

Supporting Approximate Similarity Queries with Quality Guarantees in P2P Systems

Qi Zhong[†], Iosif Lazaridis[‡], Mayur Deshpande[‡], Chen Li[‡], Sharad Mehrotra[‡], Hal Stern[‡]

[†] Microsoft Corporation, [‡] University of California, Irvine
qi.zhong@gmail.com, {iosif, mayur, chenli, sharad}@ics.uci.edu, sternh@uci.edu

Abstract

In this paper we study how to support similarity queries in peer-to-peer (P2P) systems. Such queries ask for the most relevant objects in a P2P network, where the relevance is based on a predefined similarity function; the user is interested in obtaining objects with the highest relevance. Retrieving all objects and computing the exact answer over a large-scale network is impractical. We propose a novel approximate answering framework which computes an answer by visiting only a subset of network peers. Users are presented with progressively refined answers consisting of the best objects seen so far, together with continuously improving quality guarantees providing feedback about the progress of the search. We develop statistical techniques to determine quality guarantees in this framework. We propose mechanisms to incorporate quality estimators into the search process. Our work makes it possible to implement similarity search as a new method of accessing data from a P2P network, and shows how this can be achieved efficiently.

1 Introduction

Peer-to-peer (P2P) systems have emerged as a powerful and popular alternative to traditional centralized system architectures. These systems provide many advantages: scalability, resilience to failures, self-organization, and the ability to harness remote resources. For instance, in a file-sharing P2P system, a user provides keywords to search for relevant files in the network. As an example, if a user provides a query with keywords {Beatles, Comes, Sun}, then the search process will return files in the network that are relevant to these keywords, presumably files of the song *Here Comes the Sun* by the Beatles.

We envisage a more powerful search model for P2P systems where queries are no longer simply sets of keywords, but can consist of similarity predicates defined over attributes of objects stored in peers. Consider,

for instance, a network of peers storing digital content, such as images, photographs, music, or electroencephalographic (EEG) databases in the case of a network of hospitals. Similarity retrieval over P2P networks in the new model will allow users to search not just on keywords, but also on matches based on features extracted from media, such as color, texture, time series properties, etc. Users can submit queries such as “Find images similar to a given image” and “find songs similar to *Here Comes the Sun*.” Such a search paradigm could also enable P2P-based shopping/trading applications, in which sellers of used cars, homes, etc. can be organized into a network of peers. Buyers can issue queries by specifying an approximate price, an item description, and/or other conditions to ask for the best matches, e.g., “find used cars with a price around \$6000, color close to green, and year of manufacture around 1999.” Such a generalized search model can significantly enhance current usage of P2P systems, and provide opportunities to extend benefits of P2P computing to new application domains.

A naive approach to answering a similarity query in a P2P system is to propagate the query to the entire network, collect the best answers from each peer, and merge the results at the querying node. This exhaustive search approach is, however, prohibitively expensive in a large-scale P2P network with a large amount data. For example, the Morpheus network had 470,000 users sharing a total of 0.36 petabytes of data as of October 2001 [22]. Ranking all the resources or accessing all the peers for a similarity query is virtually impossible. On the other hand, if we limit the search to a subset of peers in the network, we cannot guarantee that the best objects found so far are really the best ones in the entire network, since there could still be better objects in peers that have not been examined.

In this paper, we propose a novel framework to support approximate answering of similarity queries in P2P networks. When a query is posed, the search over the network commences, and the best objects seen so far are continuously presented to the user. These are accompanied with quality estimates which improve progressively, as more peers are searched. The user can decide to terminate the search at any time, if she finds some objects

of interest, and is satisfied with the quality of the current answers.

The rationale behind this framework is the following. First, computing the *exact* best answers to a similarity query requires accessing *all* peers in the network. Even if one peer is not accessed in the search, there could be objects in it that are more relevant than all the seen ones. Such an exhaustive search is, unfortunately, impractical for large-scale P2P networks. Second, given the inherent fuzziness of similarity search, users submitting similarity queries are often satisfied with “good enough” answers based on a subset of objects in the network. As an analogy, when a customer shops for used cars, after seeing a few vehicles at several car dealers, even though there could be many “better” cars at other dealers not seen by the shopper, she can still stop shopping and choose the best one from the set of cars she has seen, provided that she believes they are not “too far” from the really best ones. Our techniques make it possible for users to form such a belief in the process of answering a similarity query in P2P network. Third, after a query is submitted, it is important to inform the user about the progress of the search in terms of the best objects seen so far and their estimated quality. This will allow her to monitor the search process and choose when to stop it, as she can choose among the currently best objects at any time, and has a clear and continuous confirmation that the system is actively trying to improve the answer.

