Associative Search in Peer to Peer Networks: Harnessing Latent Semantics

Edith Cohen * Amos Fiat ** Haim Kaplan **

Association with other people corrupts our character.
— Friedrich Nietzsche (1844–1900), In *Thus Spoke Zarathustra*.

***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.

I. 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 [21], 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 [26]: unstructured, where search is *blind* (independent of the query or its context) and structured, where search is *routed*. The P2P architectures that spawned in Napster's void, eventually surpassing it in popularity, are *unstructured*. One of these architectures is Gnutella [13] 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 [26]. 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 [26]; 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 [32]-based P2P architectures (Morpheus, Kazaa [18], [17]) 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 dispensible; 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 [30], [27], [28], [12]), 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

* 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,haimk}@cs.tau.ac.il.

effective for applications where queries involve unique item identifiers (e.g., P2P Web caching), they require that peers store 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*. We argue that associative overlays retains 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. Another appealing feature of our design is that we do not require peers to maintain "arbitrary" data for the common good: the data stored and actions performed by each peer, including answering queries, have direct self benefit. *Associative overlays* integrate the formation of the overlay with the search process so that queries can be steered to peers that are more likely to have an answer. Our basic premise, which we substantiate in the sequel, 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.

A. Guide-rules overlays and Guided Search

A central ingredient in our design is *guided search*, which can be viewed as a middle ground between blind search used by unstructured networks and the routed search deployed by DHTs: Guided search provides a mechanism to focus the search, that is, the relevance of the peers that the query is propagated to, without tight control of the overlay and item locations.

A *guide rule* is a set of peers that satisfy some predicate. The set of peers belonging to some guide rule should contain data items that are semantically similar, *e.g.*, contain documents that deal with the philosophy of science, or contain song titles by the artist formerly known as Prince. The propagation of guided-search queries is restricted to guide-rules specified by the originating peer, and thus yields more effective search probes. P2P architectures differ in how the search is propagated. While previous architectures tailor the propagation only to the query, the premise of guide-rules overlays is to tailor the search to the originating peer, looking to exploit presumed correlations between the answer to the current query and previous requests that originated at the same peer.

The particular choice of the underlying 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 the 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.

The guide-rules dictate the overlay structure in the sense that each peer, for each guide rule it belongs to, maintains a small list of other peers belonging to the same guide rule. Forming and maintaining this overlay requires a simple mechanism for peers to join a rule and identify other peers that participate in the rule; our design integrates this mechanism with the search process. For each rule, the overlay induced by peers that participate in the rule "looks like" an unstructured network and exhibits similar connectivity and expansion properties. The search process within a rule mimics search in unstructured networks, by essentially performing a blind search. On the other hand, the search strategy of the originating peer has the flexibility of deciding which guide rules, among those that the originating peer belongs to, to use for a given search.

B. Possession rules

Guide rules can be defined on properties extracted automatically (e.g., previous selections, set of shared items, favorite genre's) or specified by the user. We focus on automatically-extracted guide rules of a very 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. Our underlying intuition, taken from extensive previous research in the Data-Mining and Text Retrieval communities (see, e.g.[16], [8], [7], [20], [15]), is that, on average, peers that share items (in particular rare items) are more likely to satisfy each other's queries than random peers. More precisely, search using possession-rules exploits presence of pairwise co-location associations between items.

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.

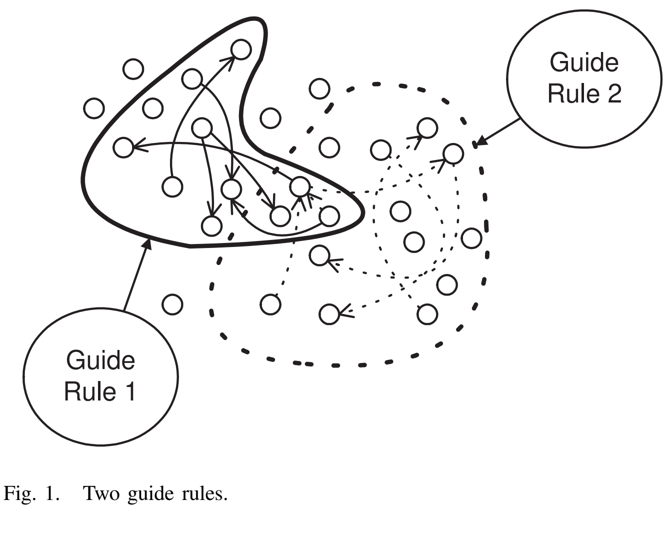


Fig. 1. Two guide rules.

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.

C. The RAPIER Algorithm.

Our first search algorithm, RAPIER, is based on the following intuition: let the areas of interest for a given peer be A, B, C, *etc.*, randomly choose one of these areas of interest and perform a blind search amongst those peers that also have interest in this area. RAPIER (Random Possession Rule) selects a possession-rule uniformly at random from the list of previously-requested items by the querying peer.

