also show that the claim above does not extend to richer symmetric function classes, such as when the function comes with a *promise* that the domain is guaranteed to satisfy some property (e.g., finding the diameter of a graph known to be connected), or the function is *indeterminate*, i.e., one of many possible outputs is allowed for "successful computation." (e.g., finding a number in the highest 10% of a set of numbers.) Likewise, with private randomness, the claim above is no longer true. (See [5] for details.)

Techniques: One of the core arguments used to prove our positive results comes from an observation in communication complexity. Consider evaluating a symmetric function $f(\mathbf{x})$ given two disjoint portions of the input $\mathbf{x} = \mathbf{x}_A \cdot \mathbf{x}_B$, in each of the two following models. In the *one-way communication model* (OCM), David knows portion $\mathbf{x}_A$, and sends a single message $D(\mathbf{x}_A)$ to

Emily who knows portion $\mathbf{x}_B$; she then outputs $E(D(\mathbf{x}_A), \mathbf{x}_B) = f(\mathbf{x}_A \cdot \mathbf{x}_B)$. In the *simultaneous communication model* (SCM) both Alice and Bob send a message $A(\mathbf{x}_A)$ and $B(\mathbf{x}_B)$ respectively, simultaneously to Carol who must compute $C(A(\mathbf{x}_A), B(\mathbf{x}_B)) = f(\mathbf{x}_A \cdot \mathbf{x}_B)$. Clearly, OCM protocols can simulate SCM protocols. Is SCM significantly weaker than OCM? This is a simplification of our question concerning mud vs. streaming.

We can simulate an OCM process in the SCM as follows. Both Alice and Bob imagine that they are David, and send messages $D(\mathbf{x}_A)$ and $D(\mathbf{x}_B)$. Carol would like to compute $E(D(\mathbf{x}_A), \mathbf{x}_B)$, but she only has $D(\mathbf{x}_B)$, and not $\mathbf{x}_B$ itself. So, she guesses a sequence $\mathbf{x}'_B$ such that $\mathbf{x}'_B = D(\mathbf{x}_B)$, and outputs $E(D(\mathbf{x}_A), \mathbf{x}'_B)$. Using the symmetry of f, it can be shown that Carol outputs $f(\mathbf{x}_A \cdot \mathbf{x}_B)$ correctly:

$$E(D(\mathbf{x}_A), \mathbf{x}'_B) = f(\mathbf{x}_A \cdot \mathbf{x}'_B) = f(\mathbf{x}'_B \cdot \mathbf{x}_A) = E(D(\mathbf{x}_B), \mathbf{x}_A) = f(\mathbf{x}_B \cdot \mathbf{x}_A) = f(\mathbf{x}_A \cdot \mathbf{x}_B).$$

To prove our main result—that mud can simulate streaming—we apply the above argument many times over an arbitrary tree topology of $\oplus$ computations, using Savitch's theorem to guess input sequences that match input states of streaming computations. This argument is delicate because we can use the symmetry of f only at the root of the tree; simply iterating the argument at each node in the computation tree independently would yield weaker results that would force the function to be symmetric on *subsets* of the input, which is not assumed by our theorem.

Discussion: The simulation in our result takes time $\Omega(2^{\text{polylog}(n)})$ from the use of Savitch's theorem. Therefore our simulation is not a practical solution for executing streaming algorithms on distributed systems; for any specific problem, one may design alternative mud algorithms that are more efficient or even practical. One of the implications of our result however is that any separation between mud algorithms and streaming algorithms for symmetric functions would require lower bounds based on time (not communication) complexity.

The MUD class includes many useful computations performed every day on massive data sets, but to fully capture the capabilities of the modern distributed systems such as MapReduce and Hadoop, we can generalize it in two different ways. First, we can allow multiple mud algorithms running simultaneously over the same input. This is implemented by computing (*key*, *value*) pairs for each input x_i, and then aggregating the values with the same key using the $\oplus$ function. Second, we can allow multiple rounds of computation, where each round is a mud algorithm, perhaps using multiple keys. Since each round constitutes a function $\Sigma^n \to \Sigma^{n'}$, mud algorithms naturally compose to produce an overall function $\Sigma^n \to \Sigma^{n'}$. Exploring the computational power of these extensions remains an important open direction for further research.

2 Algorithms

2.1 Graph Labeling

Anastasios Sidiropoulos

There are several interesting computational challenges related to the analysis of the web and other graphs that can be phrased as *classification* problems. For example, an important application scenario is the classification of webpages based on whether they contain spam or not. In the context of on-line advertising, the goal might be to identify the topic of a webpage, so that relevant ads can be displayed. One way of approaching these problems involves methods based on lexical and contextual analysis. An inherent limitation of such an approach is that it is hard to define what the objective really is, so it is also hard to compare different solutions. Moreover, in the case of the web and other graphs such as social networks, such solutions typically ignore the explicit information given by the links between objects.

A different approach that does not suffer from the above limitations is the following. Given a classification problem on a set of interlinked objects, one can define a *generative model* that for a set of classified objects, defines a *stochastic process* for generating the links between them. The probability of obtaining a set of links is therefore a function of the initial classification. In this setting, the classification problem can be phrased as a combinatorial optimization question: Given the links between the objects, compute the maximum-likelihood classification for the objects.

Perhaps the simplest possible generative model is this: We start with a set of vertices V, and a labeling $L : V \to [k]$. For each $i, j \in [k]$, the probability of having an edge from a vertex of label i to a vertex of label j is $p_{i,j}$, and each edge appears independently. We refer to this generative process as the *complete-information* model.

There are a few interesting facts that are either known, or can easily be derived for this model:

- Computing a labeling that maximizes the log-likelihood is NP-hard to approximate within any factor. This can be shown by an argument similar to the one due to Giotis and Guruswami for Correlation Clustering [30].
- Despite the above negative result, the problem is more interesting in the probabilistic case. That is, we would like to be able to recover the true labeling with high probability, for an instance chosen according to the generative process. It has been shown by McSherry [31] that this can be done efficiently.