The realization of this framework poses several challenges: How is the quality of objects quantified? What guarantees can be given about the quality of objects drawn randomly from all the objects in the system? What is the effect of the fact that peers contain “clustered” sets of objects similar to each other, rather than a random sample of all the objects in the network? Finally, how can an efficient search be implemented on top of a P2P system to produce an approximate answer of good quality? In our paper, we address these challenges, making the following specific contributions:

1. We propose a new paradigm to support similarity queries in P2P networks (Section 3). After the query is submitted, the best objects seen so far are presented, along and quality guarantees which improve progressively as more peers are searched.
2. We develop techniques to estimate the quality of objects in both the case of having a random sample of objects and the case where objects are related within peers (Section 4). Our techniques are applicable in various P2P networks.
3. We show how P2P search can incorporate the proposed quality estimation methods (Section 5).
4. We demonstrate empirically that regardless of the network size, a small number of peers suffices to provide the user with answers of good quality.

2 Related Work

Similarity search has been extensively studied in the literature of information retrieval [2], data management [4], and multimedia systems [6, 18]. Many similarity search systems are based on a centralized computing model [6, 18, 19]. Recent studies on supporting similarity queries in distributed environments are [11, 17]. Papadopoulos and Manolopoulos [17] study how to answer nearest neighbor queries on multidimensional databases in distributed environments. They propose query evaluation strategies that access all the peers. King et al. [11] propose a system called DISCOVIR to support content-based visual information retrieval in P2P networks. The authors propose a “firework” query model to limit similarity search within a subset of peers. Their algorithms do not provide theoretical guarantees on the quality of an answer.

There are studies on how to provide results with progressively improving quality in the process of answering a query [7, 15]. For instance, [7] studied how to answer aggregation queries progressively, using random sampling to provide quality guarantees. Our work differs in terms of the type of query examined (similarity queries), and setting (P2P systems vs. traditional databases). [3, 16] study progress indicators for time-consuming database queries, so that users can receive feedback pertaining to the execution progress and the time remaining to completion. In a P2P system, time to completion is less meaningful, as the query usually finishes when the user interrupts it; our work keeps the user informed on the query’s progress by providing a continuously improving guarantee about the gradually refined answer.

Manku et al. [?] have looked at the problem of computing quantiles for random sampling techniques, utilizing Hoeffding bounds for this purpose; they are thus able to compute bounds without a priori knowledge of the size of the dataset. Our work differs from theirs in that (i) we show how estimation can be achieved if data is partitioned in several sets of objects distributed among peers, (ii) we develop techniques for searching in the P2P network, since objects are not assumed to all reside in a single place, and (iii) we show how to progressively improve bounds by exploring the network and sampling more objects.

3 Formal Setting

We now present our generic framework for supporting online similarity queries with quality guarantees.

3.1 Similarity Queries

Consider a P2P network, in which each peer contains a set of objects. An object could be a relational tuple, an image, an audio file, or any other type of sharable content. Let \mathcal{B} be the bag of all objects in the network.

A *similarity query* in the P2P network is a triplet (f, q, k) , where k is an integer, q is a query point and

f is a similarity ranking function (related to this query) that takes an object o and returns a score $f(q, o)$ as its relevance to the query point q . The answer to the query are the k best distinct objects in the bag \mathcal{B} according to the function f . That is, its answer is a set of distinct objects $\mathcal{A} \subseteq \mathcal{B}$, such that $|\mathcal{A}| = k$ and $o \in \mathcal{B}, o \notin \mathcal{A} : f(q, o) > \min_{p \in \mathcal{A}} f(q, p)$. We assume that each peer can evaluate the f function over its local objects. Notice that computing the answer to the query requires accessing all the peers in the network, since unvisited peers can always have better objects. Since it is impractical to visit all the peers, it becomes interesting to answer such a query approximately based on the objects in a subset of peers.

3.2 System Framework

We present a system framework that can answer a similarity query approximately and progressively with quality guarantees. In the framework, a user poses a similarity query (f, q, k) on his own peer, called the *root* of the query. After the search process starts, the system computes an approximate answer to the query based on the objects seen so far in the search.

In order to answer a query approximately, our system has two modules running simultaneously: a *search module* and a *quality estimation module*. The search module propagates the query to other peers using a sampling method and retrieves the best k distinct objects from those peers. The sampling method adds each visited peer to a set \mathcal{V} , and produces an approximate answer. We study how to implement this module in Section 5. The quality estimation module, whose implementation is described in Section 4, generates a quality estimate for each retrieved object in the answer. Our framework continuously presents the current best k objects to the user along with their quality estimates, measured with the well-known statistical concept of a “quantile”:

Definition 1 (Quantile) Consider a query (f, q, k) with a ranking function f . The quantile of an object o w.r.t. the bag \mathcal{B} of all the objects in the network, denoted as $R(o, \mathcal{B})$, is the maximum value ϕ such that the score $f(q, o)$ is greater than or equal to exactly $\lceil \phi \cdot |\mathcal{B}| \rceil$ elements in the bag.

