Associative Search in Peer to Peer Networks: Harnessing Latent Semantics*

Edith Cohen† Amos Fiat‡ Haim Kaplan§

July 20, 2006

Abstract

The success of a P2P file-sharing network highly depends on the scalability and versatility of its search mechanism. Two particularly desirable search features are scope (ability to find infrequent items) and support for partial-match queries (queries that contain typos or include a subset of keywords). While centralized-index architectures (such as Napster) can support both these features, existing decentralized architectures seem to support at most one: prevailing unstructured P2P protocols (such as Gnutella and FastTrack) deploy a "blind" search mechanism where the set of peers probed is unrelated to the query; thus they support partial-match queries but have limited scope. On the other extreme, the recently-proposed distributed hash tables (DHTs) such as CAN and CHORD, couple index location with the item's hash value, and thus have good scope but can not effectively support partial-match queries. Another hurdle to DHTs deployment is their tight control of the overlay structure and the information (part of the index) each peer maintains, which makes them more sensitive to failures and frequent joins and disconnects.

We develop a new class of decentralized P2P architectures. Our design is based on unstructured architectures such as Gnutella and FastTrack, and retains many of their appealing properties including support for partial match queries, and relative resilience to peer failures. Yet, we obtain orders of magnitude improvement in the efficiency of locating rare items. Our approach exploits associations inherent in human selections to steer the search process to peers that are more likely to have an answer to the query. We demonstrate the potential of associative search using models, analysis, and simulations.

Keywords: Networks, search, information retrieval, peer to peer, latent semantic, rare items, algorithms

Computing Classification System (CCS): H.3.3 [INFORMATION STORAGE AND RETRIEVAL]:Information search and retrieval, F.2 ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY, C.2.4 [COMPUTER-COMMUNICATION NETWORKS]:Distributed systems, C.2.2 [COMPUTER-COMMUNICATION NETWORKS]:Network protocols

*Preliminary version of this paper appeared in the proceedings of IEEE Infocom 2003 [1].

†AT&T research labs, 180 park ave. Florham park, NJ, USA. Email: edith@research.att.com.

‡School of computer science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: {fiat}@cs.tau.ac.il.

§School of computer science, Tel Aviv University, Tel Aviv 69978, Israel. E-mail: {haimk}@cs.tau.ac.il. Research partially supported by ISF (Israel Science Foundation) grant 548/00.

1 Introduction

Peer-to-peer (P2P) networks have become, in a short period of time, one of the fastest growing and most popular Internet applications. As for any heavily used large distributed source of data, the effectiveness of a P2P network is largely a function of the versatility and scalability of its search mechanism.

Peer-to-peer networks came to fame with the advent of Napster [2], a centralized architecture, where the shared items of all peers are indexed in a single location. Queries were sent to the Napster Web site and results were returned after locally searching the central index; subsequent downloads were performed directly from peers. The legal issues which led to Napster's demise exposed all centralized architectures to a similar fate. Internet users and the research community subsequently turned to decentralized P2P architectures, where the search index and query processing, as well as the downloads, are distributed among peers.

Existing decentralized architectures can be coarsely partitioned into two groups [3]: unstructured, where search is *blind* (independent of the query or its context) and structured, where search is *routed*. (For more on the taxonomy of peer to peer networks see [4]. For a comprehensive survey of current architectures see [5]) The P2P architectures that spawned in Napster's void, eventually surpassing it in popularity, are *unstructured*. One of these architectures is Gnutella [6] under which items are only indexed by the peer that cache them; search can be resolved only by probing these peers; and peers are probed using flooding (that typically cover about 1000 nodes).

Flooding is one example for a blind search method used in unstructured networks and several others are evaluated in [3].[1] What is common to all of them is that on average, the success likelihood of a search probe can not be better than that of probing random peers [3]; thus for each item, the average success likelihood on a blind search probe is equal to the fraction of peers that index it. The recent wave of FastTrack [7]-based P2P architectures (Morpheus, Kazaa [8, 9]) incorporate improved design that allows for more efficient downloads (simultaneous from several peers and ability to resume after failure); and improved search (by designating some peers as search-hubs *supernodes* that cache the index of others).

A feature that undoubtedly contributes to the beaming success of these decentralized unstructured architectures is support for *versatile (partial-match) queries*: Shared items typically have meta-attributes describing their type and properties (e.g., title, composer, performer); the search supports partial-match queries that populate a subset of these fields and may contain typos. Another important feature of these architectures is their "loose" structure, with each particular peer being relatively dispensable; what makes the network overlay more resilient to failures and frequent joins and disconnects. On the flip side, unstructured architectures lack an important feature which Napster had offered: While popular items (current hit movies) can be located and downloaded fairly efficiently, P2P users seemed to have lost the ability to locate less-popular items (60's hits).

A different class of architectures proposed by the research community is decentralized structured P2P architectures (such as [10, 11, 12, 13, 14, 15]), commonly referred to as Distributed Hash Tables (DHTs). With DHTs, peers are required to store or index certain data items, not necessarily those items that these peers have contributed or interested in. Additionally, some hashing algorithm is used to identify the peers storing a given data item. The connections between different peers are also a function of the architecture. Thus, while DHTs can be very effective for applications where queries involve unique item identifiers (e.g., P2P Web caching), they require that peers store

[1] Random walk or multi-head random walk is another example of blind search

data for the “common good”; they incur much larger overhead than “unstructured” architectures when peers fail or leave the network; and inherently, they can not efficiently support partial-match queries.

We propose a new class of decentralized P2P architectures which we call *associative overlays*. Over these overlays we can *guide the search* towards more relevant peers. We argue that associative overlays retain the desirable properties of existing unstructured architectures, including being fully decentralized with “loose” structure, and supporting partial-match queries, while on the other hand, boost the efficiency of locating infrequent items.

Each of the overlays is defined over a *guide rule* which is a set of peers that satisfy some predicate. The idea is to pick the guide rule such that the contents of its peers are semantically similar. For example, we can have a guide rule of all peers that are interested in the the philosophy of science, or of all peers containing song titles by the artist formerly known as Prince. Another guide rule can contain all peers of a particular age group, or all peers speaking a particular language etc. Deciding which guide rules to use is a network design issue, in the sense that it should be specified by some global entity. Nevertheless our design allows the global entity to change guide-rules without significant overhead. In this work we focus on automatically-extracted guide rules of a particular form, which we call *possession rules*. Each possession rule has a corresponding data item, and its predicate is the presence of the item in the local index, thus, a peer can participates in a rule only if it shares the corresponding item.

The set of peers in a guide rule are connected to form an associative overlay. We guide the search by propagating the query only over associative overlays specified by the originating peer. By tailoring the search to the originating peer, we exploit presumed correlations between the answer to the current query and previous requests that originated at the same peer. Our basic premise is that peers that would have been able to satisfy previous queries by the originating peer are more likely candidates to answer a current query.

We study two search strategies over associative overlays. Our first search algorithm, Rapier (Random Possession Rule), picks at random possession-rule A to which peer i belongs and performs a blind-search (either by flooding or by some kind of a random walk) of some fixed depth within possession-rule A. If this blind-search fails Rapier repeats the process and picks another possession-rule to which i belongs uniformly at random and so on. We also propose a theoretically-grounded Greedy strategy. Each peer constructs its own Greedy strategy by preferring to invoke rules that would have been more effective on its past selections. The Greedy strategy approximates the strategy that would have performed best on all previous queries. Greedy incurs higher overhead than Rapier for building the search strategies, but our simulations indicate that Greedy strategies can substantially improve over Rapier search. We evaluate our design and algorithms both theoretically and experimentally. We demonstrate their advantage of making the search more effective and argue for the relative low overhead they incur.

The structure of the rest of this paper is as follows. In Section 2 we survey related work. Section 3 presents models metrics and methodology and discusses performance of blind search. Section 4 develops the associative search framework, presents Rapier, and compares it to blind-search using the Itemset model. Section 5 presents the Greedy strategy. Section 6 contains simulation results of the performance of different search strategies. Section 7 relates our approach to algebraic techniques used to optimize search in centralized architectures. We conclude in Section 8.

2 Related work

The effectiveness of blind search can be boosted by aggregation, that is, by peers summarizing the content available from other peers and routing according to this information. Prominent architectures in this category are the super-peer architectures (e.g. [7]). Other proposed architectures use meta-data and query keywords for the routing: With routing-indices [16] each node stores a summary of meta-data keywords of files on peers in its neighborhood along with the summaries of its neighbors, and this information is used to route queries to the neighbor(s) whose summary are the closest matches to the query keywords. With Neurogrid [17], peers remember all keywords of queries and responses passed through them and use this information for routing other queries. Cooper and Garcia-Molina [18] recently studied in depth what properties of the topology make index replication effective. These architectures fundamentally differ from our approach in that the information nodes accumulate and remember is not related to their local content, queries are routed solely according to the query keywords, and there is no use of the presence of associations between items or between keywords. Caching of responses and proactive replication are another mechanism that can boost blind search (see, e.g.[14]); particularly when the number of replicas of each item is carefully balanced against its query rate [19, 3]. The main drawbacks of aggregation, caching, and proactive replication is increased sensitivity of the system to malicious or selfish peer behavior, to spamming, and to very dynamic peers. Associative overlays offer an orthogonal way to boost search that could be combined with, but does not require, aggregation or replication.

Associative overlays tackle a networking problem using an approach that draws from extensive existing literature in the field of data-mining. The unique challenges of the P2P setting, however, make for fundamentally different issues than in traditional data-mining applications. A related classic data-mining problem is the *Market-basket problem*, which assumes a large number of items and consumers that fill their baskets with some subset of the items. This framework applies to many domains of human activity including supermarket shopping (customers vs items matrix), library checkouts (readers vs books), document classification (word/terms vs documents matrix), Web page hyperlinks (The basket of a Web page x contains all the Web pages x has links to), Web browsing (Users vs Web pages), and also to P2P networks (peers vs items). Common to all these datasets is that item placements in baskets are not independent. Thus, the data has certain "latent" structure [20, 21, 22, 23]. Application of Latent Semantic Analysis on such matrices (Pioneered by [20] on Document$\times$ Word matrices) typically reveals a "true" dimension that is significantly smaller from the actual one, which suggests high correlations in the data. Another mechanism to capture structure in such datasets is extracting *Association rules* [24, 25]. An example of an association rule is pairs of items that are often purchased together such as "Champaign and Caviar" or "Beer and Diapers." Such rules had been used for marketing (e.g., placing Beer and Diapers next to each other in the supermarket) and recommendation systems (e.g., recommend books to customers based on previous book purchases) [26, 27, 28, 29]. A computationally challenging important subproblem is to discover association rules that have *high correlation* but *low support* (e.g., the association rule "Champaign and Caviar" that are rare purchases but are often purchased together) [30].

Similarly to these data-mining techniques, we exploit the presence of associations; but the basic differences are imposed by our highly distributed setting. Our solution does not (and can not) explicitly obtain association rules but does heavily utilize their presence. Instead of clustering peers into communities we restrict the search to be more focused on relevant communities without explicitly identifying them. (See Section 7.)