A limitation of the above generative model is that is assumes that all the objects are related to each other. This is not true in most scenarios that appear in the graphs we derive from the web. A natural way of bypassing this restriction is to modify the model as follows. Initially we are given a graph H with a labeling $L : V(H) \to [k]$. Then, for each edge e of H, and independently, we remove e with probability $p_{i,j}$, where i and j are the labels of the end-points of e. We refer to this modified process as the *partial-information* model. Clearly, the complete-information model is the special case where H is the complete graph.

For partial-information model we can show the following (joint work with Jon Feldman and S. Muthukrishnan):

- Assuming the Unique Games conjecture, approximating the maximum log-likelihood labeling within any factor is NP-hard.
- In the probabilistic case, it is impossible to recover, say $(1-\epsilon)$-fraction of the labels with high probability for arbitrary graphs H. For example, if H has constant degree, then one can only hope to recover each vertex correctly with some probability bounded away from 1. However, in many application scenarios it is desirable to recover correctly as many labels as possible, even if this would only give a small advantage over random guessing. This seems to be an interesting and natural problem, and we can show that simple algorithms give non-trivial guarantees.

The above models are of course not the only natural ones. It would be interesting to have a theory for dealing with general classes of generative models. A challenging case seems to be models that allow dynamic graph generation, similar for example to preferential-attachment graphs.

2.2 On Ranking and Learning

Nir Ailon

The problem of *ordering* or *ranking* arises naturally in many domains such as sports, economics, politics, information retrieval, and mechanism design. This should come as no surprise: a linear order is probably the simplest, most intuitive structure that we endow on sets of items. We rank tasks based on their relative priority when we manage our time. We like our search engine results displayed in order of relevance, from top to bottom. Almost all clever data structures maintain ordered data invariantly.

Well known problems in the category of combinatorial optimization are digraph ranking (how do we rank chess players while respecting game outcome constraints?), linear arrangement (how do we order a set of elements while respecting proximity constraints?) and meta searching (how do we combine results from several search engines?) Interestingly, relevant research on the latter problem (albeit in a social choice and not IR setting) can be traced back to the 18th century, when Condorcet [17] and Borda [15] studied election systems where voters specify rankings of candidates. In this context we must also mention Arrow's celebrated result [13], explaining why in such a setting no voting scheme is satisfactory under natural social choice theoretical assumptions.

More recently, motivated by the growing need to organize unprecedented amounts of data accessed daily by millions, there has been interest in the machine learning problem of ranking. In testing, we are repeatedly given sets V of n items, and the goal is to rank these items based on information we learned earlier in training with full information. Unlike the above optimization problems, in testing we do not explicitly know the utility of a solution during test time. We only assume that V and the attached utility function are determined by a randomized mechanism consistent throughout training and testing. The best example for learning ranking is *information retrieval* (IR): Given a query and a set V of preliminary search results satisfying a rough matching criterion, we wish to return V to the client in some order, which should be as close as possible to an unknown correct relevance order.

Continuing the IR example, what type of information (labels) should we learn in training? One idea is to ask human evaluators to provide ratings for individual search results on an absolute scale, and then learn an absolute "relevance" function $f(u)$ on the universe of possible search results. In testing, we order the results based on their f value. The problem with this approach, as known by experimental psychologists and economists, is that humans' responses are not absolute and are influenced by exposure to irrelevant information, or *anchors*. For example Ariely et al. [12] showed that the task of pricing goods could be influenced by subjects' social security number. Therefore, we choose to ask for *comparison* labels on pairs of results u, v, corresponding to which results is better. Humans' responses are known to be more stable on such questions. Given such labels, the score based approach [19, 20] learns a relevance function f attempting to minimize a risk functional that adds a penalty whenever $f(u)$ is lower than $f(v)$ and there is a label expressing preference of u over v. The main complication with the score based approach is its dissonance with another risk function called AUC [22], a standard evaluation measure for IR systems.

A different, comparison based approach [14, 10, 16] suggests to learn instead a comparison function h on pairs. From a learning perspective, this is simply binary classification: the hypothesis $h(u, v)$ is ± 1, corresponding to whether u is better than v or vice versa. In a more general setting (as in [18]), the magnitude can encode confidence.

Definition. Given a prelearned comparison hypothesis h, a set V and a (possibly unknown) *true* comparison function σ induced by some correct ordering of V, we define an *idealized* risk as $R(h, \sigma) = \sum \delta_{h(u,v),\sigma(v,u)}$, where δ is the Kronecker delta function and the sum is over a (possibly unknown) subset P of pairs of elements in V.

In words, the function pays 1 for any disagreement of pairs in P. If P is the set of all pairs in V, we call it general. It turns out that there is a simple way to choose a strict subset P such that the resulting risk is compatible with AUC. We refer to this as the AUC case.

The reason $R(h, \sigma)$ is called *idealized* is because it may not be transitive (think *rock, paper, scissors*), and hence we may not be able to output it. So how do we use h? One idea [16] is to try find a ranking π minimizing $R(\pi, h)$, but this is NP-Hard [11], and what we *really* care about is $R(\pi, \sigma)$. Improving on earlier work [14], Ailon and Mohri recently showed [10] the following:

Theorem 1. Running QuickSort on V, using h as a comparison black-box, outputs a (random) ranking π such that the $E[R(\pi, \sigma)] = R(h, \sigma)$ in the AUC case, and $E[R(\pi, \sigma)] \leq 2R(h, \sigma)$ in the general case.

We reduced learning ranking to binary classification, measured our success using standard AUC, used stable comparison based labels from humans, and lost nothing in the reduction. On top of that, a testing round involves only $O(n \log n)$ invocations of h on expectation.

There are several notes in place. The half century old QuickSort algorithm [21], initially designed as an efficient divide-and-conquer sorting algorithm, has recently found new merits not only in learning, but also in combinatorial optimization [8, 9]. A *local ratio* technique is used in the analyses. The fact that QuickSort is randomized raises the question of derandomization. Based on recent work [23] this intuitively seems possible. However, Ailon and Mohri proved [10]:

Theorem 2. Any deterministic algorithm from h to a ranking on V outputs a solution with risk R at least twice that of h with respect to some true ranking, even in the AUC case.

In other words, derandomization is impossible in the learning setting. We point out a curious connection to Arrow's work noting that determinism is one of the social choice theoretical assumptions leading to his impossibility result. A more refined tradeoff between randomization and quality is an interesting open research direction.