For example, let the bag $\mathcal{B} = \{o_1, o_2, o_3, o_4\}$. Given a ranking function f , assume $f(o_1) < f(o_2) = f(o_3) < f(o_4)$. The quantile of o_4 is 1.0, since its score is greater than or equal to all the objects (including itself). Similarly, the quantiles of o_3, o_2 , and o_1 are 0.75, 0.75, and 0.25, respectively. Intuitively, the quantile $R(o, N)$ of an object o represents the relative position of this object among all the objects in terms of their scores. Thus it can be used as a good indicator of the goodness of the object.

The quantile of an object is defined on the bag \mathcal{B} of all objects in the network. Assume we have accessed a sub-bag $\mathcal{B}' \subset \mathcal{B}$ of objects in the network and computed the

best k objects in \mathcal{B}' . We need to estimate the quantile of these k objects in terms of where they stand among all the objects \mathcal{B} , even though we do not have the full knowledge about \mathcal{B} . To solve the problem, we use a probabilistic model to estimate quality.

Definition 2 (Quality Estimate) Given a query (f, k) , the quality estimate of a seen object o in the search is a pair (ϕ, p) , in which ϕ is a quantile and p is a probability. It means that, with at least probability p , the quantile of object o amongst all the objects in \mathcal{B} is at least ϕ .

We discuss the computation of quality estimates from a bag of samples of \mathcal{B} in Section 4. One important feature of such a measurement is that, as the search continues and more peers are explored, the quality estimates of the current best k objects keep improving. As a result, the user can stop the search when she is satisfied with the estimated quality for the best objects seen so far. In our framework, a user can either explicitly specify a quality threshold (τ_ϕ, τ_p) and let the search terminate automatically, or keep that information in mind and stop the search as soon as he observes that the threshold has been reached.

Figure 1 shows the user interface, for similarity queries over image files. The best 3 objects are displayed in a ranked order. The interface also shows additional information about each object, such as its ranking score and a confidence interval. For instance, the first image has the highest score of 0.99. The system has a 95% level of confidence that this image is better than 99.2% of the objects in the system.




Search Results				
Rank	Result	Score	Interval	Confidence
1		0.99	Top 99.2%	95%
2		0.97	Top 99.0%	95%
3		0.95	Top 98.5%	95%

Figure 1: Query Interface

Our framework can also be used to alleviate the cost of exact search. We can first estimate a similarity threshold using the sampling techniques proposed in this paper and broadcast this threshold and prune the network while performing an exact search. Since our interest is in approximate search, we don't discuss this extension further.

4 Quality Estimation Using Quantiles

We now show how to compute quality estimates (ϕ, p) of an object o in bag \mathcal{B} , given a bag objects $\mathcal{B}' \subset \mathcal{B}$.

4.1 Quality Estimation Under Random Sampling

We first study how to compute the (ϕ, p) pair, assuming objects in \mathcal{B}' are random samples from the entire population \mathcal{B} , based on the following theorem:

Theorem 1 (Hoeffding’s Tail Inequality) [9] *Let X_1, X_2, \dots, X_n be independent bounded random variables such that X_i is within the interval $[a_i, b_i]$ with probability 1. Let $S_n = \sum_{i=1}^n X_i$. Let $E(S_n)$ be the expectation of S_n . For any $t > 0$, we have:*

$$\Pr\{S_n - E(S_n) \geq t\} \leq e^{-2t^2 / \sum_{i=1}^n (b_i - a_i)^2},$$

where “Pr” stands for “probability” or “confidence.”

From Theorem 1, we have the following corollary.

Corollary 2 *Let X_1, X_2, \dots, X_n be independent random variables with $0 \leq X_i \leq 1$ for $i = 1, 2, \dots, n$. Let $S_n = \sum_{i=1}^n X_i$. Let $E(S_n)$ be the expectation of S_n . Then, for any $t > 0$, we have*

$$\Pr[S_n - E(S_n) \geq t] \leq e^{-\frac{2t^2}{n}}.$$

The following lemma helps us decide how to estimate the quality of top- k objects from a sample.

Lemma 3 *Let $0 \leq \delta, \epsilon \leq 1$ be two values. A total of*

$$M > \frac{\log(\delta^{-1})}{2\epsilon^2}$$

random samples from a population are enough to guarantee the following. A ϕ -quantile of these M samples is greater than the $(\phi - \epsilon)$ -quantile of the population with probability at least $1 - \delta$.