Sripanidkulchai, Maggs, and Zhang [31, 32], also recently proposed to exploit "interest locality" to optimize P2P search. The basic observation driven their work is similar to ours. That is a peer that satisfy your previous request is more likely to satisfy your forthcoming requests as well. To exploit it they suggest that each peer would keep a list of peers that answered its previous query. To answer a new query the peer first tries the peers in its shortcut list and if this fails it defaults to do the regular flooding. This method although somewhat conceptually simpler than ours is limited as it does not provide a way to "focus" query propagation beyond the first hop. Sripanidkulchai, Maggs, and Zhang [31, 32] demonstrated the effectivity of their approach using simulations with proxy logs similar to ours and also using a trace of Kazza and and Gnutella downloads.

Several indexing architectures have been suggested to make search more efficient, both in the context of unstructured and structured networks. These designs require substantial changes in the protocols of the associated networks. Ahmed and Boutaba [33] suggest an hierarchical indexing scheme for inexact pattern matching to maintain keywords and meta-data. Their scheme is based on bloom filter and aggregation of patterns. Zhang and Hu [34] suggested a design in which peers pick up their neighbors using an indexing overlay (over self interests published by the peers) that is implemented using a DHT. PeerSearch [35], and MINERVA [36] store summaries, obtained using information retrieval techniques, of peer's content using a DHT. Each peer that is responsible for a topic holds a list of peers that contains the topic's content. A query is answered by first using the DHT to get to peers that store the topics of the query, and then searching those peer by some flooding mechanism

Following the first publication of our work in [1] subsequent work with similar design concepts have been published. One is the SETS architecture of Bawa, Manku, and Raghavan [37], and another is the GES architecture suggested by Zhu, Yang, and Hu [38]. Both architectures cluster nodes into semantic segments. The search procedure routes the query in "relevant" semantic segments. These architectures differ from each other by the details of how semantic segments are maintained, and the routing protocol. They both require more extensive changes in current unstructured protocols than our design.

3 Model and Evaluation Methodology

We represent the data present in the network by the *peer-item* matrix $D \in \{0,1\}^{n\times m}$ where n is the number of peers, m is the number of items, and $D_{ij} = 1$ if and only if peer i contains data item j. We define the *support set* of the jth item $S_j \subseteq \{1,\dots,n\}$, $1 \le j \le m$, to be

$$S_j = \{\ell \mid D_{\ell j} = 1\}.$$

I.e., S_j is the set of all row indices (peers) that contain data item j. The *joint support set* of two items j, k,

$$S_{jk} = \{\ell \mid D_{\ell k} = 1 \text{ and } D_{\ell j} = 1\} ,$$

is the set of peers that contain both items. We refer to $X_i = \{j \mid D_{ij} = 1\}$ (the set of items associated with peer i) as the *index of peer* i. We use the notation $s_j = |S_j|$, $s_{jk} = |S_{jk}|$, and $x_i = |X_i|$.

We define $|D| = \sum_{i=1}^{n} x_i$ to be the combined size of all indexes. The *weight* of peer i is the relative size of the index of peer i, that is $w_i = \frac{x_i}{|D|}$. An item j has *low support* (is "rare") if s_j/n is small. An item has *low support with respect to the weights* if $\sum_{i\in S_j} w_i \ll 1$.

We view the peer-item matrix as a current instantiation of the data. We measure performance of different algorithms by treating each "1" entry, in turn, as the most recent request: For each peer i and item j such that $D_{ij} = 1$, we assume that peer i issues a query to item j when the state of the network is as indicated by the matrix D with the entry D_{ij} set to 0.[2] The search process is a sequence of probes: when a peer is probed, it attempts to match the query against its local index using some algorithm. We assume that this algorithm is perfect in the sense that a query for item j is always (and only) resolved by a probe to peer that contains item j.[3] The size of a search process (number of probes issued) for a particular algorithm A is a random variable. We define the *Expected Search Size* to be the expectation of this random variable. We denote this expectation by Ess_{ij}^{A}. (We shall omit the superscript A when the algorithm is clear from the context.) We compare different strategies by looking at all *queries* (peer-item pairs with $D_{ij} = 1$). We sweep a threshold on the maximum value of the Ess, and look at the cumulative fraction of queries (i, j) that have Ess_{ij} below a threshold.

3.1 Blind Search as Random Search

Following [19, 3] we model the performance of blind search in "traditional" unstructured networks using the *Random Search* model. The intuition of why this abstraction is valid is that the set of probed peers in a query in unstructured networks depends only on the overlay structure which is independent of the query or previous selections by the querying peer. Thus, on average, the effectiveness of each probe can not be better than that of probing a random peer. (For a discussion of the actual topologies created by unstructures peer-to-peer networks see [39].)

Formally we define the algorithm Uniform Random Search (Urand). According to Urand peer i resolves a query by probing peers uniformly at random until it finds a peer which has the answer. When it sends a probe it picks the target to be peer $\ell \neq i$ with probability $1/(n-1)$.

We present two associative search algorithms: Rapier and Greedy. We compare these algorithms to blind search and to each other. However, we must ensure that we do not compare "apples and oranges". The algorithms Rapier and Greedy are somewhat biased towards searching in peers with relatively many items. Thus, comparing these algorithms to a blind search that chooses peers uniformly at random would be unfair. One might suspect that the advantage shown experimentally is due to the choice of peers with many items, and does not reflect any other property of these algorithms.

To avoid this potential pitfall, we also consider a weighted version of the random search model where peers have different likelihood of receiving a probe. We use a weighted random search where the probability of probing peer ℓ is proportional to x_ℓ, the size of the index of peer ℓ.

Formally we define the algorithm Proportional Random Search (Prand). The algorithm Prand is identical to Urand except that peer i probes peer ℓ with probability $x_\ell/(|D| - x_i)$. We will show that Prand on average probes each peer with the same probability as Rapier and thus differences in performance between the two cannot be due to this bias.

It is easy to see that the Ess of Urand and Prand is a geometric random variable. Assume that peer i issues a query for item j. The probability that a single probe of Urand succeeds is

[2]Note that the search sequence does not depend on j, as we do not use query properties (such as meta-data terms) to determine where to search. We use j only to determine if a queried peer has the answer or not. See the introduction and conclusion sections for a discussion on this issue.

[3]This simplification is justified as the matching issue of queries to appropriate items is present with other architectures and is orthogonal to the core of our contribution.

$\frac{s_j-1}{n-1} = \frac{\sum_{k\neq i} D_{kj}}{n-1}$ and therefore

$$\mathrm{Ess}_{ij}^{\textsc{URand}} = \frac{n-1}{\sum_{k\neq i} D_{kj}} \ . \tag{1}$$

Similarly the probability of a single probe of PRand succeeds is $\frac{\sum_{k\neq i} x_k D_{kj}}{|D|-x_i}$ and therefore

$$\mathrm{Ess}_{ij}^{\textsc{PRand}} = \frac{(|D|-x_i)}{\sum_{k\neq i} x_k D_{kj}} \ . \tag{2}$$

Since in our model we use the peer-item associations also as the queries we have to assume that each item j appears in at least two different peers so that the probability of successful search is bounded away from zero for every possible query. This of course does not hold in real networks, in particular when an item is first introduced into the network. It is not a problem since in real life queries are different from peer-item associations.

4 Associative search

In this section we present the basic ingredients of our associative overlays framework, and present the Rapier strategy as a concrete instantiation of a search strategy using such overlays.

Recall that an associative overlay is defined over a guide rule which is a carefully chosen set of peers. Peers should agree upon the guide-rules being used. Therefore the set of guide-rules to use is best picked by a global entity maintaining the network. The particular choice of the set of guide-rules is constrained by both "networking" aspects, which require that the overlay has certain connectivity properties and can be formed and maintained at low cost, and the "data mining" aspects, which require that these rules meaningfully distill common interests; and thus, restricting the propagation of the query to peers within guide rules of the originating peer yields a more focused search. In Figure 1 we see a pictorial example of the sets of peers associated with two overlapping guide rules.

We focus on automatically-extracted guide rules of a particular form, which we call *possession rules*. Each possession rule corresponds to a particular data item and consists of all peers having a copy of the data item. Our underlying intuition, taken from extensive previous research in the Data-Mining and Text Retrieval communities (see, e.g.[20, 40, 41, 42, 43, 44, 45]), is that, on average, peers that share items (in particular rare items) are more likely to satisfy each other's queries than random peers. Note that in practice we can always define the entire P2P network as another guide-rule.[4] By doing this we guarantee that one can reach any peer even if it searches only within guide-rules.

Similarly to search in traditional unstructured networks, the query is propagated from peer to its neighbors, but propagation is restricted to peers belonging to a selected possession rule. To facilitate this search, we need each possession rule to have similar properties as a traditional unstructured overlay network. Namely, it should exhibit high expansion: so that by limited flooding we reach a large set of random peers.

Random constant degree graphs are known to have good connectivity and expansion properties. Traditional unstructured networks achieve their connectivity and expansion properties by resembling a random graph: Each peer selects a small "random" set of other peers as its neighbors. To

[4] Although we did not do that in our experiments.

have each possession rule exhibit similar expansion and connectivity, each peer, for each possession rule it is associated with, needs to remember a small list of "random" peers which belong to the same possession rule. The specifics can vary from a Gnutella-like design where each peer has few viable neighbors (Typical Gnutella number is 2-4) and many other peers can be reached through them, to a FastTrack-like design where search is facilitated through a core network of supernodes (in our case supernodes are associated with possession-rules).

Forming and maintaining this overlay requires a simple mechanism for peers to join a guide rule overlay. For possesion rules this is easily integrated with the search process. When a peer p finds an item X, it also reaches a peer q that holds X. Peer q already belongs to the possession rule corresponding to X. So q can hand over to p addresses of few peers in the possession rule of X from which p can reach even more peers that participate in X which is all p needs to join the possession rule of X.

Beyond the resolution of the search, possession rules provide an easy way to locate many other peers that share the item. This feature is useful for distributing the load of sending large files (parallel downloads are already practiced in FastTrack networks), or locating alternative download sites when a peer is temporarily swamped.

Our design is consistent with selfish peer behavior as participation of a peer in appropriate guide-rules serves dual purpose: it boosts the effectiveness of search through the peer, but also allows the peer to focus its own search process; peers that do not participate in guide-rules can not search more effectively than with blind search. In addition, processing of queries within a guide rule can help the peer update its neighbors within the rule.

Search strategy: A search strategy defines a search process as a sequence of guide rules and extent of search within each rule. For example: "search 100 peers that have item A and 200 peers that have item B, if this is unsuccessful, then search 400 more that have item A and 50 peers with item C, . . . " A search within each guide-rule is essentially "blind," and we model its performance the same way we model search in unstructured network, using random search [19, 3].

Our general expectation is that the total number of guide rules may be large, but a typical peer uses a relatively small number of rules. The applicability of a specific set of guide-rules depends on the implementability of the connectivity requirement. Relatively large guide-rules that resemble random graphs would be most effective.