2.3 Hierarchical Mixture Models: a Framework to Describe Large Text Collections

Mark Sandler

Introduction Text classification is one of the most fundamental and well studied problems in modern information retrieval. However, despite the vast amount of work in the area, there is relatively little work on formal models describing the underlying processes which cause texts to belong to one topic or another. In the recent years a few approaches have emerged. Perhaps the most notable work here is Principal Component Analysis, that relies on singular value decomposition to extract topics, and Latent Dirichlet Allocation that relies on underlying mixture model to follow some specific distribution shape. However to the best of our knowledge none of this work has been successfully applied as a standalone technique for unsupervised classification of large unmoderated text collections.

Our goal here is to build a mathematical framework that not only captures the difference between different documents, but also describes connections between related topics. This model provides us with a framework to formally analyze the properties of topics *before* we actually discover them and in some cases even without having to discover them. This in turn enables us to provide accuracy guarantees in the world where text collection contains documents on unknown number of topics and we only ever learn only a small fraction of the topic. We further show that our guarantees are independent of the total number of topics present in the text.

Mixture Models 101 Mixture models form one of the most widely used classes of generative models for describing structured and clustered data [26].

Various applications of mixture models include problems in computer vision, text clustering, collaborative filtering, and bioinformatics. One of the common formulations of mixture models can be described as follows. The data (e.g., texts, images, user profiles, etc.) is a collection of independent samples (e.g., individual documents, image features, etc.), each created by a sequence of independent draws from some hidden distribution over a feature space (terms, pixels, etc.). For example, in the case of text, each document is modeled as a sequence of independent trials from some underlying term distribution. More precisely, there is a set of topics, where each topic is defined by a hidden distribution over the space of all possible terms. A given document can be related to several topics (e.g., discussing the role of religion in physical sciences), and is modeled as a sample from a linear mixture (hence the name) of corresponding topical distributions. The actual mixing coefficients are defined by the document's quantitative relevance to each of the topics. A similar interpretation is possible for collaborative filtering where each user's profile (e.g. the books he bought) is modeled as a sequence of independent trials from some hidden distributions over the item space. In the recent work [27] we have shown that there exists a combinatorial algorithm which in some sense allows to reconstruct the underlying mixture model.

Hierarchical mixture models While mixture models provide a clean and appealing mathematical framework, they are also known to be too simplistic in the cases where a lot of topics are present. For instance, it seems hopeless to even count all the topics present in the web, let alone find them. In this work we depart from the basic assumption that topics are independent, and instead that there is some inherent structure in the topic themselves. More specifically, we assume that topics form an hierarchy based on the level on their generality, and there is an underlying adversary driven generative process that builds this hierarchy. The generative process that we develop in [28] is a multi-step adversary driven random process and here we give a brief overview of it.

The construction proceeds as follows. First the adversary chooses the baseline topical distribution $T^{(0)}$, which will be at the root of our hierarchy tree; we don't restrict the adversary regarding how he chooses the base topic. Given the root of the tree, the adversary starts building the rest of the tree. On every topic he decides how many children it will have, and then chooses the distributions for all immediate children topics at once. Suppose topic T is a parent topic, and the adversary is choosing the parameters of a child topic T'. For each term i in a child topic, he chooses a probability distribution D'_i satisfying some specific feasibility conditions. Then, the random process samples a value $\mathcal{E}'_i$ for each D'_i and sets $T'_i \leftarrow T_i + \mathcal{E}'_i$. After the change is applied, the vector T' is normalized by a factor α' so that it represents a valid distribution. In general, for a topic $T^{(i)}$, the change vector that was applied to its parent to generate $T^{(i)}$ is denoted by $\mathcal{E}^{(i)}$ and the normalization constant by α_i.

At each step the adversary can expand any leaf in the constructed hierarchy. The change vector $\mathcal{E}'$ satisfies the following constraints:

(i) The resulting topic cannot contain any negative elements (as otherwise it would not be a distribution) thus for any term i, the change is bounded so that $\mathcal{E}'_i \geq -T_i$.

(ii) For any term we have $\mathrm{E}\left[\mathcal{E}'_i\right] = 0$. It is important to note that we *do not* require the change to be symmetric, and in fact it can be arbitrarily skewed toward increasing probabilities as long as the expectation stays zero.

(iii) The difference between child and parent topics should be large enough: $\mathrm{E}\left[||\mathcal{E}'||_1\right] \geq \Delta$ for some constant Δ, where $\mathcal{E}'$ – is a resulting vector change to the entire distribution.

(iv) Finally, changes cannot be concentrated on just a few items. Therefore, if each $\mathcal{E}'_i$ has range $[A_i, B_i]$, we define

$$\mathrm{slope}[\mathcal{E}'] = \frac{\mathrm{E}\left[||\mathcal{E}'||_1\right]}{||A - B||_2}$$

and require $\mathrm{slope}[\mathcal{E}'] \geq \frac{1}{\varepsilon_2}$, where ε_2 is some constant. Note that from condition (i) we have $A_i \geq -T_i$.

Given the hierarchy, each document is produced using a mixture of the topics from this hierarchy. The topics might include both leaf topics or internal topics.

Results and Analysis Overview Our main technical result can be formulated as follows. Suppose we know all topics on a path of the hierarchy starting from the root for instance $T_0, T_{i_1}, \ldots T_{i_k}$, with T_0 being the base topic. We show that topics in the unknown part of the hierarchy in some very precise sense are indistinguishable from the *closest* topic that belongs to the path. This in particular enables us to provide guarantees of the following form.

Lemma 2 (Document relevance to a path)**.** *Consider a path starting from the root $T^{(a_0)}$, …, $T^{(a_l)}$, and the corresponding weight matrix $W = [T^{(0)}, \ldots, T^{(l)}]$ and let V be a pseudoinverse of W. Let a document is sampled from a mixture*

$$D = p_1 T^{(b_1)} + \ldots + p_{l'} T^{(b_{l'})}$$

(not necessarily overlapping with the topics in the path) and let $\tilde{D}$ be the term count vector of this document, then the vector $\tilde{\mathbf{r}} = V\tilde{D}$ would approximate the coefficients of the projection of D onto the path.