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.

D. The GAS Algorithm.

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. Thus, if we had a probability distribution on the query being issued, we could in principle find an optimal sequence of guide rules to use. Note that both what guide rules and the order in which they are probed is important.

We propose and evaluate a theoretically-grounded GAS strategy (Greedy Guide Rule). Each peer constructs its own GAS strategy by preferring to invoke rules that would have been more effective on its past selections. The GAS strategy (Greedy Guide Rule) approximates the strategy that would have performed best on all previous queries. GAS incurs higher overhead than RAPIER for building the search strategies, but our simulations indicate that GAS strategies can substantially improve over RAPIER search.

E. Structure of the rest of this paper.

In Section II we survey related work. Section III presents models metrics and methodology and discusses performance of blind search. Section IV develops the associative search framework, presents RAPIER, and compares it to blind-search using the Itemset model. Section V presents the GAS strategy. Section VI contains simulation results of the performance of different search strategies.

II. 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. [32]). Other proposed architectures use meta-data and query keywords for the routing: With routing-indices [14] 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 [22], peers remember all keywords of queries and responses passed through them and use this information for routing other queries. 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.[12]); particularly when the number of replicas of each item is careful balanced against its query rate [11], [26]. 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 (Web pages vs Web pages), 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 [16], [23], [5], [1]. Application of Latent Semantic Analysis on such matrices (Pioneered by [16] 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* [2], [3]. 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) [6], [19], [24], [4]. 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) [9].

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.

Other recent proposals that exploit "interest locality" to optimize p2p search include [29], where an existing p2p network is extended by nodes linking directly to nodes that satisfied previous queries; This basic approach does not provide a mechanism to "focus" query propagation beyond the first hop. At the other end of the spectrum, PeerSearch [31], attempts to import traditional vector space Information Retrieval (at the cost of tightly controlled DHT overlay and communication overhead).

III. 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 $S_j \subseteq \{1,\ldots,n\}$, $1 \le j \le m$, to be

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

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,

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

is the set of peers that contain both items. We refer to $X_i = \{j | D_{ij} = 1\}$ (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 $W_i = \frac{x_i}{|D|}$, where $|D| = \sum_{i=1}^{n} x_i$ is the combined size of all indexes. 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 refer to the request that corresponds to the i, j entry as the *query* (i, j). Each query triggers a search process, which depends on the matrix D with the entry D_{ij} set to 0 and on the peer i.[1] 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 of the form (i, j) can always (and only) be resolved by a probe to peer that contains the item j.[2] The size of a search process is a random variable, and the *Expected Search Size* $\textsc{Ess}_{ij}^{A}$ is the expectation of this random variable. 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 $\textsc{Ess}$, and look at the cumulative fraction of queries (i, j) that have $\textsc{Ess}_{ij}$ below a threshold.

A. Blind Search as Random Search

Following [11], [26] 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 on 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.

We present two associative search algorithms: RAPIER and GAS. 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 GAS 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 advantages shown experimentally are 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 consider weighted versions of the random search model where hosts have different likelihood of receiving a probe: Each peer i has a weight w_i such that $\sum_i w_i = 1$, and the likelihood that a peer is visited in a random search probe is proportional to w_i. Weighted random search is used as a benchmark for the performance of our associative search algorithms. To obtain a fair comparison, we need to consider weights that reflect the bias of the associative search algorithms towards peers with larger index sizes.

We shall consider both of the following natural weighting schemes:

- Uniform Random Search (URAND) where all peers are equally likely to be probed ($w_i = 1/n$). This models pure blind search.
- Proportional Random Search (PRAND), where the likelihood that a peer is probed is proportional to the size of its index $w_i = W_i \propto \sum_{j=1}^{m} D_{ij}$. This models blind search biased towards peers with larger indices. We will show that this bias is exactly equal to the bias introduced by RAPIER and thus differences in performance between the two cannot be due to this bias.

With weighted random search, the size of the search for a query (i, j) is a Geometric random variable. The $\textsc{Ess}$ is the mean of this random variable.

A weighted random search for item j by peer i has likelihood of success in each probe

$$p_{ij} = \frac{\sum_{k\neq i} w_k D_{kj}}{1 - w_i} \ .$$

and thus for any weighted random search algorithm A

$$\textsc{Ess}_{ij}^{A} = p_{ij}^{-1} = \frac{1 - w_i}{\sum_{k\neq i} w_k D_{kj}} \ .$$

[1]Note that the search sequence does not depend on j, as query properties (such as meta-data terms) are not used to determine where to search. It is used only as a stopping condition. See the introduction and conclusion sections for a discussion on this issue.

[2]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.

(The search is performed on all peers excluding peer i).