There is seemingly a major issue in that a peer in a guide-rule network may keep track of many other peers, proportional to the number of guide rules it belongs to. Even when bounding the number of guide-rules a peer participates in, the number of neighbors is considerably larger than in existing architectures. This is in contradiction to the philosophy used by existing P2P architectures, which promotes having a small number of neighbors. We argue, however, that there is no reason for guided search to abide by this rule whereas there are clear reasons for other P2P architectures to keep it.

DHTs require neighbors to be accessible since they fill specific roles in routing certain queries. A neighbor going offline require communication to repair the region it was responsible for and involves some local restructuring of the network. Unstructured architectures use small number of neighbors, but from a different reasoning altogether: As search in these networks is essentially random, neighbors of your neighbors are as good as your neighbors; searches that go "deeper" are less likely to suffer from collisions, where peers get probed more than once on the same query; Low diameter can be obtained with low degree anyway; and low degree implies tighter association between hop-count limit (which is used as a stopping rule) and the total size of a search.

For our application, there is no particular advantage to impose severe limits on the number of neighbors. Unlike DHTs, we do not incur maintenance costs when neighbors go offline; we may discover it when trying to search through these peers and may then remove them from our list following one or more unsuccessful tries; replacements are easy to find if at least some of the guide-rule neighbors are active. More importantly, as our search process is guided, only a small number of peers will have the search request conveyed to them. The vertex induced subgraph of the peers in a single guide rule is still a low degree network.

In the sequel, we assume that our network is a possession-rule overlay. Each sub-overlay resembles an unstructured network and we model the search within a rule as a uniform random search. The bias towards peers with many items exists since such peers participate in a large number of guide rules.

4.1 Rapier strategy

Rapier search repeats the following until search is successful or resource limits exceeded:

1. Choose a random item from your index.
2. Perform a blind search on the possession-rule for the item to some fixed predetermined depth.[5]

Notice that in case the same item is picked more than once in this process we want that the blind search each time the item is picked would reach a different set of peers. One can approximately achieve that by starting each blind search from a different set of random peers within the overlay.

Random selection of a guide rule is particularly appealing since it is highly insensitive to neighbors going offline. Even though it is important to search within a rule, all rules that apply to the user are equally important. Thus, the search can be pursued even if a large number of neighbors on the list are unreachable.

Evidently, if there are no correlations between items, Rapier has no advantage over blind search. We use a two-pronged evaluation of Rapier: First, we use a simple intuitive data model (the Itemset model) to learn how the effectiveness of Rapier grows with the amount of "structure" in the data. Second, we evaluate Rapier on actual data, using large datasets of users accessing web sites. We obtained that Rapier performs significantly better than blind search, in particular, it is orders of magnitude more effective in searching for infrequent items.

The main parameter we look at is the expected size of the search which is the total number of peers probed. To estimate that **we model Rapier search by the following process:** For a query for item j issued by peer i, a column k is drawn uniformly from $X_i \setminus \{j\}$ (the index of i excluding j). Then a peer r is drawn uniformly from $S_k \setminus \{i\}$. The search is successful iff $D_{rj} = 1$, otherwise we repeat and pick another peer in exactly the same way.

That is we model the search within a possession-rule overlay by a random choice of a peer (other than peer i that issues the query) in the possession-rule overlay. This is consistent with our assumption that the search inside a guide-rule overlay is conducted as search in unstructured peer-to-peer network, and with the definition of Urand. In practice when we search within an overlay using flooding or random walk, not every probe reaches a new peer, but if the overlay resembles a random graph the overhead is low. We do not consider this overhead in our model, since we assume that it is similar for all the algorithms that we compare.

[5] This strategy can be "parallelized" by selecting a group of items from the index at each iteration.

The likelihood of success for RAPIER per step is

$$p_{ij} = (x_i - 1)^{-1} \sum_{k \in X_i \setminus \{j\}} \frac{s_{kj} - 1}{s_k - 1} \ , \tag{3}$$

and since the expected search size is a geometric random variable we obtain that

$$\mathrm{ESS}_{ij}^{\mathrm{RAPIER}} = 1/p_{ij} \ . \tag{4}$$

To ensure that the probability of successful search is bounded away from zero for every query, we have to require that for each item j and peer i, there is at least one other peer k containing j that has an item $z \neq j$ in common with peer i. As for PRAND this is a technical assumption which stems from the fact that we use the same peer-item associations to model our queries. It is somewhat stronger than our assumption for PRAND since we did not use the entire network as a guide-rule. In practice this assumption need not (and does not) hold.

4.2 RAPIER versus PRAND

As discussed earlier, search strategies may differ in the extent that they utilize peers of different index sizes. RAPIER, in particular, is more likely to probe peers with larger indices, since such peers participate in many guide rules. The fact that RAPIER tends to query heavy peers may alone give it an advantage over URAND. To check whether it really exploits associations between files held by peers, we introduced PRAND that also tend to search heavy peers but does not exploit associations. In this section we quantify the bias of RAPIER towards heavy peers and relate it to the bias of PRAND.

Given a query of peer i for item j PRAND probes peer k with probability proportional to x_k, the size of the index of peer k. That is we normalize the sizes x_k, for every $k \neq i$ by $|D| - x_i = \sum_{k \neq i} x_k$, and use the normalized sizes as a probability distribution from which a peer to probe is drawn. The following Lemma shows that if we average the probabilities that RAPIER probes peer i over all possible queries then the averaged probability is also proportional to x_i.

Lemma 4.1 *Averaged over all queries, the probability that* RAPIER *probes peer i first, when resolving a query is $w_i = x_i/|D|$.*

Proof. We assume that a query is drawn uniformly at random and then RAPIER draws a single peer to probe. We show that the probability that peer i is probed is w_i.

RAPIER can probe peer i only when it resolves a query initiated by a peer different from i, and use as a guide rule an item j that peer i holds. The probability that this event happens for a fixed guide rule j containing peer i is

$$\sum_{k \in S_j \setminus \{i\}} w_k \frac{x_k - 1}{x_k} \frac{1}{x_k - 1} = \sum_{k \in S_j \setminus \{i\}} \frac{1}{|D|} = \frac{s_j - 1}{|D|}$$

since w_k is the probability that the query is by peer k, $\frac{x_k - 1}{x_k}$ is the probability that when peer k issues a query it is for an item other than j, and $\frac{1}{x_k - 1}$ is the probability that when peer k issues a query to an item other than j it chooses j as the guide rule.

When a peer different from i issues a query and selects item j as the guide-rule it queries peer i with probability $\frac{1}{s_j-1}$. So for a fixed item j in the index of peer i the probability that i is probed through guide rule j is

$$\frac{s_j-1}{|D|}\frac{1}{s_j-1}=\frac{1}{|D|}\ .$$

Now observing that there are x_i possible guide rules through which peer i can be probed we obtain that the overall probability that RAPIER probes peer i is $w_i = x_i/|D|$. □

Remarks: If we average the probability that PRAND probes peer i over all possible queries we may not get $w_i = x_i/D$. This average equals

$$\sum_{k\neq i}\frac{x_k}{|D|}\frac{x_i}{|D|-x_k}=w_i\sum_{k\neq i}\frac{x_k}{|D|-x_k}$$

The factor $\sum_{k\neq i}\frac{x_k}{|D|-x_k}$ is one if the x_k's are uniform, and close to one if $|D|-x_k \approx |D|-x_i$ for every $k \neq i$, but in general it can be different from 1.

However Lemma 4.1 does show that the average (over all queries) probability that RAPIER probes peer i is proportional to the probability that PRAND probes peer i while resolving a query of a particular peer k (for any k). Thus providing some quantification that shows that PRAND and RAPIER have similar bias towards querying heavy peers.

4.3 RAPIER is no worse than PRAND

We define a relaxed version of RAPIER denoted by RAPIER'. RAPIER' is the same as RAPIER except that when resolving a query of peer i for item j we allow peer i to use the guide-rule that corresponds to the query item j itself, in which case RAPIER' finds the answer with probability 1. Furthermore within each guide-rule peer i can also choose itself as the peer to probe, in which case we again find the answer with probability 1. Another way to put it is to say that for each peer i that holds item j (an entry $D_{ij}=1$) we simulate a query by an imaginary peer i' to an imaginary item j'. The new peer i' contains the same items that peer i contains, and the item j' is stored in all peers that store j.

Similarly we define a relaxed version of PRAND, denoted PRAND'. When peer i issues a query for item j, PRAND' probes each peer ℓ, including peer i itself with probability $x_\ell/|D|$. If peer i is indeed drawn to be the peer to probe then PRAND' assumes that it answers the query.

We consider the apparently unrealistic RAPIER' and PRAND' since it is technically easier to compare the success probability of these two algorithms rather than the success probability of the real versions RAPIER and PRAND. Clearly the success probability of a probe of RAPIER' is larger than the success probability of a probe of RAPIER. However if we assume that each peer participates in at least K guide rules, and each guide-rule contains at least L peers then the success probability of RAPIER' is larger than the success probability of RAPIER by at most $\frac{1}{K}+\frac{1}{L}$. Similarly the success probability PRAND' is higher than the success probability of PRAND. But this bias is not large if the size of the index of each single peer is not too large with respect to the overall number of peer-item pairs in D.

We show that RAPIER' performs at least as well as PRAND' in the following sense: for each queried item, the average over peers of the per-probe success probability of RAPIER' is always at least as large as that of PRAND'. When each guide-rule contains at least L peers and each peer participates in at least K guide-rules it follows that the per-probe success probability of RAPIER is smaller than that of PRAND by at most an additive factor of $\frac{1}{K} + \frac{1}{L}$.

Lemma 4.2 *Consider a query issued for an item j. The average success probability, over all peers in S_j, of a probe picked by* RAPIER*' is no smaller than the average success probability of a probe picked by* PRAND*'.*

Proof: Consider a query for item q. It is easy to see that the success probability of a probe of PRAND' is $\frac{\sum_{i \in S_q} x_i}{|D|}$.

We now consider the success probability of a RAPIER' probe. The probability that the selected guide-rule is of item j is $\frac{s_{q \cap j}}{\sum_{i \in s_q} x_i}$. Assuming that RAPIER' picked guide rule j then the probability that we probe a peer that has item q is $s_{q \cap j}/s_j$. Therefore the success probability of a RAPIER' probe for item q is

$$\sum_j \frac{(s_{q \cap j})^2}{s_j \sum_{i \in s_q} x_i} \; .$$

To complete the proof of the lemma we need to show that

$$\sum_j \frac{(s_{q \cap j})^2}{s_j \sum_{i \in s_q} x_i} \geq \frac{\sum_{i \in S_q} x_i}{|D|} \; . \tag{5}$$

Denote $z_q = \sum_{i \in S_q} x_i$, and notice that $|D| = \sum_j s_j$ we obtain that (5) is equivalent to

$$\sum_j \frac{s^2_{q \cap j}}{s_j} \geq \frac{(z_q)^2}{\sum_j s_j} \; .$$

By substituting $z_q = \sum_j s_{j \cap q}$ we obtain

$$\sum_j \frac{s^2_{q \cap j}}{s_j} \geq \frac{(\sum_j s_{j \cap q})^2}{\sum_j s_j} \; .$$

Equivalently,

$$\sum_j (s_j) \sum_j (s^2_{q \cap j}/s_j) \geq (\sum_j s_{q \cap j})^2 .$$

Cauchy Schwartz inequality states that $(\sum a_j^2)(\sum c_j^2) \geq (\sum (a_j c_j))^2$. Substituting $a_j = \sqrt{b_j}$, $c_j = b_{q \cap j}/\sqrt{b_j}$ completes the proof. □

4.4 The Itemsets model

Frequency and size distributions of items and peers are reasonably-well understood. But even though these distributions capture enough aspects of the data to evaluate the performance of blind search, they do not capture correlations that are necessary for evaluating associative search. Models which capture correlations present in market-basket data and Web hyperlink structure had been proposed [25, 28, 46]. We use one such model, the *Itemsets model* (which resembles models in [25, 28]), to convey intuition why and when we anticipate RAPIER to perform well on real data.