Our empirical results include an algorithm that reconstructs the hierarchy from unlabeled data and we demonstrate its applicability by using it to cluster a collection of scientific archives.

Our algorithm is based on general (non-hierarchical) mixture model algorithm from [27]. We use concentration bounds to show that with high probability the topics produced by our generative process are independent and in some sense parent topics are indistinguishable from (unknown) child topics. Our main tools here is Hoeffding's inequality, and the framework introduced in [27] which allows to quantitatively measure independence of topics using L_1-norm.

Open problems and connections to other areas One interesting connection that we mention here is between the topic hierarchy and phylogenetic tree reconstruction problem. In the latter the input is vectors of attributes for all existing species. There, the assumption is that there is an evolutionary tree of all the species, where each node contains some aggregate description of a particular species (e.g. number of legs, presence of specific gene, etc), the root contains a description of a common ancestor of all currently existing species, and the transition from the parent is modeled via a mutation process. There has been a lot of work done on phylogenetic trees including several results that guarantee close to optimal reconstruction. We refer to [24, 29, 25] for more details. Our problem is different in several aspects. First, our attributes (terms) are defined on a continuous domain and share the same normalization, whereas in phylogenetic trees they could be drawn from different domains. Second, our samples (documents) do not contain full information about the node they belong to, but rather a small sample drawn from it (whereas a single animal sample would contain all or most of the properties for a given species.) Finally, document distributions could be a mixture of distributions from several topics (in the case of a phylogenetic tree this would correspond to observing a mix of a dog and a mouse). However it would be very interesting to connect these different lines of research.

2.4 Clustering large data sets

Gereon Frahling

The large data sets Google deals with are often retrieved from a set of sources of different quality and size. Google News fetches terabytes of articles from many different news sites all over the world. Google Maps uses different Yellow Page sources, websites, and entries written by business owners themselves as sources for business listings. Froogle fetches product descriptions from various internet sites. In all these cases massive sets of raw data is collected. Using clustering algorithms we try to reconstruct the underlying set of items from the different (in some cases erroneous) descriptions.

Hierarchical agglomerative clustering (HAC) algorithms are used widely in the information retrieval community to identify such groups of similar items. Using an arbitrary distance function between clusters of items, we start with a set of n clusters consisting of one item each. Then we iteratively join the two clusters having the smallest distance to each other until we reach a certain distance threshold D between all clusters.

Although HAC algorithms are known to output good clustering results certain heuristics have to be used to reduce their runtime. Since a naive implementation would need $\Omega(n^2)$ distance computations, practical implementations have to preselect a small set of pairs of clusters to only evaluate and compare their distances.

For the huge data sets mentioned above the data does not even fit on one machine. We therefore have to parallelize the cluster computation. This involves the partition of the input set into small digestible groups to be fed to the computers and strategies to merge the clusters at the borders of these groups. There are some general approaches to achieve this partitioning in practice (see *canopy clustering*), but it is not fully understood when such general approaches output a clustering comparable to the output of a sequential HAC algorithm.

In our research we try to find requirements on the data which suffice to prove properties of the parallel algorithm. We first stated formal descriptions of clusterable data:

A data set is called clusterable if

- *It can be covered with k balls of radius D;*
- *Increasing the radius of such a ball by a factor of $O(1)$ only increases the number of covered items by a factor of c.*

For this description (and other weaker descriptions) of clusterability we were able to show that clusterable data can be clustered by a parallel HAC algorithm. The output of the algorithm is equal to the output of a *relaxed* sequential HAC algorithm which always joins a pair of clusters having distance at most $d \cdot (1 + \epsilon)$, where d denotes the current smallest distance between a pair of clusters. The algorithm can run on Google's MapReduce system and needs $O(c \cdot \log n/\epsilon)$ rounds of the parallel system to cluster the whole data set.

Tests on Google News data shows that the clusterability requirements needed for the execution of such a parallel algorithm are fulfilled. It remains to be tested if such general approaches are able to compete with parallel clustering algorithms invented and highly optimized for a specific problem. This research is joint with Krzysztof Onak of MIT.

3 Game Theory

3.1 Game Theory of Sponsored Search Auctions

Eyal Even-Dar

The Internet economy has been revolutionized by the introduction of sponsored search links. Sponsored links are a small number of advertisements that the search engine displays in addition to the standard search results. These ads are arranged in *positions* top to bottom, typically on the side. Currently, the advertiser pays only when the user clicks on the link (known as *pay per click (PPC)*). It is a difficult task to set a fixed price for each position because the search queries vary widely and with them the value of the positions. Hence, typically, auctions are used to determine the prices and the position allocation, and these are called *position auctions.* A major task for the search engine is to determine the rules of the auction.

Today, both Google and Yahoo! use the *generalized second price auction* (GSP) to determine prices and allocation. The GSP auction ranks the ads by the product of the advertiser's bid with a quality score, which is often abstracted as the *click-through rate (ctr)*—the probability that the user will click on the advertisement. Then, the ad in position i is charged based on the bid of the ad on position $i + 1$ and their quality scores. The well known Vickrey-Clarke-Groves (VCG) [36, 37, 39] mechanism can also be applied to position auction. A key assumption in the analyses and the auction design is that of *ctr separability* [33], i.e. the ctr of advertiser i in position j is $\alpha_i\beta_j$, where α_i depends only on the advertiser and β_j depends only on the position. We are now ready to define envy free Nash equilibrium.

Definition 1. *Suppose there are K positions and N advertisers. Let $P = \{p_1, \ldots, p_n\}$ be a set of prices where $p_{k+1} = \cdots = p_n = 0$ and $loc : N \to K$ be the position allocation. Then advertiser i (with minimum reserve price R_i set by the search engine)* envies *the advertiser in position j if*

$$u_i(loc(i)) \;=\; (v_i - \max\{p_{loc(i)}, R_i\})\beta_{loc(i)} < (v_i - \max\{p_j, R_i\})\beta_j \;=\; u_i(j).$$

An Envy free Nash equilibrium is prices in which no advertiser envies any other advertiser.

The first studies of position auctions were conducted by Edelman et al. [35] and Varian [36] and their main result is as follows.