Thus, a URAND search for item j by peer i has

$$\mathrm{ESS}_{ij}^{\text{URAND}} = \frac{n-1}{\sum_{k \neq i} D_{kj}} \ ; \tag{1}$$

and a PRAND search has

$$\mathrm{ESS}_{ij}^{\text{PRAND}} = \frac{1-W_i}{\sum_{k \neq i} W_k D_{kj}} \ . \tag{2}$$

IV. ASSOCIATIVE SEARCH

We present the basic ingredients of our guide-rules overlay framework, and present the RAPIER strategy as a concrete instantiation of a search strategy on such overlays. A *guide rule* is a set of peers whose index satisfies some predicate. An example of a guide rule is *possession-rule* which tests the presence of a certain item in the index. Peers that participate in the same guide-rule form a sub-overlay that resembles a "traditional" unstructured network. The sub-overlays of different rules can be overlapping. Search is conducted through guide rules. Similarly to search in traditional unstructured networks, it is propagated from peer to neighbors but propagation is restricted to peers belonging to the selected guide rule. To facilitate search, it is sufficient that each guide-rule sub-overlay has similar properties to a traditional unstructured overlay network: for each guide-rule it is associated with, a peer needs to remember a small list of peers which belong to the guide rule; and neighbors should be such that guided-search reaches a large number of peers. 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 guide-rules). The specifics are orthogonal to our basic approach, we only need to make sure that our selected guide rules are such that the underlying unstructured network can form.

Search strategy: : A search strategy defines a search process as a sequence of guide rule 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 [11], [26].

Our general expectation is that the total number of guide rules may be large, but a typical peer uses a bounded number of rules. The applicability of a specific set of guide-rules depends on the implementability of the connectivity requirement. This requirement has two parts, first there should be a simple mechanism to locate a peer (and through it other peers) that belong to the same guide rule. It is also a requirement that this selection should result in large connected components. Below we argue that possession-rules fill the first part. As for large components, practice shows that simple neighbor selection strategies of current P2P implementation result in large connected components, and thus, we argue that selections within a guide-rule are likely to result in large components. (Random connections are known to yield large components and apparently actual selections are "sufficiently random" to obtain this property). In any case, the same issue of obtaining large components exists in traditional unstructured architectures and the connectivity algorithms deployed in these networks can be adapted to our context. There is thus no need to re-tackle this issue.

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. Our approach is to limit the number of guide-rules that peers with very large index can participate in. We also argue that we have no particular reason to severely limit 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 induced by the peers in a guide rule is a low degree network.

Possession-rule overlays are such that peers that index a certain item should maintain a list of other peers which index that item. This infrastructure is self-boosting: If peer-A conducts a search for item i that is resolved by peer-B then it is able to obtain through peer-B a list of other peers that index item i. As a result, each peer has a neighbor list which is an array of (item,peer) pairs for (most) items in its index. Thus, for possession rules, the construction of the overlay is symbiotic with the search process. When a disconnected peers wishes to rejoin rules, it can do so by iterative searches for items it has. At first, the search is blind, and thus can reveal only the rules for more popular items; subsequent iterations focus on rules found in previous iterations and thus reconnect the peer to rules of related rarer items.

In the sequel, we assume that our network is a possession-rule overlay. Each sub-overlay resembles an unstructured network and we use the model of random search used in [11], [26] to capture the performance of search within a rule.

A. RAPIER *strategy*

RAPIER is a simple search strategy that uses possession-rules overlay. 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 predetermined depth.

Random selection of 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.

The main parameter we look at is the size of the search which is the total number of peers probed. 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 $D_{rj} = 1$.

Thus, the likelihood of success for RAPIER per step is

$$p_{ij} = (x_i - 1)^{-1} \sum_{k \in X_i \setminus \{j\}} \frac{s_{kj} - 1}{n - 1} .$$

and thus

$$\text{Ess}_{ij}^{\text{RAPIER}} = \frac{(x_i - 1)(n - 1)}{\sum_{k \in X_i \setminus \{j\}} (s_{kj} - 1)} . \tag{3}$$

B. RAPIER *versus* PRAND

As discussed earlier, search strategies may differ to 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 share items with a larger number of other peers. The following Lemma relates the probing likelihood of each peer under RAPIER and under PRAND.

Lemma 4.1: Averaged over queries, the likelihood that a peer is probed under RAPIER is equal to W_i (its likelihood to be probed under PRAND).