The Itemsets model partitions the items into M sets (which we refer to as *itemsets*). For any itemset d_i, $1 \leq i \leq M$, and peer, independently, the peer decides to have items from d_i with probability $p(d_i)$. We assume that we have infinitely many peers. Our independence assumption implies that even if we restrict ourselves to peers that pick a certain itemset d then the probability that such a peer picks an itemset d' is still $p(d')$. We also assume that when a peer decides to have items of some itemset d then it a fraction f of the items in d picked uniformly at random. Let $k = \sum_{i=1}^{M} p(d_i)$. Then clearly k is the expected number of itemsets that each peer contains items of.

It follows that if we fix two items ℓ_1 and ℓ_2 and look at the set of peers S_{ℓ_1} containing item ℓ_1, and the set of peers S_{ℓ_2} containing item ℓ_2 then S_{ℓ_1} and S_{ℓ_2} would be correlated of ℓ_1 and ℓ_2 are in the same itemset and not correlated if ℓ_1 and ℓ_2 are in different itemsets. Our expectation is that if peers belong to many itemsets (at the extreme, all peers have all itemsets), there is no advantage for Rapier over Prand; however, we still expect Rapier to perform similarly to Prand. When peers belong to a small number of itemsets we expect Rapier to perform better; and we expect this advantage to increase as the number of itemsets decrease. We try to quantify this intuition below.

Consider a query of peer j to item ℓ which belongs to itemset $d(\ell)$. Assume that peer j contains items from exactly k different itemsets. Then it is easy to see that the success probability of a Prand probe is

$$R_\ell = fp(d(\ell)) \ . \tag{6}$$

On the other hand the probability of a Rapier probe is

$$C_\ell = \frac{f}{k} + \frac{k-1}{k} R_\ell = \frac{f}{k}(1 + (k-1)p(d(\ell))) \ , \tag{7}$$

The first term accounts for the case where the guide rule is in the same itemset as item ℓ. This happens with probability $1/k$ and assuming that it happens the query succeeds with probability f. The second term accounts for the case where the guide rule is from a different itemset. Our assumption that the support of different itemset are not correlated implies that the success probability in this case is the same as the success probability of Prand.

The ratio

$$\frac{1}{R_\ell} / \frac{1}{C_\ell} = C_\ell / R_\ell = \frac{1}{kp(d(\ell))} + \frac{k-1}{k}$$

is the ratio of the expected search size under Prand to the expected search size under Rapier for item ℓ by a peer that holds items from exactly k itemsets. Since $p(d(\ell)) \leq 1$, Rapier is always at least as effective as Prand in the itemset model. When $p(d(\ell)) \ll 1/k$, Rapier is much more efficient than Prand.

This simplistic model provides some intuition to when Rapier is more effective than Prand: Rapier benefits, when users interests are more "focused" (small k) and for items in rare itemsets (small $p(d(\ell))$).

The difference between Prand and Rapier is further illustrated by considering itemsets that are equally popular, that is, $p(d) = k/M$ for every itemset d. Substituting into Equation (6) we obtain that the success probability of the probe picked by Prand is fk/M and by substituting into Equation (7) we obtain that the success probability of a probe picked by Rapier is $\frac{f}{k} + \frac{f(k-1)}{M}$. We note that the success likelihood of Prand decreases to 0 when M increases whereas that of Rapier is bounded by a constant independent of M. The performance ratio between Rapier and Prand is $M/k^2 + (k-1)/k$ and thus Rapier dominates when $k \ll \sqrt{M}$. Our intuition says that M can be very large on a large diverse human population, but k (the interest areas of a single human being) is inherently bounded, thus, Rapier would perform better with large number of peers.

5 Greedy: Optimizing the Search Strategy

The random choice of a guide rule is clearly non-optimal. Some guide rules may contribute much more to the probability of a successful search than others. Unsuccessful searches also contribute important information. For example, imagine that we've already searched extensively in one area of interest, this greatly increases the probability that the item being searched for actually belongs to another area of interest. In this section we describe how one can optimize the search strategy, and find a better sequence of guide rules to be probed.

We model the problem with a matrix $M \in R^{m \times m}$ with one row per guide rule and one column per data item, M_{jk} is the probability that a uniformly chosen random peer ℓ belonging to guide rule j, contains item k. Using the notation of Section 3 we have that $M_{jk} = \frac{s_{jk}}{s_j}$. Note that M is not necessarily a stochastic matrix (neither rows nor columns necessarily sum to one).

We also assume that each peer has a *query distribution*: $q =< q_1, q_2, \ldots, q_m >$, where $\sum_{i=1}^m q_i = 1$. We interpret q_i as the probability that a query issued by the peer is for item i.

In general, a peer does not know the matrix M and its query distribution q. So a peer deploying a Greedy strategy will use 1) An estimate of a submatrix of M with rows corresponding to guide-rules the peer participates in; and 2) an estimate of its query distribution q. In Section 5.3 we discuss which M and q we use in our simulation and how would a peer estimate M and q in practice. For now lets just assume that each peer has such estimates and show how to use them to build a search strategy.

It follows from our definitions that the success probability of random probe from guide-rule i on a query drawn from q is the inner product $(M_i, q) = \sum_{k=1}^m M_{ik} q_k$, where M_i denoted the i'th row of the matrix M. Given a (multi) sequence of guide-rules[6] $\sigma = \langle \sigma_1, \sigma_2, \ldots \rangle$ where $\sigma_i \in \{1, \ldots, m\}$ and a query distribution q, an *experiment* is a random choice of an item x to search for, according to the distribution q and an application of probes from the guide-rules in σ to x, one from each guide-rule in order until an answer is found. Let $\sigma_i(x)$ be an indicator random variable that is one if and only if the probe from guide rule σ_i has item x. The outcome of an experiment for q and σ is a pair (x, j) where x was chosen from q and $1 \leq j \leq |\sigma|$ is derived as follows. Let $S \subset \{1, \ldots, |\sigma|\}$ be a random variable, $S = \{i | \sigma_i(x) = 1\}$, we define $j = \min S$ if $S \neq \emptyset$ and $j = |\sigma|$ if $S = \emptyset$. Let $D(\sigma)$ be the distribution over outcomes defined by q and σ.

If $S \neq \emptyset$ we say that the experiment was successful, if $S = \emptyset$ then the experiment is unsuccessful. We denote by $P(\sigma)$ the probability that an experiment is successful. This is the probability that sequentially trying a probe of $\sigma_1, \sigma_2, \ldots$ to answer the (randomly chosen) query will succeed.

We denote the prefix of length i of σ by $\mathrm{pref}(i, \sigma)$. I.e. $\mathrm{pref}(i, \sigma) = \langle \sigma_1, \sigma_2, \ldots, \sigma_i \rangle$. It follows from our definitions that

$$P(\sigma) = \sum_{i=1}^{|\sigma|} P(\mathrm{pref}(i, \sigma)) - P(\mathrm{pref}(i-1, \sigma))$$

[6]Each guide-rule may appear more than once in the sequence.

$$= \sum_{i=1}^{|\sigma|} (M_{\sigma_i}\ , q * (\bar{1} - M_{\sigma_1}) * (\bar{1} - M_{\sigma_2}) * \cdots * (\bar{1} - M_{\sigma_{i-1}})) \tag{8}$$

Recall that M_{σ_i} denotes the row in M which corresponds to guide rule σ_i, $\bar{1}$ denotes the all-one vector, and $v * u$ denotes the Hadamard product (component wise multiplication) of v and u. (Note that the sum above may be infinite.)

We will be interested in sequences σ such that $P(\sigma) \geq f$. Let $T^{f,\sigma}$ be the minimum ℓ such that $P(\mathrm{pref}(\ell, \sigma)) \geq f$, define $T^{f,\sigma} = \infty$ if σ is infinite and $P(\sigma) = f$. We denote by $E(f, \sigma)$ the expected number of probes that we send when we try to answer a query by first applying σ_1, if it fails then applying σ_2, up to $\sigma_{T^{f,\sigma}}$. More formally,

$$E(f, \sigma) = \sum_{\substack{(q,i) \in D(\sigma), \\ i < T^{f,\sigma}}} i \cdot \mathrm{Prob}((q, i)) + T^{f,\sigma} \cdot \sum_{\substack{(q,i) \in D(\sigma), \\ i \geq T^{f,\sigma}}} \mathrm{Prob}((q, i)) \ ,$$

where $\mathrm{Prob}((q, j))$ is the probability over $D(\sigma)$ of the outcome (q, j). Recall that an application of $\sigma_1, \ldots, \sigma_{T^{f,\sigma}}$ on a query can be successful or unsuccessful, where the probability that it is successful is $\geq f$.

Given f, q, and M we define the optimal sequence of guide-rules, σ^{opt}, to be the sequence such that $E(f, \sigma^{opt}) \leq E(f, \sigma)$ for every sequence σ such that $P(\sigma) \geq f$. A particular special case of this optimization problem is when $f = 1$, which means that we restrict ourselves to sequences of probes that would find the item with probability one.

5.1 The Greedy search sequence

Finding an optimal sequence of guide-rules is a hard problem (see Section 5.2). On the positive side, a simple Greedy strategy gives a sequence whose expected number of probes is within a small constant factor of that of the best sequence. In this section we define the Greedy sequence of guide-rules for M and $\mathbf{q}$. We refer to the algorithm constructing it as the Greedy algorithm.

The Greedy algorithm computes its strategy by repeatedly finding the guide-rule j which maximizes the likelihood of resolving a query drawn from q, and then updating q to the corresponding posterior distribution obtained if the query is not resolved after probing guide-rule j. Specifically, the first guide rule which Greedy chooses, denoted by σ_1^g, is the guide rule j that maximizes (M_j, q).