Theorem 2. *[35, 40] There exists a set of bids such that*

- *each advertiser is allocated to the same position as she would be in the dominant-strategy equilibrium of VCG,*
- *the winner of each position pays the same total price as she would have in the dominant-strategy equilibrium of VCG, and*
- *the advertisers are in an envy-free Nash equilibrium.*

The key idea in the proof relies on the "locally envy-free" notion and was used to prove the existence of equilibrium and to characterize GSP. Locally envy free property implies that if a player at position i does not envy the players at position $i \pm 1$. In particular, the "locally envy-free" property captures the dynamics of advertisers trying to move up or down the list of positions and plays a crucial role in understanding the equilibrium properties of GSP. The authors then show that in GSP local envy free prices implies envy free Nash equilibrium.

The initial studies have provided important insights while using an idealistic model. However, the implementation of the mechanism in practice contains additional features. As we shall see here the results are not robust, and small changes may result in a different advertiser behavior. This motivates the question that is the focus of this section: What are the strategic changes in the outcome of GSP–and more generally, other position auctions—in presence of additional features? The additional features that we will elaborate on are minimum specific advertiser prices [34] and prefix preferences of the different advertisers [32].

Minimum specific advertiser prices - minimum specific advertiser prices are currently used by Google as a method to encourage high quality advertisements. In our study we focus on analyzing GSP and implementing truthful auctions. A first step is to study VCG, and doing so immediately reveals that a naïve post-VCG enforcement of advertiser-specific minimum bid prices can break the truthfulness property, in sharp contrast to the simple implementation of VCG in previous results. Being more careful, we show suitably modified allocation and pricing that is a truthful variant of VCG; We then turn our attention to GSP. Since GSP is not truthful to begin with, we study the effect of advertiser-specific minimum prices on the equilibria. Furthermore, we see that an important property enjoyed by basic GSP, ie., locally envy-free property, no longer holds. Our main results is that despite losing envy locality, GSP with advertiser-specific minimum prices still has an envy-free equilibrium.

Theorem 3. *[34] The GSP mechanism with advertiser-specific minimum prices has set of bids that induce envy-free Nash equilibrium.*

To derive the prices of this equilibrium, we define a specialized Tâtonnement process that takes a global view of the best-response relationship between advertisers and positions. This global view was unnecessary in the basic GSP analysis such as in [35] because of envy locality. We prove that the process converges to a set of prices from which an envy-free equilibrium set of bids is derived.

Bidding on prefixes - In this scenario each advertiser can bid both the value she is willing to pay and the position cutoff, i.e the lowest position she is interested. Once again one can show that neither naive implementation of VCG nor of GSP would yield a reasonable mechanism. We then show how to make adequate changes to the mechanism that generalize the desirable properties of current auctions that do not have position constraints. We now state the main result.

Theorem 4. *[32] In the top-down prefix auction, there exists a set of bids and stated position cutoffs such that*

- *each advertiser is allocated to the same position as she would be in the dominant-strategy equilibrium of VCG,*
- *the winner of each position pays the same total price as she would have in the dominant-strategy equilibrium of VCG, and*
- *the advertisers are in an envy-free Nash equilibrium.*

While in this section we only considered two specific features on top of the idealized auction model, as the number of advertisers and specialized advertisement companies grow, the auction is likely to become more and more sophisticated. To make the theoretical analysis of the auction sound and have practical implications, future research should look at additional features and aspects of the game. Aspects of the game which are of a great interest include the effect of advertisement quality on future user behavior, customer satisfaction, and connection between different auctions. Additional features which seems to be attractive from the advertiser point of view are traditional ones, such as frequency of showing an ad, different prices for different positions, better targeting and many more. The influence of all of the above on the auction might be enormous and understanding the influence will be of significant value, both from theoretical and practical point of view.

3.2 Sponsored Search Auctions with Markovian Users

Gagan Aggarwal

Introduction. Sponsored search involves running an auction among advertisers who wish to have their ad shown next to search results for specific keywords. To figure out the best assignment of advertisers to slots, we need to know the value of each possible ad assignment, for which we need to understand the behavior of a search engine user when she sees the displayed ads. Previous work on sponsored search assumes that the probability of a user clicking on an ad is independent of the other ads shown on the page. In fact, most prior work, implicitly or explicitly, makes the *separability* assumption [33], i.e. the user behavior can be captured by two types of parameters: ad-specific click-through rates p_i and position-specific visibility factors α_j, such that the probability of an ad i in position j receiving a click is equal to $p_i \alpha_j$. An intuitive explanation is that a user looks at the ad in position j with probability α_j independent of which ads are shown on the page, and having looked at an ad i, clicks on it with probability p_i. Under this model, ranking by $v_i p_i$ maximizes efficiency, where v_i is advertiser i's value per click.

While the assumption that the probability of an ad getting clicked is independent of *other ads* on the page does lead to a simpler model, it does not always hold in practice [38]. We propose a simple Markovian user model which tries to capture the effect of seeing other ads on the subsequent click-behavior of a user. The most efficient assignment under the new model turns out to be different than ranking by $v_i p_i$. We show that the efficient assignment has the desirable "monotonicity property" and also present an algorithm to find it. A truthful auction then follows from an application of the Vickrey-Clarke-Groves (VCG) mechanism.

Related Work. Sponsored search has been an active area of research in the last several years after the early papers explored the foundational models [33, 35, 40]. In general, the motivation for the work that followed is that sponsored search in practice is much more complex than as described by the first models. Some papers have taken on the effect of advertiser budgets as well as analyzing bidder strategy and dynamics. There have also been several papers offering extensions to GSP, or entirely new models and mechanisms.

The model. Our model is based on a user who starts to scan the list of ads from the top, and makes decisions (about whether to click, continue scanning, or give up altogether) based on what he sees. More specifically, we model the user as the following Markov process: "Begin scanning the ads from the top down. When position j is reached, click on the ad i with probability p_i. Continue scanning with probability q_i." In this model, if we try to express the click probability of an ad i in position j is equal to $p_i \Pi_{i' \in A} q_{i'}$, where A is the set of ads placed above ad i.

The auction mechanism. As we show in the paper, one can use dynamic programming to find the optimal assignment of ads to positions. This allocation combined with VCG [36, 37, 39] pricing yields a truthful mechanism.