Proof: The likelihood that a particular item j is selected as a guide rule is the sum, over all peers in $i \in S_j$, of the likelihood that i issued a query to an item other than j times the likelihood that j is selected by i. The likelihood that i issued a query to an item is W_i, the item is other than j with likelihood $(x_i - 1)/x_i$, and the likelihood that j is selected given the above is $1/(x_i - 1)$. Thus, the fraction of queries that use possession-rule j is

$$\begin{aligned} \sum_{i \in S_j} W_i((x_i - 1)/x_i)(1/(x_i - 1)) &= \sum_{i \in S_j} W_i/x_i \\ &= \sum_{i \in S_j} 1/|D| = s_j/|D| . \end{aligned}$$

The likelihood that a particular peer receives a search probe is the sum, over items in its index X_i, of the likelihood that the item constitutes a guide rule in a query not initiated by i times the likelihood that i receives the probe. The likelihood of a probe to item j not initiated by i is $s_j/|D| - W_i/x_i = (s_j - 1)/|D|$. The conditional probability that i receives the probe is $((s_j - 1)/|D|)/(s_j - 1) = 1/|D|$. The sum of this over all items in X_i is $x_i/|D| = W_i$. □

The Lemma suggests that it is fair to use PRAND as a benchmark for RAPIER since *per-search*, they have the same bias towards peers with larger index sizes. We compare the performance of the two algorithms on the Itemset model and using simulations.

C. 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 [3], [24], [25]. We use one such model, the *Itemsets model* (which resembles models in [3], [24]), to convey intuition why and when we anticipate RAPIER to perform well on real data.

The Itemsets model partitions items into N sets (which we refer to as *itemsets*). Each peer belongs to some subset of the itemsets, and contains f fraction of items (picked uniformly at random) in each itemset it belongs to.

Items in different itemsets are generally not correlated, and items in the same itemset are correlated. 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 a to a small number of itemsets we expect RAPIER to perform better; and we expect this advantage to increase as the number of itemsets decreases. We formalize this intuition below.

D. RAPIER *is no worse than* PRAND

We show that even when peers can belong to a large number of itemsets, 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. Note that this does not imply that the ESS is better per search or even that the average ESS per query is better.[3]

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: Let Y be the total number of (peer,itemset) pairs, and let y_i be the number of itemsets associated with peer i. Let B_j be the set of peers that have itemset j. Let $b_j = |B_j|$ and let $b_{k \cap j} = |B_k \cap B_j|$ be the number of peers containing both itemset k and itemset j. We denote by $z_j = \sum_{i \in B_j} y_i$, the sum of the "sizes" of peers that contain itemset j.

It is easy to see that for a query q, the success likelihood of a PRAND probe is $f \cdot z_q/Y$. We now consider the success likelihood of a RAPIER probe for item q. The likelihood that the selected guide-rule is for an item in itemset j is $b_{q \cap j}/z_q$. The conditional probability of success in this case is $f \cdot b_{q \cap j}/b_j$. Therefore the success likelihood of a RAPIER probe for item q is $\sum_j f \cdot b_{q \cap j}^2/(z_q b_j)$.

To complete the proof of the lemma we need to show that $\sum_j b_{q \cap j}^2/(z_q b_j) \geq z_q/Y$, and $\sum_j b_{q \cap j}^2/b_j \geq (z_q)^2/\sum_j b_j$.

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

$$\sum_j b_{q \cap j}^2/b_j \geq (\sum_j b_{j \cap q})^2/\sum_j (b_j) .$$

Equivalently,

$$\sum_j (b_j) \sum_j (b_{q \cap j}^2/b_j) \geq (\sum_j b_{q \cap j})^2 .$$

Cauchy's 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. □

[3] the average ESS is the average, over peers, of the inverse per-probe success probability which is not the same as the inverse of the average.

E. Bounded number of itemsets per peer

Suppose that each peer belongs to exactly k itemsets[4], and these itemsets are independent or positively correlated, that is, if $p(x)$ is the fraction of peers belonging to itemset x, and $p(x \cap y)$ is the fraction of peers belonging both to itemsets x and y, then $p(x \cap y) \geq p(x)p(y)$.

Lemma 4.3: Let D be a peers vs. items matrix generated according to this restricted itemset model. Let $x(\ell)$ be the itemset of item ℓ and let $p(x(\ell))$ be the fraction of the peers that contain itemset $x(\ell)$. Consider a query made to an item ℓ. Then the success probability of a PRAND probe is $R_\ell = fp(x(\ell))$; and the success probability of RAPIER probe is $C_\ell = \frac{f}{k} + \frac{k-1}{k}R(\ell) = \frac{f}{k}(1 + (k-1)p(x(\ell)))$.

It follows from Lemma 4.3 that the ratio of the ESS under PRAND to the ESS under RAPIER for item ℓ by any peer is $\frac{1}{kp(x(\ell))} + \frac{k-1}{k}$. Since $p(x(\ell)) \leq 1$, RAPIER is always at least as effective as PRAND in the restricted itemset model. When $p(x(\ell)) \ll 1/k$, 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 $p(x(\ell))$).