Let S_ϕ be the score of a ϕ -quantile element in the M samples. Let $N_{\phi-\epsilon}$ be the score of a $(\phi - \epsilon)$ -quantile element in the entire population. By Definition 1, S_ϕ is the $\lceil \phi \cdot M \rceil$ -th smallest element in the samples S . So if there are more than $\lceil \phi \cdot M \rceil$ elements with a score smaller than $N_{\phi-\epsilon}$, then S_ϕ will become smaller than $N_{\phi-\epsilon}$. In other words, the property mentioned in the lemma does not hold if and only if more than $\lceil \phi \cdot M \rceil$ elements are chosen from the population whose score is no greater than $N_{\phi-\epsilon}$. The probability to draw such an element is $\phi - \epsilon$.

If we define the event of drawing an element with a score no greater than $N_{\phi-\epsilon}$ in the population as a “success,” then the process of drawing M samples from the entire population can be viewed as M independent Bernoulli trials with probability $\phi - \epsilon$. We use a sequence of numbers X_1, X_2, \dots, X_M to present the result of such a sequence of trials. $X_i = 1$ means that the i^{th} trial is a success, and $X_i = 0$ otherwise. Thus $S_n = \sum_{i=1}^n X_i$ represents the total number of successful trials, i.e., the total number of elements whose score is no greater than

$N_{\phi-\epsilon}$. Since the expected number of successful trials $E(S_n) = (\phi - \epsilon) \cdot M$, we can get the following inequality based on Corollary 2:

$$\Pr[S_n \geq \lceil \phi M \rceil] \leq \Pr[S_n \geq \phi M] \quad (1)$$

$$= \Pr[S_n - E(S_n) \geq \epsilon M] \quad (2)$$

$$\leq e^{-2\epsilon^2 M} \quad (3)$$

$$\Pr[S_\phi \geq N_{\phi-\epsilon}] = 1 - \Pr[S_n \geq \lceil \phi M \rceil] \quad (4)$$

$$\geq 1 - e^{-2\epsilon^2 M} \quad (5)$$

The last inequality implies that, as long as we have:

$$1 - e^{-2\epsilon^2 M} = 1 - \delta \quad (6)$$

the probability that $S_\phi \geq N_{\phi-\epsilon}$ is at least $(1 - \delta)$. Based on the above equation, we have

$$M = \frac{\log(\delta^{-1})}{2\epsilon^2} \quad (7)$$

Therefore, in order to achieve the property in the lemma, the number of samples needed is $\frac{\log(\delta^{-1})}{2\epsilon^2}$. Also, we can get the following from Equation (6):

$$\delta = e^{-2\epsilon^2 M} \quad (8)$$

$$\epsilon = \sqrt{\frac{\log(\delta^{-1})}{2M}} \quad (9)$$

Lemma 3 tells us that if the quantile of object o is ϕ in the M samples, then with probability at least $1 - \delta$ the quantile of o is $\phi - \epsilon$ in all objects in the network, where δ and ϵ are computed from Equations (8) and (9). In other words, we can estimate the quality of o to be $\{\phi - \epsilon, 1 - \delta\}$. From Equation (8), we know that, if ϵ is fixed, the confidence $(1 - \delta)$ becomes higher as more objects are sampled. From Equation (9), we know that, if δ is fixed, the quantile $(\phi - \epsilon)$ is getting higher as more objects are sampled.

The following lemma suggests how to estimate the quality of an approximate answer to a similarity query.

Lemma 4 *Let $\mathcal{B}' = \{o_i\}$ be a bag of random samples from a population \mathcal{B} . Let ϕ_i be the quantile of object o_i in \mathcal{B}' . Let $\tau_p = (1 - \delta)$ be a probability. Then the quality for o_i in \mathcal{B} , is $(\phi_i - \epsilon, \tau_p)$, where $\epsilon = \sqrt{\frac{\log(\delta^{-1})}{2|\mathcal{B}'|}}$.*

We use an example to show how to use Lemma 4 to estimate the quality of approximate results. Suppose that during the search to answer a similarity query with $k = 2$, we have seen a set \mathcal{B}' of 100 random objects from the population \mathcal{B} so far. The best two distinct objects in \mathcal{B}' are $\{o_a, o_b\}$, where $f(o_a) > f(o_b)$. Assume the predefined confidence is $\tau_p = 1 - \delta = 0.95$, i.e., $\delta = 0.05$. Based on Equation (9), we have $\epsilon = 0.12$. Since the quantile of object o_a in \mathcal{B}' is 1.0, we claim that its quality

is (0.88, 95%) in \mathcal{B} , i.e., with a probability at least 95%, object o_a is better than 88% of the objects in the entire population. Similarly, we can show that the quality for o_b is (0.87, 95%).