Assuming that the first experiment of Greedy (or any other experiment) turns out to be unsuccessful we can update the query distribution q to take into account the unsuccessful probe. For a guide rule j and a query q, we define the *posterior* query distribution to be $q' =< q'_1, q'_2, \ldots, q'_m >$, where q'_k is the probability that the query is for item k given that "probe of guide-rule j for q" failed. One can calculate q'_k as follows.

$$\begin{aligned} q'_k &= \mathrm{Prob}\,(\text{query is for item k} \mid \text{probe of guide rule } j \text{ for } q \text{ unsuccessful}) \\ &= \frac{\mathrm{Prob}\begin{pmatrix} (\text{query is for item k}) \cap \\ (\text{probe of guide rule } j \text{ for } q \text{ unsuccessful}) \end{pmatrix}}{\mathrm{Prob}\,(\text{probe } j \text{ for } q \text{ unsuccessful})} \\ &= \frac{q_k(1 - M_{jk})}{1 - (M_j, q)}. \end{aligned} \tag{9}$$

The second guide rule which Greedy chooses, denoted by σ_2^g, is the guide rule that maximizes the success probability with respect to the query vector q'. Specifically, Greedy picks as its second guide-rule, σ_2^g, the guide rule j that maximizes (M_j, q'). In general Greedy repeats the following steps.

1. Find the index j which maximizes $(M_j, q) = \sum_{k=1}^{m} M_{jk} q_k$. Output j.
2. Compute $q' = (q'_1, q'_2, \ldots, q'_m)$ where

$$q'_k = \frac{q_k(1 - M_{jk})}{1 - (M_j, q)} \ .$$

   Set $q \leftarrow q'$.

As an example consider the matrix M and the query distribution q depicted in Figure 2. It is easy to see that the probability of a probe from any of the guide rules to succeed is $1/4$. So lets assume that Greedy would break the tie by picking σ_1^g to be guide rule 1. The posterior query distribution is $q' = (2/9, 2/9, 2/9, 1/3)$. Now with respect to this query distribution a probe of guide-rule 5 maximizes the success probability. So Greedy continues setting σ_2^g to be 5. The posterior query distribution after probing a peer from guide-rule 1 and a peer from guide-rule 5 is $q'' = (1/3, 1/3, 1/3, 0)$. Again all guide-rules have the same success probability with respect to this query distribution. So Greedy may pick $\sigma_3^g = 1$. The posterior query distribution $q^{(3)}$ remains $(1/3, 1/3, 1/3, 0)$. So Greedy may continue and set $\sigma_i^g = 1$, for every $i \geq 3$.

Lets assume that $f = 1$ so $T^{f,\sigma^g} = \infty$, and calculate the expected number of probes that we issue using σ^g. The first probe succeeds with probability $((1/3, 1/3, 1/3, 0), (1/4, 1/4, 1/4, 1/4)) = 1/4$. Assuming that the first probe fails the second one succeeds with probability $1/3$, and assuming that the first i probes failed then the $i{+}1^{st}$ probe succeeds with probability $((1/3, 1/3, 1/3, 0), (1/4, 1/4, 1/4, 1/4)) = 1/4$. So

$$E(1, \sigma^g) = 1 + \frac{3}{4} + \frac{3}{4}\frac{2}{3} + \sum_{i=1}^{\infty} \frac{3}{4}\frac{2}{3}(\frac{3}{4})^i = 3\frac{3}{4} \ .$$

Since using the Greedy sequence we always make 1 probe, we make a second with probability $\frac{3}{4}$ which is the probability that the first probe fails. We make the third probe with probability $\frac{3}{4}\frac{2}{3}$ which is the probability that the first and the second probes fail and so on. Notice that the sequence $\sigma = 2, 3, 4, 5$ also has success probability $P(\sigma) = 1$. However $E(1, \sigma) = (1 + 2 + 3 + 4)/4 = 2.5$ which is better than the average number of probes used by the Greedy sequence.

5.2 Theoretical background

The problem of finding the sequence that minimizes the expected number of probes for a given matrix M, query distribution q, and covering fraction f, is a generalization of a problem which has been studied recently, under different names, one of which is Min-Sum Set Cover (MSSC).

The input to the MSSC problem (like the input to the classical Set Cover problem) consists of a collection of sets $\mathcal{F} = \{S_1, \ldots, S_m\}$ containing elements from some universe $\mathcal{U}$. The goal is to find a permutation $S_{\pi(1)}, S_{\pi(2)}, \ldots, S_{\pi(m)}$ that minimize the average *covering time* $f(e)$ for all $e \in \mathcal{U}$. The covering time of $e \in \mathcal{U}$ is the minimum j such that $e \in S_{\pi(j)}$.

Our problem degenerates to the MSSC when the matrix M is a zero-one matrix, the query distribution is uniform, and $f = 1$. I.e. for each guide rule ℓ and item i, a probe from ℓ either has i

with probability 1 or does not have i with probability 1. When this is the input then we can think of each guide rule as a set; that is the set of the items that each probe of that guide rule has. An optimal sequence of guide-rules corresponds to a sequence of the sets that minimizes the average covering time.

The MSSC problem is related to the Min Sum Coloring (MSC) problem where the input is a graph and the output is a linear ordering of its independent sets so that if we color each vertex with the index of the first independent set containing it then the sum of the colors is minimized. This problem models the issue of minimizing response time in a distributed task allocation system, where the graph models conflicts between tasks. There is an easy reduction from MSC to MSSC which makes the independent sets of the graph the sets of the corresponding MSSC instance, but obviously this reduction is not polynomial in the size of the graph. In fact, the MSC problem turns out to be harder than MSSC due to the need to find independent sets in the graph.

Bar-Noy et al. [47] show that the greedy algorithm for MSC which iteratively picks the largest independent set is a 4-approximation algorithm. Their proof also applies to MSSC and shows that the greedy algorithm of set cover in fact gives a 4-approximation for MSSC. Bar-Noy, Halldorson and Kortsarz [48] give examples where the performance of greedy for MSC is tight. Their examples apply also to MSSC. More recently Feige, Lovasz, and Tetali [49] gave a simpler proof that greedy is a 4-approximation for MSSC and also showed that it is NP-hard to approximate this problem to within a factor smaller than 4.

While not being aware of the work of Bar-Noy et al. [47] and Feige et al. [49] we showed [50] that the Greedy algorithm in the more general setting of probabilistic guide rules, arbitrary query distribution and coverage f, as defined in Section 5.1 is within a constant factor of optimal. However a careful examination of the proof of Feige et al. shows that it can be generalized to establish that the Greedy algorithm in our more general setting is in fact a 4-approximation (that is better than the factor established in [50]). It also follows from the work of Feige et. al. that no algorithm can achieve a better approximation guarantee than Greedy for our problem unless $P = NP$.

5.3 Estimating M and $\mathbf{q}$

Recall that X_i denotes the index of peer i. Consider a peer i, searching for item j. This peer chooses $\mathbf{q}$ and M based on items in $X_i \setminus \{j\}$, compute the Greedy sequence using its choice of $\mathbf{q}$ and M and apply it for the search.

The q which we assume in our simulation, which is also a reasonable one for a peer to use in practice is a uniform distribution $q_k = 1/(x_i - 1)$ over all items $k \in X_i \setminus \{j\}$ (all other entries are 0).[7]

The set of guide-rules we use (rows of the matrix M) correspond to the items in $X_i \setminus \{j\}$. Let M_k be the guide-rule that corresponds to the kth item in $X_i \setminus \{j\}$. For each pair of items k and ℓ in $X_i \setminus \{j\}$ ($k \neq \ell$) we used in our simulation $M_{k\ell} = (s_{k\ell} - 1)/(s_k - 1)$, which is simply the success likelihood of probing for ℓ according to possession-rule of the item k.

Extending the definition above to the case where $k = \ell$ would imply that we set the "diagonal" entries in M to 1. However we defined $M_{kk} = 0$ for every k. This heuristic is intended to circumvents over fitting: Greedy uses the same set of items to construct both the query and the guide-rules, and diagonal entries of 1 would result in Greedy obtaining a search strategy of size at most the

[7] Although we assume a uniform initial query distribution we considered the problem for general query distributions in Section 5.1. This was essential since Greedy modifies the query probability distribution as unsuccessful searches are performed. The posterior probability distribution is arbitrary even if the initial one is uniform.

size of the index that is not likely to perform well on the "removed" item j. Setting the diagonal to 0 is similar to valuing each guide-rule M_k according to how well it performs on items *other than* k.[8]

Of course, in practice peer i does not know $s_{k\ell}$ and s_k and it does not have the matrix M that we use in our model. So to apply GREEDY each peer needs to collect information on how well its guide-rules perform on items in its index. (To save resources a peer can also work with a subset of its guide-rules, and estimate their effectiveness on a subset of the items in its index.) It could do that by initiating artificial searches for items that it has, using blind-searches in different guide-rule overlays. Using the results of these searches the peer would know how effective is each guide-rule in resolving each item. The accuracy of the estimate depends on the size of the search, but fortunately, the analysis in [49, 51] shows that for the purposes of obtaining a good approximation with GREEDY, it is sufficient to estimate only the "large" entries. This means that the first steps proposed by the GREEDY strategy are likely to be more accurate than later steps; and specifically this means that GREEDY is more likely to be practical if its benefit can be reaped in the first few steps of the search.

In the next section we evaluate the performance of GREEDY against other search algorithms. In our simulations we ran GREEDY exactly as defined in Section 5.1 using M and q as defined above. However in a practical setting one might want to issue more than single probe for each guide-rule σ_i^g in the GREEDY sequence. We can for example perform a blind search to some predetermined depth in the sub-overlay of σ_i^g. If that search fails proceed to do the same in the sub-overlay of σ_{i+1}^g. If $\sigma_j^g = \sigma_i^g$ for some $j > i$ and we reach σ_j^g while conducting the search according to the GREEDY sequence, we want to probe different set of items in σ_j^g than the ones which we probed at the same guide-rule at step i. We can achieve that by starting from a different set of peers and/or increasing the depth of the blind search within the overlay.

6 Data and Simulation Results

The experimental evaluation requires an index of a large number of peers. Unfortunately, peer-item data available from currently active distributed P2P networks reflects the limited ability to locate rare items, and Napster data is not publicly available. We thus opted to use similarly-structured ("market-basket") data of Web proxy logs obtained from AT&T Research and Boeing [52]; both sets of servers constitute lower-level proxies serving end users. From each log we extracted the matrix of users vs hostnames [9] and eliminated all singletons (hostnames accessed by only one user and users that accessed only one hostname), which resulted in about 3%-5% decrease in the size of the matrix. Since the Boeing log was sanitized on a daily basis (both users and hostnames), we use each day separately. The resulting sizes of the data matrices are depicted in Table 1. In the sequel, we refer to users as *peers* and to hostnames as *items*.

As typical for such data, we observed high skew in both the size of the index peers have (see Figure 3) and the support-size of items (see Figure 4), with large fraction of peers having small

[8] An alternative approach we considered for GREEDY is to partition items for those used for guide-rules and those used for queries, but this is too restrictive for peers with small index (since good guide-rules might lie in the "query" set and not be available for the strategy.).

[9] Another natural choice is the URLs vs users matrix, but URLs exhibit very high correlations, that would lead to (perhaps artificially) very good performance results for our algorithms, as each Web pages consists of multiple URLs that are automatically fetched together. Aggregating to the hostname (Web site) level may better reflect the "human factor" of the selection process.

index sizes and large fraction of items being present at a small fraction of peers. Figure 5 shows a cumulative fraction of queries that are issued for items with small support; Although 60% of queries are issued to items whose support is over 0.01 of peers, the remaining 40% of queries target unpopular items.