The difference between PRAND and RAPIER is further illustrated by considering itemsets that are equally popular, that is, $p(x) \equiv k/N$, where N is the number of itemsets. Substituting into the formula of Lemma 4.3 we obtain that the success probability of the probe picked by PRAND is fk/N and the success probability of a probe picked by RAPIER is $\frac{f}{k} + \frac{f(k-1)}{N}$. We note that the success likelihood of PRAND decreases to 0 with N whereas that of RAPIER is bounded by a constant independent of N. The performance ratio between the two algorithms is $n/k^2+(k-1)/k$ and thus RAPIER dominates when $k \ll \sqrt{N}$. Our intuition says that N 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.

V. GAS: OPTIMIZING THE SEARCH STRATEGY

As mentioned above, for a given set of guide rules, it is clear that RAPIER need not be optimal. In this section we describe how one can optimize the search strategy, the sequence of guide rules to be probed. We model the problem with a matrix $P \in R^{m \times m}$ with one row per guide rule and one column per data item, P_{jk} is the probability that for a uniformly chosen random peer ℓ belonging to guide rule j, contains item k ($D_{\ell k} = 1$). Note that P is not necessarily a stochastic matrix (neither rows nor columns necessarily sum to one). The matrix P will not be explicitly known, and instead, each peer deploying a GAS strategy will use an estimate of a submatrix with rows corresponding to guide-rules the peer participates in and columns to items in its index; the entries are estimated by deploying guided-searches. From here on the notation P will correspond to the submatrix of a particular peer.

Let $q =< q_1, q_2, \ldots, q_m >$ be a vector of probabilities that sum to one. We interpret q_i as the probability that the query is for item i and refer to q as the *query distribution*. In general, we will not know the query distribution. Given a query distribution, one can define the optimization problem of finding the best search strategy for the distribution.[5] This optimization problem is studied in [10], which show that it is *NP*-complete, but on the positive side, a simple GREEDY strategy gives a worst-case constant approximation ratio; that is, the average search time according to the sequence produced by GREEDY is within a small constant factor of the average search time that one can achieve with any other sequence.

The GAS strategy is obtained by applying GREEDY when P and q are selected in a certain way. Similarly to RAPIER, GAS is constructed by each peer and is based only on past selections by that peer. In our simulations, for each peer, we remove one item at a time from its index, build GAS based on the remaining index, and then apply GAS to search for the removed item.

Below we show how the GREEDY strategy can be constructed from P and $\mathbf{q}$, but before that we explain how GAS selects values for P and $\mathbf{q}$.

A. Choosing P and $\mathbf{q}$

Let X_i be the index of peer i. Let i_ℓ be the identity of the ℓth item in X_i. We explain how peer i, searching for item j, chooses $\mathbf{q}$ and P based on the items in $X_i \setminus \{j\}$. GAS assigns to the query vector $\mathbf{q}$ equal probabilities of $q_k = 1/(x_i - 1)$ for all items $k \in X_i \setminus \{j\}$ (all other entries are 0).

The set of guide-rules we use (rows of the matrix P) correspond to the items in $X_i \setminus \{j\}$. Let P_k be the guide-rule that corresponds to the kth item in $X_i \setminus \{j\}$.

For each pair of items i_k and i_ℓ in $X_i \setminus \{j\}$ ($i_k \neq i_\ell$) we have $P_{k\ell} = (s_{i_k i_\ell} - 1)/(s_{i_k} - 1)$, which is simply the success likelihood of probing for i_ℓ according to possession-rule of the item i_k.

A heuristic that requires more explanation is that GAS sets the "diagonal" entries to $P_{kk} = 0$ (instead to 1). This heuristic is intended to circumvents over fitting: GAS 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 size of the index that is not likely to perform well on the "removed" item i_j. Setting the diagonal to 0 is similar to valuing each guide-rule P_k according to how well it performs on items *other than* i_k.

The construction of a GAS strategy is more involved than that of RAPIER strategy. The underlying set of guide-rules is the same (possession-rules), and thus both are supported by the same overlay; but the difference is that in order to build a GAS strategy, the peer needs to collect information on how well (a sample or all) of its guide-rule perform on a sample or all) items in its index. This information can be collected by conducting guided-searches to estimate $P_{k\ell} = (s_{i_k i_\ell} - 1)/(s_{i_k} - 1)$; that is, determine how effective is each guide-rule in resolving each item. The accuracy of the

[4]Similar results would hold when we assume that each peer belongs to at most k itemsets

[5]We consider the problem for general query distributions. In the GAS strategy, we start with a query distribution that is "uniform" over previous "examples," but this generality is crucial because we will be modifying the query probability distribution as time goes by and unsuccessful searches have been performed. The posteriori probabilities can be anything.

estimate depends on the size of the search, but fortunately, the analysis in [10] 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 GAS strategy are likely to be more accurate than later steps; and specifically this means that GAS 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 GAS aagainst other search algorithms. GAS then applies GREEDY strategy with $\mathbf{q}$ and P, and obtain a search sequence of possession-rules.