There are two variables ϕ and p in our quality estimate. During a search, we generally fix one variable by setting $\phi = \tau_\phi$ or $p = \tau_p$. Figure 2 shows a procedure that takes samples and outputs the current top- k objects with their quality estimate.

```

function: EstimateQuality
Input: bag of samples( $\mathcal{B}'$ ), random sample
size( $S$ ), confidence( $\tau_p$ ), int( $k$ )
Output: best  $k$  distinct objects( $\{o_i\}$ ),
quality (  $\{(\phi_i, \tau_p)\}$  ) ( $1 \leq i \leq k$ )
1. extract best  $k$  distinct objects  $\{o_i\}$  from  $\mathcal{B}'$ ;
2. compute the quantile  $\phi_i^*$  of  $o_i$  w.r.t.  $\mathcal{B}'$ ;
3.  $\delta = 1 - \tau_p$ ;
4. for  $i = 1$  to  $k$ 
5.    $\phi_i = \phi_i^* - \sqrt{\frac{\log(\delta^{-1})}{2S}}$ ;
6. endFor
7. return  $\{o_i\}$  and  $\{(\phi_i, \tau_p)\}$ ;

```

Figure 2: Quality Estimate

4.2 Estimating Quality with Related Objects

If we sample a subset of peers from the entire network in a random manner, then we have access to all the objects in each of these peers. However we may *not* be able to treat these as random samples from the entire population, because objects within each peer could be related, because, e.g., users may be interested in content of a certain type.

This presents a problem, because the quality estimation approach of Section 4.1 requires a random sample of objects. To overcome this, we might try a naive approach, picking a single object from each sampled peer and repeating this step until enough objects are accumulated to attain the desired quality. This method, though statistically valid, is impractical. We employ a more workable approach, using statistical methods developed in the context of *clustering sampling* to determine an “effective” random sample size for each peer. The intuition is to analyze the variability of object scores within peers (denote this variance σ_w^2) and the variability of object scores between peers (denote this variance σ_b^2). By comparing σ_w^2 and σ_b^2 , we have a measure of the “relatedness” of the objects drawn from a single peer. If σ_w^2 is small compared to σ_b^2 , then this indicates the presence of strong correlation within peers, so treating the objects from a peer as a random sample from the total object bag is extremely inaccurate and the contribution of each peer is better viewed as equivalent to only a few random samples. Conversely, if σ_w^2 is very large compared to σ_b^2 , the samples from each peer may in fact be treated as a random sample from the total object bag.

Cluster sampling is used often in data collection when it is impossible or impractical to obtain a random sam-

ple. Suppose that a survey is to be done in a large town and that individuals need to be given a questionnaire. Suppose further that the town contains 20,000 people and a sample of size 200 is needed. A simple random sample of 200 could well spread over the whole town incurring high costs and much inconvenience (e.g., travel) for the researcher. Instead, he might choose to randomly sample 40 streets and then randomly sample 5 individuals on each of those streets.

Clustered sampling has several advantages over random sampling: (i) the sampling cost is reduced as only a few peers must be seen; (ii) it is applicable even if no complete list of objects is available at the outset—as long as a complete list of clusters (peers) is available and each peer maintains a list of all the objects contained therein. The disadvantage of cluster sampling is that units within a cluster (e.g., people on a street in our example) may be more alike than randomly sampled units, so that the cluster sample is not as effective as a random sample of the same size. In the following discussion we quantify this drop in “effectiveness.”

To keep our argument straightforward, we frame our discussion in terms of the average object score. The result is applicable to our setting because the quantile of a particular object score is in fact the average of binary scores of all sampled objects (1 if the sampled object is less than or equal to the object score at hand; 0 otherwise). To develop our “effective” sample size we require more detailed notation than in the earlier sections. Let Y_{ij} denote the score of the j^{th} sampled object in peer i . The scores within a cluster (peer) are modeled as following a probability distribution with cluster mean μ_i and within cluster variance σ_w^2 . Further, the cluster means are assumed to follow a probability distribution with overall mean μ (the mean of the total object bag) and between cluster variance σ_b^2 . The theorem and lemma below are given for the special case in which the number of objects in each peer is the same, M , and the within cluster variance σ_w^2 is the same for each peer. The consequences of relaxing these assumptions are described at the end of this subsection. A further simplification is that we present formulas for the case of an infinite population of objects, thereby ignoring the so-called “finite population correction”. This simplifies the expressions for the variances in the formulas that follow without affecting the determination of the effective sample size.