Monotonicity property. For the above mechanism to be of practical use, the allocation function needs to be (weakly) monotone in individual bids, i.e. as an advertiser raises her bid, she cannot move down and her click probability can only increase. Firstly, it makes sense given practical considerations – often higher positions have benefits other than clicks which the auction does not directly take into account. Moreover, it is an expectation that advertisers have formed from interacting with the auction currently being used – the *Generalized Second Price* (GSP) auction, where each bidder i submits a bid b_i stating the maximum amount they are willing to pay for a click, and the bidders are placed in descending order of $b_i p_i$, where p_i is the *click-through-rate* of advertiser i, and each bidder pays the minimum bid she needs to make in order to be keep her position. We dig deeper into the structure of the optimal assignment under the new model and show that the optimal assignment is indeed weakly monotone. In the paper, we also show how to use this property to get a faster algorithm to compute the efficient allocation.

To show the monotonicity property, we first introduce the notion of "adjusted bid per impression (a-bpi)" $a_i = b_i * p_i/(1 - q_i)$. Intuitively, this quantity is the impression value adjusted by the negative effect this bid has on the ads below it. Using a simple exchange argument, we show that:

Lemma 3. *In the most efficient assignment, the ads that are placed are sorted in decreasing order of adjusted bpi $a_i = b_i * p_i/(1 - q_i)$.*

While this lemma tells us the selected subset of ads are sorted in the optimal assignment, it does not tell us *which* subset of ads to chosen. It is not hard to construct examples where the top k by "bpi" or "a-bpi" do not work. Next we show a "bidder dominance" lemma, using which we show a subset relationship between the optimal assigned sets of ads for different number of positions.

Lemma 4. *For all bidders i in an optimal assignment, if some bidder j is not in the assignment, and $a_j \geq a_i$ and $e_j \geq e_i$, then we may substitute j for i, and the assignment is no worse.*

Lemma 5. *Let $j \in \{1, \ldots, k\}$ be some number of positions, and let C be an arbitrary set of bidders. Let $OPT(C, j)$ denote the set of all optimal solutions for filling j positions with bidders from the set C. Then, for all $S \in OPT(C, j-1)$, there is some $S' \in OPT(C, j)$ where $S' \supset S$.*

To show the above, we use an interesting exchange argument. In particular, we show that if the above is not true, then there exists an ad in the optimal set for $j - 1$ positions that can be used to improve the solution for j positions.

Next we consider the optimal assignments before and after an advertiser changes her bid (keeping all other bids fixed) and some intermediate assignments obtainable by doing simple exchanges. Using the subset structure of optimal assigned sets and some intricate exchange arguments between various assignments, we show the monotonicity property.

Theorem 5. *With all other bids fixed, the probability of receiving a click in the optimal solution is non-decreasing in one's bid. In addition, the position of a particular bidder in the optimal solution is monotonic in that bidder's bid.*

Future Work. This section described the work done jointly with Jon Feldman, S. Muthukrishnan and Martin Pál. In the future, it will be interesting to consider even more powerful models, e.g. models that allow the continuation probability q_i to be a function of position as well and models that endogenize the actions of the user as she navigates on the landing page.

3.3 Auctions with Budget Limits

Noam Nisan

This section describes joint work of Noam Nisan (Google Tel-Aviv), Ron Lavi (Technion), and Shahar Dobzinski (Hebrew U.) [1].

The starting point of almost all of auction theory is the set of players' *valuations*: how much value (measured in some currency unit) does each of them assign to each possible outcome of the auction[1]. When attempting actual implementations of auctions, a mis-match between theory and practice emerges immediately: *budgets.* Players often have a maximum upper bound on their possible payment to the auction – their budget. This budget limit is not adequately expressible in existing auction theory, and in fact, the nature of what this budget limit means for the bidders themselves is somewhat of a mystery[2]. There seems to be some risk control element to it, some purely administrative element to it, some bounded-rationality element to it, and more. What can not be disputed is that these budget constraints are very real to the bidders, in fact, usually more concrete and real than the rather abstract notion of the valuation.

As budgets are main elements in most of economic theory (e.g. in the Arrow-Debreu model), it is quite surprising that so little attention has been paid to them in auction theory, and in particular in auctions of multiple units or goods. The reason seems to be that budgets take us out of the clean "quasi-linear" setting, which changes many of the "rules of the game", e.g. VCG mechanisms are no longer incentive compatible. Some previous papers have attempted modeling the budget limit as an upper bound on the value obtained by the bidder rather than on his payment. This model

[1]This "quasi-linear" setting is needed due to the Gibbard-Satterthwaite Impossibility Theorem for the more general setting.

[2]It does not seem to simply reflect the true liquidity constraints of the bidding firm.

maintains the quasi-linear setting but misses the point when the budget limit is a real constraint, i.e. is lower than what the payment would be without it. In such cases, the marginal utility of the bidder from acquiring additional goods would be incorrectly modeled as being zero. This would lead to several completely unreasonable artifacts, e.g. that a social-welfare maximizing allocation may not even be Pareto-optimal and that VCG prices would be zero. We are aware of only two recent papers that have attempted designing auctions with multiple items while taking budget limits into account: [41, 42][3].

The following example demonstrates what may happen in the presence of budgets. Let us consider the following very simple case: two bidders competing for a large number of identical items, which we model as a single infinitely divisible good. Further more, we consider the special case where the budget limits of these two players are publicly known and equal:

The Setting: Player i ($i = 1, 2$) has a private value of v_i (for the whole good) and a publicly known budget of $b_i = 1$. We are auctioning a single infinitely divisible good. The outcome of the auction is an allocation where each player i gets $x_i \geq 0$ fraction of the good (with $x_1 + x_2 \leq 1$), and pays a total of $p_i <= b_i$ for it (i.e. at most his budget limit). The utility of player i in this case is $u_i = x_i v_i - p_i$.