B. *The* GREEDY *search sequence*

We next explain how GREEDY computes the search strategy from P and q. Let P_j be the j'th row of the matrix P above, $P_j = < P_{j1}, P_{j2}, \ldots, P_{jm} >$. We also denote by (v, u) the inner product of v and u. Let $\bar{1} \in R^m$ denote the all-one vector.

The search strategy is a multi-sequence of guide-rules (each guide-rule can appear multiple times.) The GAS 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 the guide-rule j:

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

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

Set $q \leftarrow q'$.

C. *Metrics*

We compare different search strategies by a curve which shows the expected fraction of queries that can be answered using a search of a certain size. The calculation we use for GAS assumes that q and P are obtained as described above and that the respective sequence of guide-rules $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_{|\sigma|}$ is produced using the GREEDY algorithm. For an item i_j and peer i we calculate $E_k(i, i_j)$ which is the success likelihood of resolving the query (i, i_j) using the sequence $\sigma_1, \sigma_2, \ldots, \sigma_k$.

Formally,

$$E_k(i, i_j) = 1 - \Pi_{\ell=1}^{k}\left(1 - \frac{s_{i_{\sigma_\ell}, i_j} - 1}{s_{i_{\sigma_\ell}} - 1}\right) .$$

is the success likelihood of the following process:

1) Set Success $\Leftarrow$ False, $\ell \Leftarrow 0$.
2) While Success $\neq$ True and $\ell < k$:
   a) $\ell \Leftarrow \ell + 1$.
   b) Probe the possession-rule i_{σ_ℓ} for i_j. If successful set Success $\Leftarrow$ True.

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 I

DESCRIPTION OF THE DATA

VI. DATA AND SIMULATION RESULTS

The experimental evaluation requires 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 [33]; both sets of servers constitute lower-level proxies serving end users. From each log we extracted the matrix of users vs hostnames [6] 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 we use each day separately. The resulting data matrices are described in Table I. 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 2) and the support-size of items (see Figure 3), with large fraction of peers having small index sizes and large fraction of items being present at a small fraction of peers. Figure 4 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.

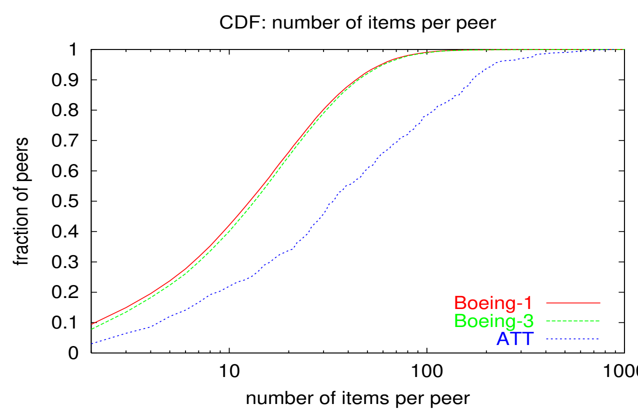


Fig. 2. Cumulative count of fraction of peers which have at most x items. (ATT and Boeing logs), different days of the Boeing logs had almost identical curves, and thus were omitted.

We 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 Equation 3.
- 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

[6] 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.

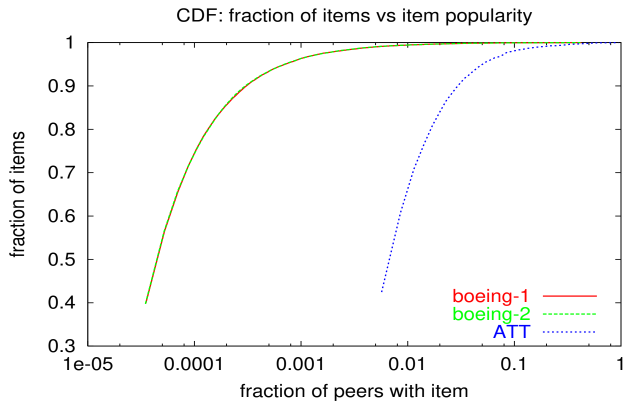


Fig. 3. Cumulative count of items vs fraction of peers that have the item (ATT and Boeing logs). The difference between the Boeing and ATT curves stems from much fewer peers in the ATT log.

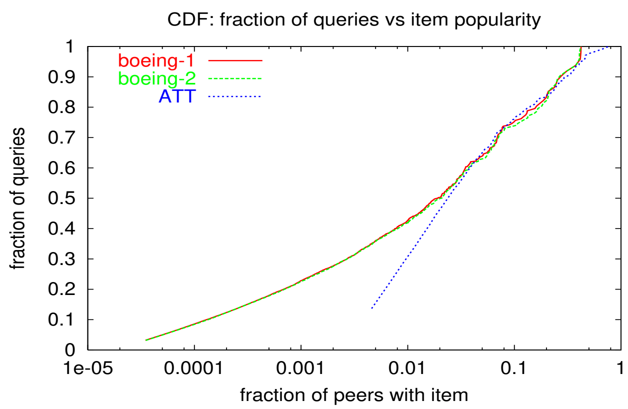


Fig. 4. Cumulative count of queries vs fraction of peers that have the result (ATT and Boeing logs).