Theorem 5 [5] *Suppose an infinite population of units is divided into disjoint clusters of M units each, and clusters are selected with cluster-level random sampling. Suppose we have sampled n random clusters, containing nM units, the sampling variance of the mean of all sampled objects ($\bar{Y} = \frac{1}{nM} \sum_{i=1}^n \sum_{j=1}^M Y_{ij}$) is:*

$$V(\bar{Y}) = \frac{\sigma_b^2 + \sigma_w^2}{nM} (1 + (M-1)\rho) \quad (10)$$

where ρ is the within-cluster correlation, defined as:

$$\rho = \frac{\sigma_b^2}{\sigma_b^2 + \sigma_w^2} \quad (11)$$

σ_b^2 is the variance between clusters;
 σ_w^2 is the variance within clusters.

From Theorem 5, we can calculate the *effective random sample size* which corresponds to a cluster sample of size nM units.

Lemma 6 *Suppose an infinite population of units is divided into disjoint clusters of M units each, and clusters are selected with cluster-level random sampling. Suppose we have sampled n random clusters, containing nM units in total. These samples are as effective as nM/λ independent random samples, where:*

$$\lambda = (1 + (M - 1)\rho) \quad (12)$$

If all $S = nM$ samples are independently drawn, then the variance of the mean is as follows:

$$V(\bar{Y}) = \frac{\sigma_b^2 + \sigma_w^2}{N} \quad (13)$$

If the $S = nM$ samples are from n clusters, then the variance of the mean is:

$$V(\bar{Y}) = \frac{\sigma_b^2 + \sigma_w^2}{nM} (1 + (M - 1)\rho) \quad (14)$$

By comparing Equation 13 and Equation 14, it is easy to show that nM samples from cluster sampling correspond to $\frac{nM}{1+(M-1)\rho}$ random samples.

We can develop some intuition about Lemma 6 by examining three cases.

Low Correlation: If σ_w is very large such that $\sigma_w \gg \sigma_b$, then $\rho \rightarrow 0$ and $\lambda \rightarrow 1$. If units within each cluster are quite random, then the nM cluster samples can be regarded as nM independent samples.

High Correlation: If σ_w is very small such that $\sigma_w \ll \sigma_b$, then $\rho \rightarrow 1$ and $\lambda \rightarrow M$. If units within each cluster are strongly correlated, then nM cluster samples correspond to only n independent samples (essentially we need to examine only one object per sampled cluster/peer).

Medium Correlation: If σ_w is comparable to σ_b , then ρ takes an intermediate value in $(0, 1)$ and λ takes an intermediate value in $(1, M)$. This indicates that the set of objects built with cluster sampling yields the same effectiveness (i.e., the same variance) as a smaller set selected with object-level random sampling.

To apply the above Lemma we require an estimate of ρ (also known as the intraclass (or intracluster) correlation) using the samples at hand. We estimate σ_b^2 and σ_w^2 and then use these estimates to construct an estimate of ρ . Let Y_{ij} denote the score of the j^{th} object from peer i , \bar{Y}_i the average score of the objects from peer i , and \bar{Y} the overall average of the objects from all

peers. We can compute the between-cluster (between-peer) mean Square(J_b) and the within-cluster (within-peer) mean Square(J_w) as follows:

$$J_b = \sum_{i=1}^n M (\bar{Y}_i - \bar{Y})^2 / (n - 1) \quad (15)$$

$$J_w = \sum_{i=1}^n \sum_{j=1}^M (Y_{ij} - \bar{Y}_i)^2 / (nM - n) \quad (16)$$

The expectation of J_b and J_w can be represented using σ_b^2 and σ_w^2 as [13] (The detail of the derivations of Equation 15-18 can be found in the full version [1]):

$$E(J_b) = \sigma_w^2 + M \sigma_b^2 \quad (17)$$

$$E(J_w) = \sigma_w^2 \quad (18)$$

As a result, we can estimate σ_b^2 , σ_w^2 via the Method of Moments by equating the observed J_b and J_w with their expectations to yield (\hat{x} is the estimate of x):

$$\hat{\sigma}_w^2 = J_w \quad (19)$$

$$\hat{\sigma}_b^2 = \frac{J_b - J_w}{M} \quad (20)$$

$$\hat{\rho} = \frac{\hat{\sigma}_b^2}{\hat{\sigma}_b^2 + \hat{\sigma}_w^2} = \frac{J_b - J_w}{J_b + (M - 1)J_w} \quad (21)$$

The above algorithm is fully distributed and can be computed incrementally. We don't need to store and sort all the accessed objects each time. Instead, we only need to know the best k distinct objects from the peer and the two summary statistics required to compute the mean squares, the sample sum of squares $\sum_{j=1}^M (Y_{ij} - \bar{Y}_i)^2$, and the sample mean, \bar{Y}_i . The mean squares J_b and J_w and hence the effective sample size can be computed at the query source given these summaries. Procedure UpdateEffectiveSampleSize() (Figure 3) implements this scheme. It is running on the query node and gets invoked whenever the summary from a newly sampled peer arrives. When it completes, the total number of effective samples is updated.