An allocation $\{(x_i, p_i)\}$ is *Pareto-optimal* if for no other allocation $\{(x'_i, p'_i)\}$ are all players better off, $x'_i v_i - p'_i \geq x_i v_i - p_i$, including the auctioneer $\sum_i p'_i \geq \sum_i p_i$, with at least one of the inequalities strict. An auction is *Incentive-compatible* if no player ever has the incentive to report a false value, i.e. for every value of v_i, v_j ($i \neq j$) and for every v'_i, we have that $x'_i v_i - p'_i \leq x_i v_i - p_i$, where $\{(x_i, p_i)\}$ is the allocation obtained for v_i, v_j and $\{(x'_i, p'_i)\}$ is the allocation obtained for v'_i, v_j.

A Mechanism: The mechanism makes the following allocations and payments for $v_i < v_j$:

- If the low player has $v_i \leq 1$ then the high player gets everything at the second price: $x_j = 1, p_j = v_i$ (and $x_i = 0, p_i = 0$).
- Otherwise, the low player gets $x_i = 1/2 - 1/(2v_i^2), p_i = 1 - 1/v_i$ and the high player gets $x_j = 1/2 + 1/(2v_i^2), p_j = 1$. In particular the low player's value determines the allocation between the two players as well as his own payment, while the high player exhausts his budget.

(If $v_i = v_j = v$ then they split the good $x_i = x_j = 1/2$, and pay for it $p_i = p_j = v$ if $v \leq 1$, and $p_i = p_j = 1/2 - 1/(2v)$ if $v > 1$.)

One may verify that this mechanism is Incentive-compatible and Pareto-optimal. Surprisingly we show that this is the only such mechanism, at least among those that are anonymous, i.e. treat the two players identically[4].

Theorem 6. *The mechanism above is the only anonymous mechanism for the setting above that is Incentive-compatible and Pareto-optimal.*

From this we can also deduce that the assumption that the budgets were publicly known was essential:

Theorem 7. *There is no annonymous Incentive-compatible Pareto-optimal mechanism for the setting above if the budgets b_i as well as the valuations v_i are private information.*

3.4 Online Ad Slotting with Cancellations

Florin Constantin

Current sponsored search systems only offer advertisers the opportunity to compete in a *spot auction* for ad slots. Many advertisers are however interested in reserving ad slots ahead of time.

[3]Several other works have studied algorithmic issues that ignore incentives or studied "standrad" auction with budget-limited bidders.

[4]Assuming that a player that wins nothing pays nothing, i.e., $x_i = 0 \rightarrow p_i = 0$, without which the budget constraints can be easily side-stepped.

Since campaigns can use numerous keywords, target web sites and bids may change often, a simple automated advance-booking system that has good performance is desired.

We aim for a system that takes sequential decisions and outputs an assignment of value always within a constant factor of the optimum achievable offline (if all bids were known ahead of time). Achieving that is not possible without any assumption on bidders' values and/or arrivals. We allow costly cancellations - the seller can *bump* a bidder (cancel her reservation), resulting in a utility loss for the bidder. We propose a simple bidder model and an allocation system with good revenue, efficiency and game-theoretic properties.

In our model, bidders arrive sequentially, with each bidder specifying a subset of slots she is interested in, and a bid (which may be different than her true value) for such a slot. Each bidder demands only one slot and requires an immediate answer from the seller. If the seller grants a reservation to a bidder but cancels it later, the bidder incurs a loss equal to an α fraction of its value.

If bidder i has value $v(i)$ for a slot, we model her utility as $\lambda \cdot v(i) - x(i)$ where $x(i)$ is i's transfer to the seller and λ equals 0 if i is rejected, 1 if i is accepted and granted her item and $-\alpha$ if i is accepted but bumped.

The allocation system in Algorithm 1 has the power of revising its decisions – it can re-assign a bidder to a different slot or bump an already accepted bid. It builds upon an approximation algorithm for bipartite matching by McGregor [43].

Note that if all items wanted by a new bidder i are reserved for other bidders, i can always be swapped into the current set of accepted bidders R by swapping a suitable[5] bidder j out of R. However, Algorithm 1 only does so if i improves by a γ factor over i^*, the lowest-bidding suitable j. If i is accepted, bumping i^*, then i^* is awarded a bump payment amounting to an α fraction of its bid.

A bidder in the final assignment F pays the seller the lowest bid for which she would have still been in F. However, if for a lower bid she would have been bumped, she receives a discount equal to her highest possible bump payment. This ensures that winners that bid their true value do not prefer being bumped.

Algorithm 1 Allocation algorithm and payments. A new bidder is accepted if she improves over her lowest-bidding indirect competitor by at least a γ factor. Bumped bidders are given a bump payment to make up for their utility loss.

```
for each arriving bidder $i$ bidding $b(i)$ do
  Let $R$ be the set of bidders currently holding a reservation.
  if $R \cup \{i\}$ is satisfiable then grant $i$ a reservation.
  else let $i^*$ be the lowest-bidding $j \in R$ such that $R \cup \{i\} \setminus \{j\}$ is satisfiable.
      if $b(i) < (1+\gamma)b(i^*)$ then reject $i$.
      else grant $i$ a reservation
          cancel $i^*$'s reservation and pay her $\alpha b(i^*)$
end for
Fix a bidder $i$ in the final assignment $F$ and all other bids. $i$ pays the lowest bid $\underline{b}(i)$ for which
she remains in $F$. If for some bid $b'(i) < \underline{b}(i)$, $i$ would have been granted a reservation that is
canceled later, give $i$ an $\alpha\underline{b}(i)$ *discount*.
```

Algorithm 1 has many desirable properties as we prove:

- A bidder should always bid at least her true value; winners have incentives to bid truthfully. No bidder has negative utility if bidding her true value.
- Its revenue is a constant fraction of the revenue of the Vickrey-Clarke-Groves system (a standard offline truthful allocation system outputting an optimal assignment) on the set of bids.

[5]We call a set S of bidders *satisfiable* if all bidders in S can be assigned to a slot they want. Thus j is *suitable* for i given R if $R \cup \{i\} \setminus \{j\}$ is satisfiable.

- The final assignment F is a constant approximation to the optimum offline assignment given the set of bids. The same statement holds for the set of *true values* under rational behavior.
- For $\alpha > 0.618$ and a particular $\gamma(\alpha)$, no algorithm has a higher worst-case *effective* total value (winners' values - α times bumped bidders' values).