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 Figures 5 (Boeing day-3 - results for other days were very similar) and Figure 6 (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 its 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 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 the same visit rate to each peer and thus the same average index size seen per probe). 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; and also emphasizes the qualitative difference between RAPIER and aggregation-based search strategies.

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 on 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 GAS against the 4 other search strategies. Our evaluation of GAS required us 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. 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 7, however, show similar performance gaps to what we observed earlier for the 4 strategies across all peers. The figures also show that GAS 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 GAS 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 V, GAS is considerably more involved than RAPIER, but the results suggest that most of the benefit of GAS is reaped on the first few probes (small search size). This suggests that a hybrid strategy that first applies GAS with a small-size search then (if needed) continues with RAPIER with a larger-size search can be effective.

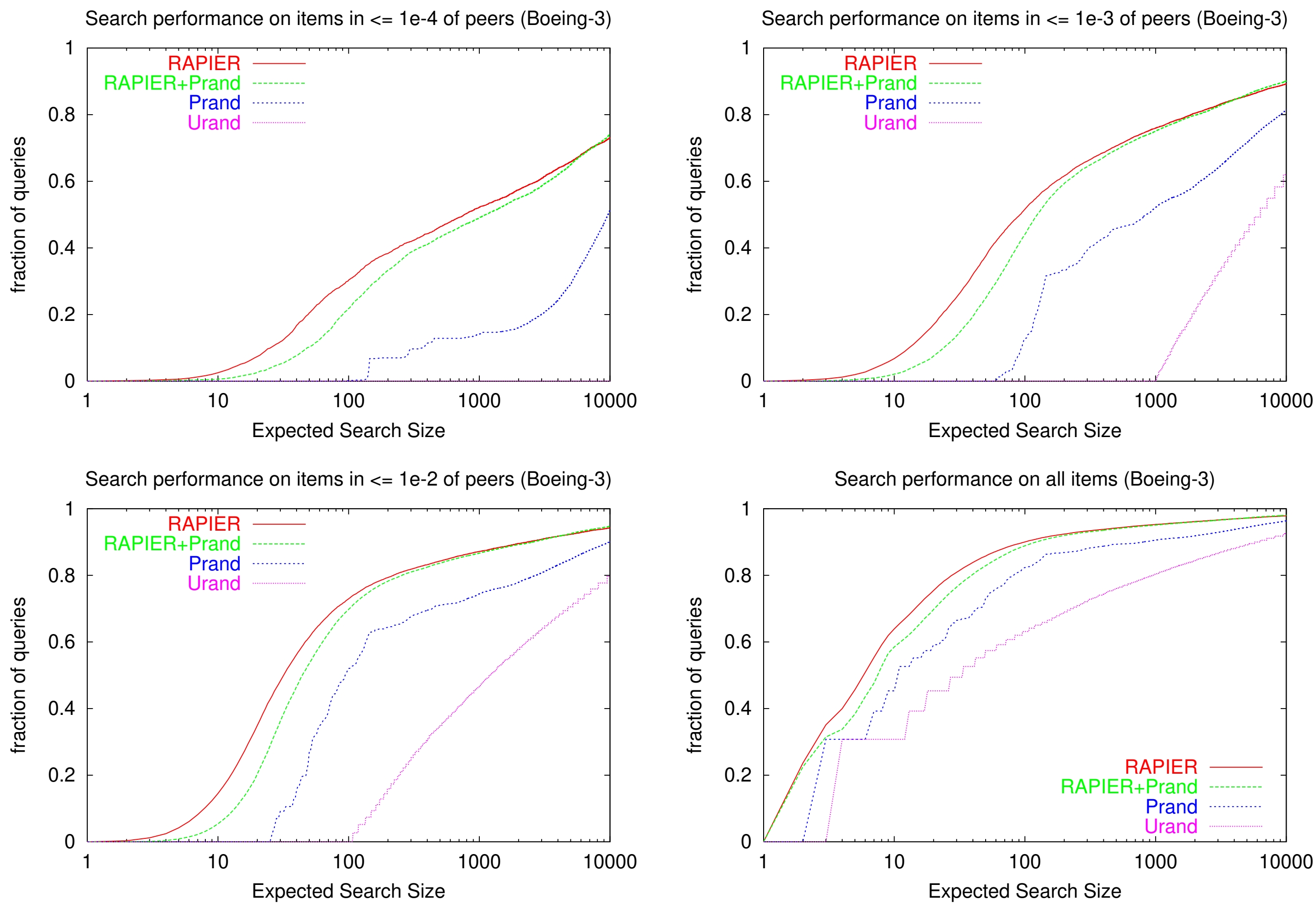


Fig. 5. Search performance on items that are available from a fraction (1e-4, 1e-3, 1e-2, all) of peers (Boeing-Day3 log).