We first evaluated the performance of 4 search algorithms.

- Urand, where the Ess was computed according to Equation (1).
- Prand, where the Ess was computed according to Equation (2).
- Rapier, where the Ess was computed according to Equations (3) and (4).
- A hybrid of Prand and Rapier where each step is 50% likely to be according to Rapier and 50% likely to be according to Prand. The success probability per step is the mean of the two searches; thus this hybrid provides "insurance" against long searches as the Ess on any particular query is at most twice that of the better algorithm.

Due to the large size of Boeing matrix, we computed performance on a sample of the queries. We took a random sample of 1800 items and computed search performance, across all peers, on queries issued to one of these items (we had total of 53K queries). The results of the simulations are shown in Figure 6 (We only show the result of the Boeing day-3 log - results for other days were very similar) and Figure 7 (ATT data). The figures show a cumulative fraction of queries that have Ess below a certain threshold. They show the performance for items across support levels and also focus on items that have lower support (occur in the index of at most 10^{-2}-10^{-4} of peers). The figures show that Urand is the worst algorithm out of the 4. The Ess of Urand on an item is the inverse of the fraction of peers that index it, thus, when focusing on items occurring in at most 10^{-4} of users, the respective Ess is over 10K, and the Urand curve coincides with the x-axis. The Prand strategy that prefers peers with larger index sizes manages to see more items in each probe and performs considerably better than Urand, across items of different support levels. On the ATT dataset, each Urand search saw on average an index size of 69 per probe whereas Rapier and Prand saw 205 items in each probe (recall that averaged over searches, Rapier and Prand have approximately the same visit rate to each peer and thus the same average index size seen per probe. See Section 4.2). Similarly, the respective averages on the Boeing datasets, were 21 for Urand and 133 for Prand and Rapier.

We observe that Rapier, which has the same bias towards peers with larger index as Prand, outperform Prand; moreover, the performance gap is significantly more pronounced for items with low support. This indicates strong presence of the semantic structure Rapier is designed to exploit.

We also observe that the performance gap between Rapier and Prand is larger for the Boeing dataset than in the ATT dataset; this is explained by the much larger number of peers in the Boeing data which allows the existence of items with very low support.

For a typical Gnutella search size, estimated to cover about 1000 peers, the simulations on the Boeing dataset show that Rapier covers 52% of queries made to items that are present in at most 10^{-4} fraction of peers, whereas Prand covers only 14% of queries. Out of all queries, Rapier covers 95% and Prand covers 90%. On a smaller search size of a 100, Rapier and Prand, respectively, cover 30% and 1.3% of items with support below 10^{-4} fraction of peers, and cover 90% and 80% of all items. For search sizes where Prand covers most queries, Rapier obtains about half the failure rate of Prand.

We next describe an evaluation of Greedy against the 4 other search strategies. Our evaluation of Greedy required to compute the joint support of all pairs of items a peer has. We thus took a random sample of 866 peers that had index sizes between 20 and 30. As we can deduce from Figure 3 this is about 20% of the peers of that index size. Our results were robust to variations in this sample. The sample involved 5K distinct items, 147K distinct pairs of items that shared at least one peer, and 21K queries. We calculated the performance of each of the search strategies for all queries issued by the sampled peers; but used the complete matrix (all peers and their indexes) in each calculation. This simulation is more restrictive than the previous one since it focuses only on peers of certain sizes. The results, which are shown in Figure 8, however, show similar performance gaps to what we observed earlier for the 4 strategies across all peers. The figures also show that Greedy is much more effective than Rapier for small-size searches but if the query is not resolved then it has no advantage in subsequent steps. The reasoning for this behavior is that the Greedy strategy tends to converge; after many steps it iterates between a small number of guide-rules; queries that are not effectively covered by this small set of guide-rules have a better bet of getting covered by the broader selection of guide-rules provided by Rapier.

As discussed in Section 5, Greedy is considerably more involved than Rapier, but the results suggest that most of the benefit of Greedy is reaped on the first few probes (small search size). This suggests that a hybrid strategy that first applies Greedy with a small-size search then (if needed) continues with Rapier with a larger-size search can be effective.

7 Relation to classic search techniques

After laying out our design and simulation results, we can substantiate our earlier claim that guide-rules capture some essential properties of classic search techniques, while circumventing the difficulties of applying these techniques in a distributed setting.

There has been a great deal of work in the text retrieval and data mining literature on identifying "areas of interest", primarily through the use of algebraic techniques such as the Singular Value Decomposition (SVD) used in latent semantic analysis [20]. In this context, items and queries are mapped to some lower dimension vector space, and similarities between items or queries and items correspond to distances of the respective vectors. This "mapping" is then used to answer queries and cluster similar documents.

A major hurdle in using SVD is that it is not suited for our decentralized setting where every location (peer) stores a single row of the matrix.[10] If we assume that an SVD or SVD-like decomposition is somehow available, we can conceive of several design approaches for the P2P overlay. One approach is to apply SVD when the columns of the matrix correspond to meta-data terms. Each peer then produces a vector which corresponds to its meta-data terms and each query is represented by the vector of its query terms. SVD can then be used to map these vectors to a low-dimensional vector space. The search process is then tailored to the query terms in the query and we would like to route it to peers such that their term vectors are "closer" to the query vector in terms of distance in this metric space. An alternative approach is to apply SVD to the peers versus items matrix. In this case, the search is tailored to the originating peer, and the goal is to route it to peers that are "close" to the originating peer.

[10] Recent research demonstrated that sampling techniques can yield a decomposition with potentially similar properties in a distributed setting, but this approach involves large "constant factors" and is far from being deployable.

The next challenge is the construction of an overlay that supports this routing, that is, would be able to focus on the relevant neighborhood in this metric space. Our possession rules approach can be viewed as "limiting" the search to peers that are more likely to be close in this metric space, but without explicitly computing SVD; as we "work" with the "raw" matrix of peer versus items. The matrix has very high dimension, making nearest neighbor searches inefficient, but we circumvent these issues by searching within rules, thus restricting the search to peers that are "likely" to be close.

An issue that also arises in classic applications is that finding a set of "close matches" (nearest neighbors in the metric space) is insufficient: What is needed is to reach a set of peers that jointly offer a comprehensive coverage of the "interest areas" of the originating peer. We illustrate this issue through an example: consider a peer with interests (and queries) that are 0.5 "basketball" and 0.5 "Prince titles," and suppose that these two interest areas are fairly independent, with 0.2 fraction of peers having "basketball" interest and 0.01 fraction having the "Prince" interest. A simple similarity-based matching in the vector space will find many "basketball" peers before it finds even one "Prince" peer. Intuitively, what would work better is to distribute the search effort according the the originating peer "interests distribution:" that is, to focus 50% of our search effort in the "basketball" peers and 50% of the effort in the "Prince" peers.

The classic solution to this issue [21, 22, 23] is to cluster all close matches, and make sure that all clusters, even reasonably small ones (in terms of number of participating peers), are represented in the response; e.g., somehow group all the peers that match on "basketball" and all the peers that match on 'Prince", and equally query peers in each set. The eigenvalue computations used for computing these clusterings are not suited for a distributed setting. Our possession-rules overlay circumvents the need for explicit clustering by providing a way to obtain representatives from the "clusters:" The Rapier algorithm effectively concentrates equal effort in "basketball" and "Prince," regardless of the relative overall popularity of these two interests.

8 Conclusion and Future Work

We argued that associative overlays retain the advantages of unstructured architectures (such as Gnutella and FastTrack); including relative insensitivity to peer failures and support for partial-match queries; but can offer orders of magnitude improvement in the scalability of locating infrequent items. Our design exploits presence of associations in the underlying data. Such associations were previously exploited for Web search, Data-mining, and collaborative filtering applications, but the techniques were not portable to the P2P setting which requires simple, resilient, and fully decentralized protocols. Our approach can be viewed as maintaining the essence of these techniques while striking a balance with the challenges of the P2P setting.

We showed that Rapier, the simplest search strategy on possession-rule overlays, can significantly increase the effectiveness of search for rare items over that of plain unstructured networks. It is likely that better search performance on possession-rule overlays can be achieved by preferring rules that correspond to recently acquired items or rules where the meta data of the corresponding items is more related to the query terms.

The more involved Greedy strategy offers a more fine-tuned search by approximating the strategy that would have performed best on previous queries. Its design uncovers the more effective guide rules and takes into account dependencies in the coverage of different rules. We observed that Greedy outperforms Rapier but constructing it incurs overhead of performing "test" searches.

References

[1] E. Cohen, A. Fiat, and H. Kaplan. Associative search in peer to peer networks: Harnessing latent semantics. In *Proceedings IEEE INFOCOM 2003*, 2003.

[2] Napster Inc. The napster homepage. In *http://www.napster.com/*, 2001.

[3] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th annual ACM International Conference on supercomputing*, 2002.

[4] J. Risson and T. Moors. Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks*, In press, Available online 3 March 2006.

[5] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, 2005.

[6] Gnutella. `http://en.wikipedia.org/wiki/Gnutella`.

[7] FastTrack Peer-to-Peer technology company. FastTrack. In *http://www.fasttrack.nu/*, 2001.

[8] Morpheus file sharing system. Morpheus. In *http://www.musiccity.com/*, 2002.

[9] KaZaA file sharing network. KaZaA. In *http://www.kazaa.com/*, 2002.

[10] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of SIGCOMM'2001*, 2001.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proceedings of SIGCOMM'2001*, 2001.

[12] A. Rowstron and P. Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of SOSP'01*, 2001.

[13] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Department, 2001.

[14] Open Source Community. The free network project - rewiring the internet. In *http://freenet.sourceforge.net/*, 2001.

[15] A. Fiat and J. Saia. Censorship resistant peer-to-peer content addressable networks. In *Symposium on Discrete Algorithms*, 2001.

[16] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *ICDE*, 2002.

[17] S. Joseph. Neurogrid: Semantically routing queries in peer-to-peer networks. In *Proceedings of the International Workshop on Peer-to-Peer Computing*, 2002.

[18] B. F. Cooper and H. Garcia-Molina. Sil: A model for analyzing scalable peer-to-peer search networks. *Computer Networks*, 50(13):2380–2400, 200.

[19] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *Proceedings of the ACM SIGCOMM'02 Conference*, 2002.

[20] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

[21] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[22] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the WWW7 Conference*, 1998.

[23] D. Achlioptas, A. Fiat, A. Karlin, and F. McSherry. Spectral analysis of data. In *Symposium on Theoretical Computer Science, pp. 619-626*, 2001.

[24] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[25] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases*, pages 487–499. Morgan Kaufmann, 1994.

[26] ACM SIGGROUP resource page on collaborative filtering. `http://www.acm.org/siggroup/collab.html`.

[27] D. Goldberg, D. Nichols, Oki B. M., and D. Terry. Using collaborative filtering to weave an information tapestry. In *Communications of the ACM, 35(12)*, pages 51–60, 1992.