All the results extend to the more general setting of *matroids*, where combinatorial constraints between the items for sale are allowed: e.g. when the items for sale are edges in a network and bidders bid to form a spanning tree.

This subsection represents work done jointly with Jon Feldman, S. Muthukrishnan and Martin Pál of Google during summer 2007. There are numerous extensions for our current model that look promising. Our mechanism has good worst-case performance, but better revenue and/or assignments may be obtained under perhaps additional assumptions about value distributions and bid arrivals. Bidders may have the option to pay the seller more for higher γ (making it harder for future bidders to bump them) or higher α (being refunded more when bumped). Finally, bidders may demand bundles of slots which will require a combinatorial extension of our system.

4 Acknowledgements

We sincerely thank our engineering and research colleagues at Google.

References

[1] S. Dobzinski, R. Lavi, and N. Nisan. Multi-unit Auctions with Budget Limits. *Working paper*, 2008.

[2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *STOC*, pages 20–29, 1996.

[3] A. Bialecki, M. Cafarella, D. Cutting, and O. O'Malley. Hadoop: a framework for running applications on large clusters built of commodity hardware, 2005. Wiki at `http://lucene.apache.org/hadoop/`.

[4] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004.

[5] J. Feldman, S. Muthukrishnan, A. Sidiropoulos, C. Stein, and Z. Svitkina. On distributing symmetric streaming computations. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 710–719, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[6] M. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *Technical Note 1998-011, Digital Systems Research Center, Palo Alto, CA*, 1998.

[7] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 2005.

[8] N. Ailon. Aggregation of partial rankings, p-ratings and top-m lists. In *SODA*, 2007.

[9] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: Ranking and clustering. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, 684–693, 2005.

[10] N. Ailon, M. Mohri. An efficient reduction of ranking to classification. To appear in *COLT*, 2008

[11] N. Alon. Ranking tournaments. *SIAM J. Discrete Math.*, 20(1):137–142, 2006.

[12] D. Ariely., G. Loewenstein, and D. Prelec. "Coherent arbitrariness": Stable demand curves without stable preferences *The Quarterly Journal of Economics*, 118(1):73–105, 2008

[13] K. J. Arrow. A difficulty on the concept of social welfare. *Journal of Political Economy*, 58(4):328–346, 1950

[14] M. F Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, G. B. Sorkin. Robust reductions from ranking to classification. In *Annual Conference on Learning theory (COLT)*, volume 4539 of *Lecture Notes in Computer Science*, 604–619. Springer, 2007.

[15] J. C. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences*, 1781.

[16] W. Cohen, R E. Schapire, Y. Singer. Learning to order things. *J. Artif. Intell. Res. (JAIR)*, 10:243–270, 1999.

[17] M.-J. Condorcet. Éssai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. 1785.

[18] C. Cortes, M. Mohri, and A. Rastogi. Magnitude-preserving ranking algorithms. In *24th International Conference on Machine Learning (ICML 2007)*, 169–176, 2007.

[19] C. Cortes, M. Mohri, C. Rudin, and R E. Schapire. Margin-based ranking meets boosting in the middle. In *18th Annual Conference on Learning Theory (COLT)* 27–30, 2005.

[20] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[21] C.A.R. Hoare. Quicksort: Algorithm 64. *Comm. ACM*, 4(7):321–322, 1961.

[22] E. L. Lehmann. *Nonparametrics: Statistical Methods Based on Ranks.* Holden-Day, San Francisco, California, 1975.

[23] D. P. Williamson and A. van Zuylen. Deterministic algorithms for rank aggregation and other ranking and clustering problems. In *Proceedings of the 5th Workshop on Approximation and Online Algorithms (WAOA)* 260–273, 2007.

[24] A. Ambainis, R. Desper, M. Farach-Colton, and S. Kannan. Nearly tight bounds on the learnability of evolution. In *FOCS*, 1997.

[25] M. Farach and S. Kannan. Efficient algorithms for inverting evolution. In *STOC*, 1999.

[26] G. McLachlan and K. Basford. *Mixture Models, inference and applications to clustering.* Marcel Dekker, 1987.

[27] J. Kleinberg and M. Sandler. Using mixture models for collaborative filtering. In *STOC*, 2004.

[28] M. Sandler. Hierarchical mixture models: a probabilistic analysis. In *KDD*, 2008.

[29] M. Sharikar and N. Ailon. fitting tree metrics: hierarchical clustering and phylogeny. In *FOCS*, 2005.

[30] I. Giotis and V. Guruswami. Correlation clustering with a fixed number of clusters. In *SODA*, 2006.

[31] F. McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42th Annual IEEE Symposium on Foundations of Computer Science*, 2001.

[32] G. Aggarwal, J. Feldman, and S. Muthukrishnan. Bidding to the top: VCG and equilibria of position-based auctions. In Proc. Workshop on Approximation and Online Algorithms (WAOA), 2006.

[33] G. Aggarwal, A. Goel, and R. Motwani. Truthful auctions for pricing search keywords. In *ACM Conference on Electronic Commerce (EC)*, 2006.

[34] E. Even-Dar, J. Feldman, Y. Mansour and S. Muthukrishnan. Sponsored search with bidder-specific minimum prices. *Submitted*, 2008.

[35] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. In Second workshop on sponsored search auctions, 2006.

[36] W. Vickrey. Counterspeculation, auctions and competitive-sealed tenders. *Finance*, 16(1):8–37, 1961.

[37] E. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[38] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 87–94, New York, NY, USA, 2008. ACM.

[39] T. Groves. Incentives in teams. Econometrica, 41(4):617–631, 1973.

[40] H. Varian. Position auctions. *International Journal of Industrial Organization*,25(6): 1163–1178, 2007.

[41] J. Feldman, S. Muthukrishnan, E. Nikolova, M. Pl. A Truthful Mechanism for Offline Ad Slot Scheduling. *Proc. SAGT*, 2008.

[42] C. Borgs, J. Chayes, N. Immorlica, M. Mahdian, and A. Saberi. Multi-unit auctions with budget-constrained bidders. *Proc. EC*, 2005.

[43] A. McGregor. "Finding Graph Matchings in Data Streams". *In APPROX-RANDOM*, 170-181, 2005.