The algorithm uses point estimates of the various parameters to generate the estimated effective sample size. It is natural to wonder how errors in the estimated variance components propagate through to the estimated effective sample size. In general we expect σ_w^2 to be well estimated since it combines information from all sampled peers. The variability in our estimate of σ_b^2 depends on the number of sampled peers. It is possible to derive a confidence interval giving upper and lower limits of plausible values for ρ (one method is given in [13]). These limits could also be integrated with our quality estimation procedure if desired.

```

function: UpdateEffectiveSampleSize
Input: number of objects at new peer
(objNum), average score at new peer ( $\bar{s}$ ),
variance of scores at new peer ( $v$ )
Output: updated effective random sample
size( $S^*$ )
1. static n = 0, totalObjNum = 0, S = 0, S2 = 0, V = 0;
2. n++;
3. totalObjNum += objNum;
4. S+ = objNum *  $\bar{s}$ ;
5. S2+ =  $\bar{s}^2$ ;
6. V+ = v;
7.  $J_b = M * (S2 - S^2 / (n * M^2)) / (n - 1)$ ;
8.  $J_w = V / (nM - n)$ ;
9.  $\rho = (J_b - J_w) / (J_b + (M - 1)J_w)$ ;
10.  $\lambda = 1 + (M - 1)\rho$ ;
11. return  $S^* = totalObjNum / \lambda$ ;

```

Figure 3: Estimate the Number of Random Samples

4.3 Discussion

The results above have assumed that the number of units in each cluster is the same and that the within peer variance of objects is the same across all peers. Relaxation of these assumptions is discussed next, focusing first on the number of units in each cluster/peer. If the number of units within each peer varies greatly, then randomly sampling peers is not optimal. One can still get reasonable estimates of the quantile rank of an object but the variance tends to be high (i.e., the estimated quantile rank will vary greatly depending on whether peers with large numbers of objects have been included in the random sample). For such cases alternative sampling strategies such as sampling peers with probability proportional to the number of objects is sometimes used (see, for example, [5]). If we do opt to use random sampling, then the unequal cluster sizes effect our calculations in two ways. First, the expressions in Lemma 6 for defining the effective sample size change and second, the procedure for estimating the variance parameters that determine ρ must also change. The way in which they change depends on how we choose to estimate the overall average \bar{y} . If we let the number of objects in peer i be M_i , then one natural definition of the average in the case of unequal clusters is $\bar{y} = \frac{\sum_{i=1}^n \sum_{j=1}^{M_i} Y_{ij}}{\sum_{i=1}^n M_i}$. This just totals up the scores for all the objects in the total object bag and computes the average. It clearly gives more weight to peers with large numbers of objects as seems appropriate in this case. (An alternative estimate of the overall average is obtained by averaging the peer averages as in $\bar{y} = \frac{1}{n} \sum_{i=1}^n \left(\frac{1}{M_i} \sum_{j=1}^{M_i} Y_{ij} \right)$ but we don't pursue this case here). An appropriate modification of the Lemma (details in the full version [1]) yields the conclusion that the $\sum_{i=1}^n M_i$ object scores in the sample correspond to $\sum_{i=1}^n M_i / \left(1 + \left(\frac{\sum_{i=1}^n M_i^2}{\sum_{i=1}^n M_i} - 1 \right) \rho \right)$ random samples. If the M_i 's vary greatly then there is a much more substantial reduction in the effective sample size. Estimation of ρ must also be changed to reflect the unequal

number of objects within peers. The usual estimate of σ_w^2 is essentially unchanged, $\hat{\sigma}_w^2 = \frac{\sum_{i=1}^n \sum_{j=1}^{M_i} (Y_{ij} - \bar{Y}_i)^2}{\sum_{i=1}^n M_i - n}$ except that the expression for the total number of objects in the denominator now reflects the unequal sample size. We can still compute a version of J_b , specifically $J_b = \sum_{i=1}^n M_i (\bar{Y}_i - \bar{Y})^2 / (n - 1)$, but it turns out that $E(J_b) = \sigma_w^2 + \sigma_b^2 \frac{\sum_{i=1}^n M_i - \frac{\sum_{i=1}^n M_i^2}{\sum_{i=1}^n M_i}}{n - 1}$. The final modification then is to construct an estimate of σ_b^2 by setting this last expression to J_b . The formulas match those in (19)-(21) except that the common cluster size M is replaced by $\frac{1}{n-1} \left(\sum_{i=1}^n M_i - \frac{\sum_{i=1}^n M_i^2}{\sum_{i=1}^n M_i} \right)$.