[28] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. In *IEEE Symposium on Foundations of Computer Science*, pages 664–673, 1998.

[29] Y. Azar, A. Fiat, A. R. Karlin, F. McSherry, and J. Saia. Web search through hub synthesis. In *Symposium on Foundations of Computer Science*, 2001.

[30] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. D. Ullman, and C. Yang. Finding interesting associations without support pruning. *Knowledge and Data Engineering*, 13(1):64–78, 2001.

[31] K. Sripanidkulchai, B. M. Maggs, and H. Zhang. Efficient content location using interest-based locality in peer-to-peer systems. In *INFOCOM*, 2003.

[32] K. Sripanidkulchai, B. Maggs, and H. Zhang. Enabling efficient content location and retrieval in peer-to-peer systems by exploiting locality in interests. *ACM SIGCOMM Computer Communication Review*, 32(1), January 2002.

[33] R. Ahmed and R. Boutaba. Distributed pattern matching for P2P systems. In *IEEE/IFIP Network Operations & Management Symposium*, 2006.

[34] R. Zhang and Y. C. Hu. Assisted peer-to-peer search with partial indexing. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1514– 1525, 2005.

[35] C. Tang, Z. Xu, and M. Mahalingam. Peersearch: Efficient information retrieval in peer-to-peer networks. In *Proceedings of HotNets-I, ACM SIGCOMM*, 2002.

[36] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: collaborative p2p search. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 1263–1266. VLDB Endowment, 2005.

[37] M. Bawa, G. S. Manku, and P. Raghavan. Sets: search enhanced by topic segmentation. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 306–313, New York, NY, USA, 2003. ACM Press.

[38] Y. Zhu, X. Yang, and Y. Hu. Making search efficient on gnutella-like p2p systems. In *IPDPS*. IEEE Computer Society, 2005.

[39] W. Acosta and S. Chandra. Unstructured peer-to-peer networks - next generation of performance and reliability. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 2005.

[40] S. Chakrabarti, B. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Spectral filtering for resource discovery, 1998.

[41] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. *Computer Networks and ISDN Systems*, 30(1–7):65–74, 1998.

[42] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Hypersearching the web. *Scientific American*, June 1999.

[43] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Research and Development in Information Retrieval*, pages 104–111, 1998.

[44] T. Hofmann. Probabilistic latent semantic indexing. In *Research and Development in Information Retrieval*, pages 50–57, 1999.

[45] J. Dean and M. R. Henzinger. Finding related pages in the world wide web. *WWW8 / Computer Networks*, 31(11-16):1467–1479, 1999.

[46] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the Web. In *Proceedings of 25th International Conference on Very Large Data Bases*, pages 639–650, 1999.

[47] A. Bar-Noy, M. Bellare, M. M. Halldórss on, H. Shachnai, and T. Tamir. On chromatic sums and distributed resource allocation. *Inf. Comput.*, 140(2):183–202, 1998.

[48] A. Bar-Noy, M. M. Halldórsson, and G. Kortsar z. A matched approximation bound for the sum of a greedy coloring. *Information Processing Letters*, 71(3-4):135–140, 1999.

[49] U. Feige, L. Lovász, and P. Tetali. Approximating min-sum set cover. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 94–107, 2002.

[50] E. Cohen, A. Fiat, and H. Kaplan. Efficient sequences of trials. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 737–746, 2003.

[51] E. Cohen, A. Fiat, and H. Kaplan. Efficient sequences of trials. In *Proc. 14th ACM-SIAM Symposium on Discrete Algorithms*. ACM-SIAM, 2003.

[52] Web traces and logs.
`http://www.web-caching.com/traces-logs.html`.

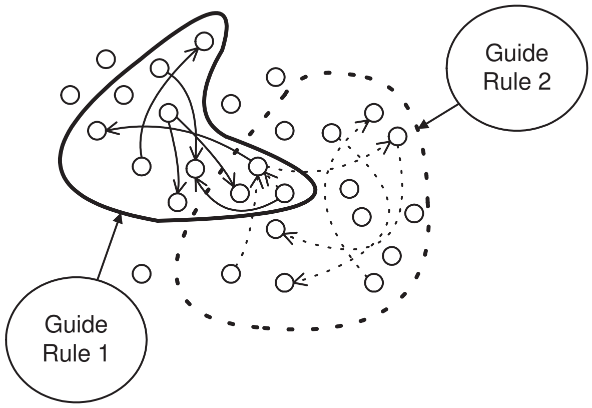

Figure 1: Two guide rules. In graph-theoretic language, any algorithm seeking to exploit these guide rules must ensure that the vertex induced subgraph of the vertices belonging to a guide rule consists of large strongly connected components with large expansion properties. The outgoing edges of every peer are labeled with the guide rule to which they belong.

$$M = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad q = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix}$$

Figure 2: An example of a matrix of guide rules M and a query distribution q. The sequence GREEDY computes for this input is $1, 5, 1, 1, \ldots$.

log	peers	items	(peer,item)
ATT (18 days, Nov 8-25,96)	432	4.4K	30K
Boeing-1 (1 day, March 1,99)	57K	46K	115K
Boeing-2 (1 day, March 2,99)	57K	46K	115K
Boeing-3 (1 day, March 3,99)	57K	45K	119K

Table 1: Description of the data

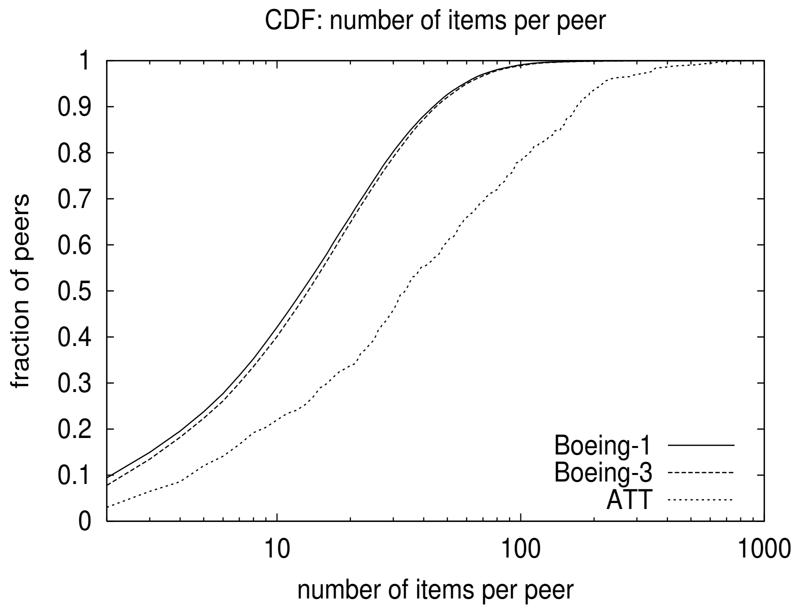

Figure 3: Cumulative count of fraction of peers which have at most x items. (ATT and Boeing logs), different days of the Boeing logs had almost identical curves, and thus were omitted.

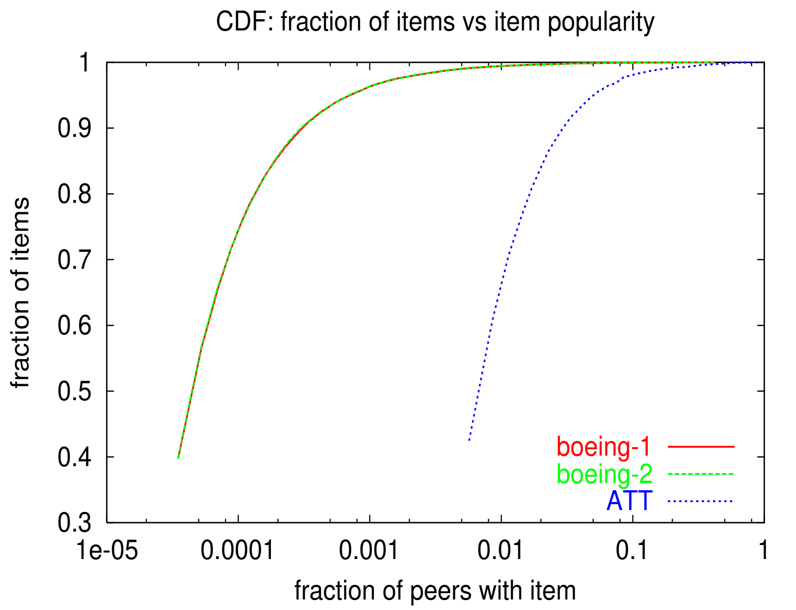

Figure 4: Cumulative count of items vs fraction of peers that have the item (ATT and Boeing logs), different days of the Boeing logs had almost identical curves, and thus were omitted. The difference between the Boeing and ATT curves stems from much fewer peers in the ATT log.

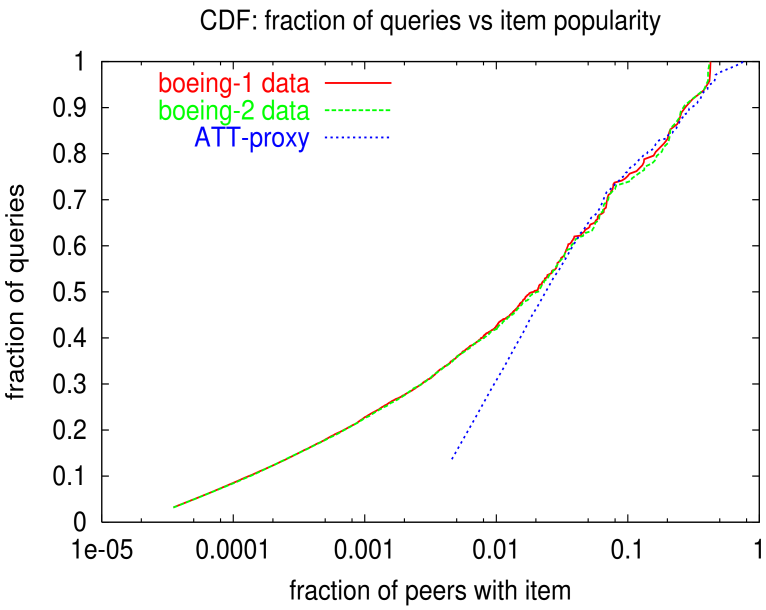

Figure 5: Cumulative count of queries vs fraction of peers that have the result (ATT and Boeing logs).