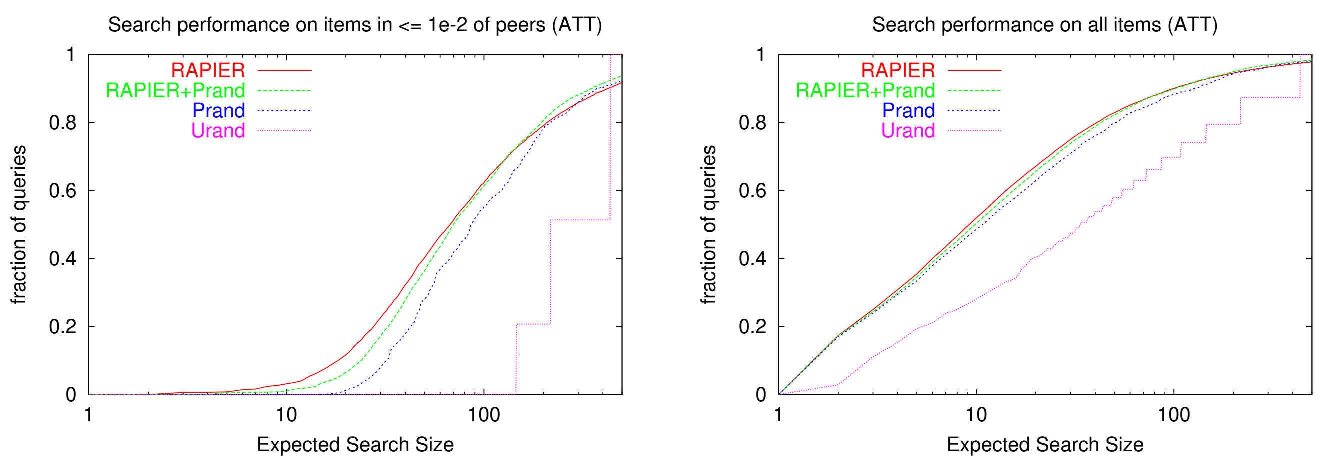
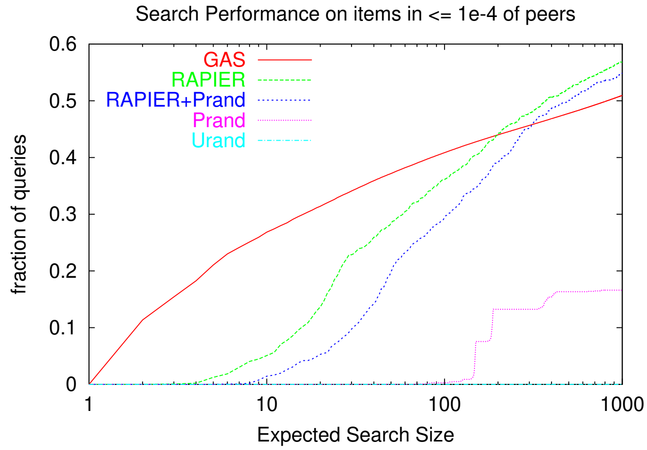


Fig. 6. Search performance on items that are available from a fraction (10e-2, all) of peers (ATT log).

VII. 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 dramatically increase the effectiveness of search for rare items over that of plain unstructured networks. It is likely that better search performance on



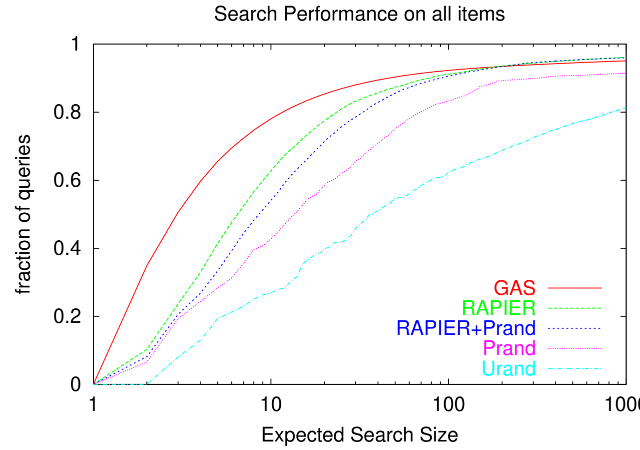


Fig. 7. Search performance for a sample of peers with index sizes between 20-30, on items that are available from a fraction (1e-4, all) of all peers (Boeing-Day1 log).

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

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.

[9] 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.

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

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

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

[13] Open Source Community. Gnutella. In *http://gnutella.wego.com/*, 2001.

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

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

[16] 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.

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

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

[19] 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.

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

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

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

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

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

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

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

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