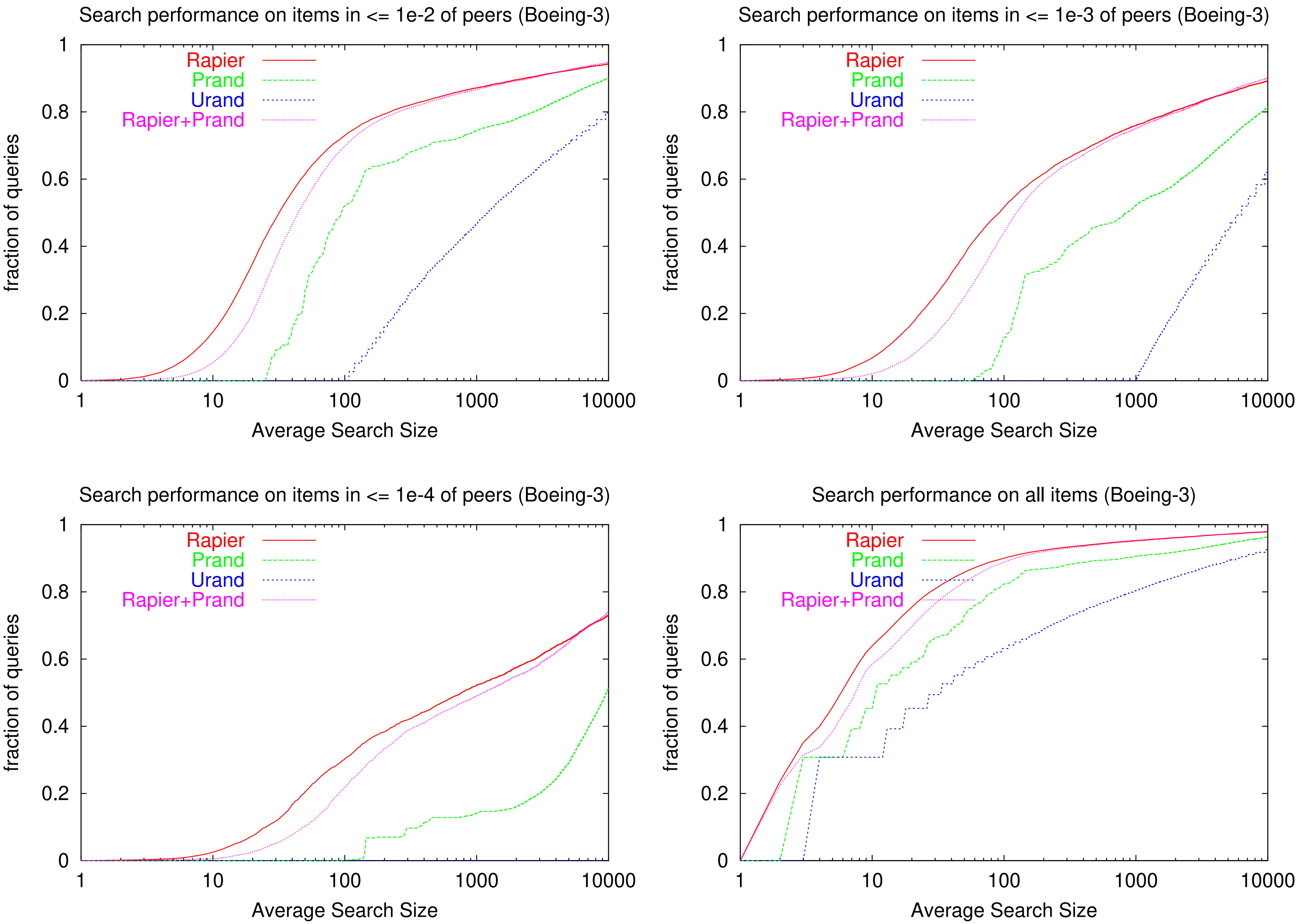

Figure 6: Search performance on items that are available from a fraction (1e-4, 1e-3, 1e-2, all) of peers (Boeing-Day3 log).

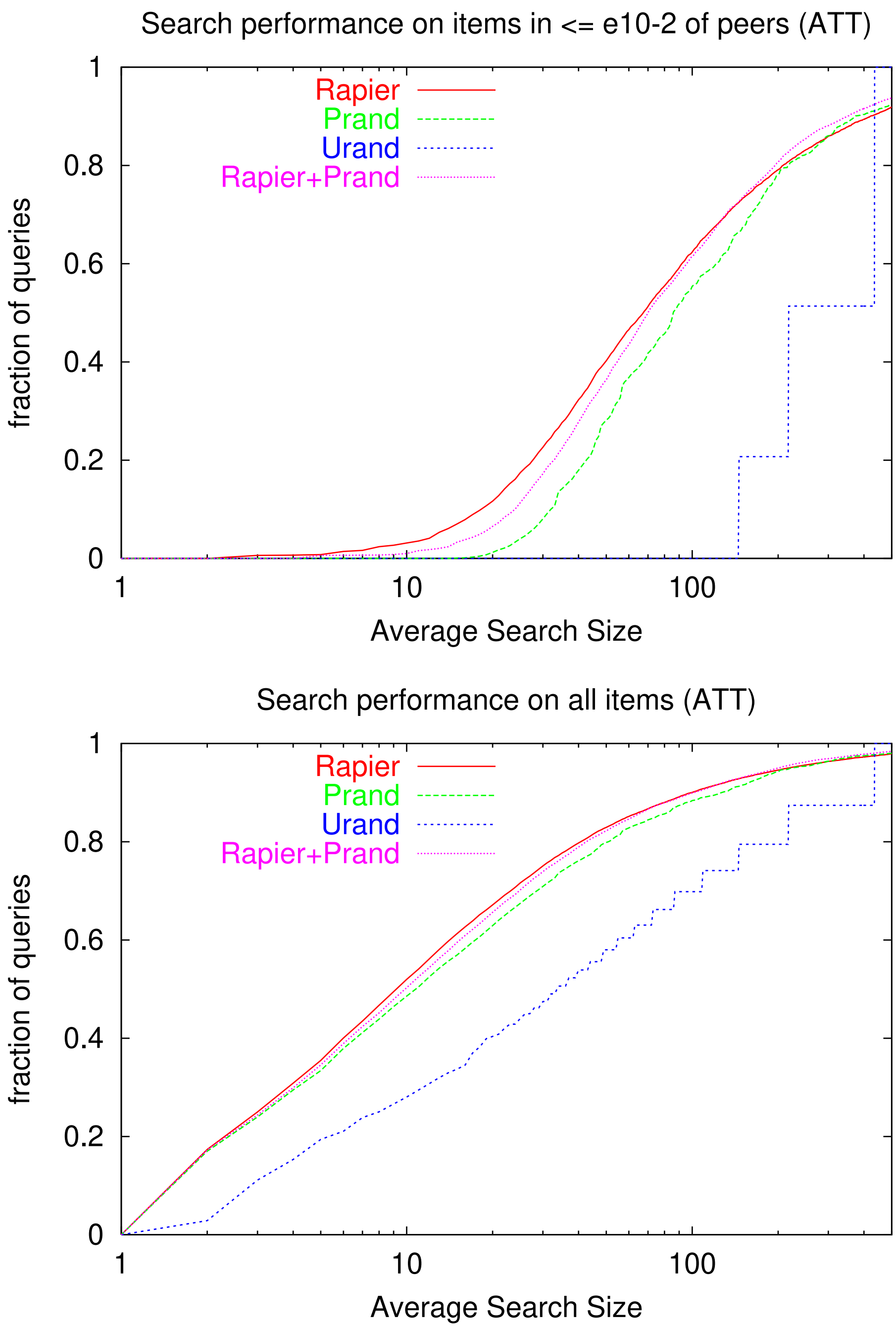

Figure 7: Search performance on items that are available from a fraction (10e-2, all) of peers (ATT log).

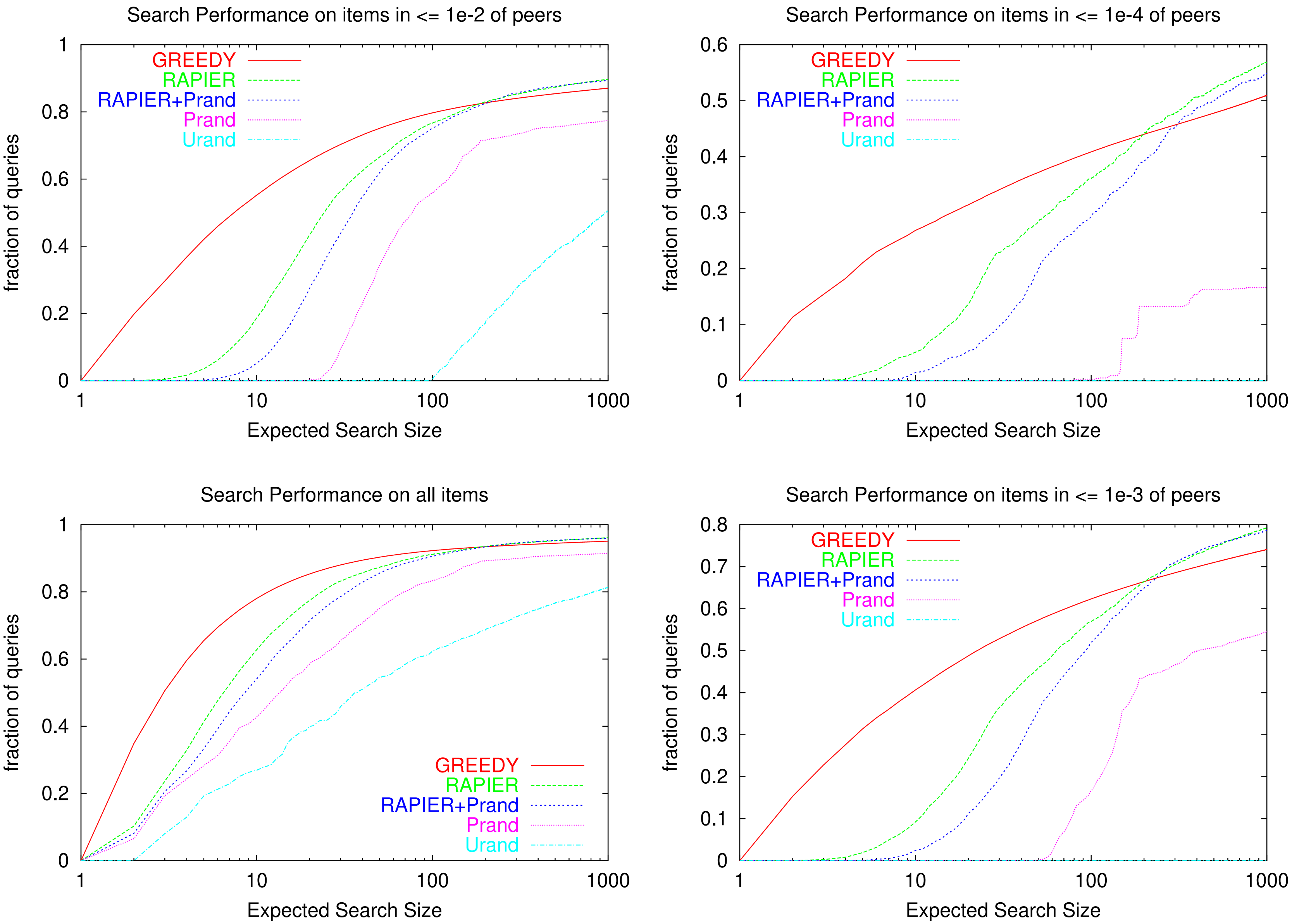

Figure 8: Search performance for a sample of peers with index sizes between 20-30, on items that are available from a fraction (1e-4, 1e-3, 1e-2, all) of all peers (Boeing-Day1 log).