Another possible complication occurs if the variation of object scores within peers differs among peers. This impacts the assumption of a single correlation parameter that we use to "correct" the sample size. There are a number of approaches that can be used to address this issue. It may be that there are types of peers, with the within peer variance similar among nodes of a given type but differing across types. If so, the effective sample size calculation might be done separately for each type. In the limit it is possible to adjust each peer's output to reflect that peer's local variation, i.e., using $\rho_i = \sigma_b^2 / (\sigma_b^2 + \sigma_{w,i}^2)$ to compute the effective number of objects from that peer, assuming that it is still reasonable to treat the peers as providing information about the same population of interest. Unfortunately for peers with a small number of objects the local estimate of the variation is likely to be highly uncertain itself thus we prefer to avoid single peer corrections of this type.

5 Supporting Quality Estimation in P2P search protocols

In this section we show how similarity queries can be implemented in P2P systems. We propose a two-level approach: (i) a *generic protocol* that is independent of the actual P2P system (Section 5.1) and (ii) *customization* of the protocol for various P2P systems (Section 5.2). This approach allows easy portability of the protocol to various P2P systems. To customize the protocol, a way to sample peers uniformly in a specific P2P system is needed. This is not straightforward and recent research has suggested several ways in which it can be done. In Section 5.2, we will explain random-sampling approaches for three important classes of P2P networks.

5.1 Generic Protocol: SiQueL

Our protocol (SiQueL for **S**imilarity **Q**uery **p**rotocol) is fully distributed and decentralized. SiQueL consists of a Request-Response pair of messages called *SimQueryMsg* and *SimQueryHitMsg* respectively:

SimQueryMsg: Carries information about a similarity query (f, q, k), Time-To-Live (TTL), and the query-initiator, *root*.

SimQueryHitMsg: Carries information about the k best distinct objects o_1, \dots, o_k at a peer, their number of copies $\{(o_j, c_j)\}$, and a summary of all local objects (objNum, average score(\bar{s}), score variance(v)).

An initiator-peer (*root* for short) starts SiQueL by sending a *SimQueryMsg* (with the necessary parameters) to another peer chosen uniformly at random with a specified TTL. The receiving peer (*target* for short) performs appropriate processing on its local objects (Figure 5) and sends a *SimQueryHitMsg* back to the *root*. While processing the query on its local objects, the *target* concurrently forwards the *SimQueryMsg* (after decreasing the TTL by 1) to another peer, chosen uniformly at random. Thus, while the original query is propagated in the P2P overlay, the *root* receives responses from target peers and continually updates the quality measure.

When the TTL hits zero, the *target* peer that receives this zero-TTL message, replies with a special type of *SimQueryHitMsg* to the *root*. The reply message has a **TTL_EXPIRED** field set to true. When the *root* receives this message, it checks to see if the appropriate quality bound has been achieved, failing which, it picks a new random *target* and starts the whole protocol again. The full SiQueL protocol is described in Figure 4. Two important sub-procedures, UpdateEffectiveSampleSize and EstimateQuality, are defined in Section 4.

Discussion: SiQueL is flexible and can be extended to trade speed for message overhead. We can forward *SimQueryMsg*, instead of one peer at a time, to c peers simultaneously. The number of peers targeted at each round increases exponentially; thus c and TTL must be chosen carefully.

SiQueL can also be applied to query for the *exact* top- k matches of a similarity query. Initially, SiQueL is run to form an estimate of the lower bound (threshold) of the top- k objects in the entire network with a certain confidence. This should only require contacting a few random peers. Once this is complete, a flooding of the network can be done with the similarity query along with the threshold. A node receiving this flooding query computes its top- k -object set, but transmits only those objects which exceed the threshold: if every object is below the threshold, no reply needs to be transmitted, saving some messaging cost. Each peer can also use this threshold to optimize its local search by limiting a local k -nearest neighbor search within a bound computed from the threshold.

5.2 Uniform Random Sampling of Peers

SiQueL assumes a method, *SAMPLE()*, that every peer uses to find a new random peer. We explain how *SAMPLE()* can be implemented in three classes of P2P systems: (1) Centralized, (2) Decentralized but structured and (3) Decentralized and unstructured.

In *Centralized* P2P systems, e.g., Napster¹ or Kazaa,²

¹<http://www.napster.com/>

²<http://www.kazaa.com/>

```
function: DoSimQuery
Input: time-to-live(tll), query ((f, q, k)),
threshold( $\{\tau_\phi, \tau_p\}$ )
Output: approximate answers and their
qualities to interface
1. qMsg = new SimQueryMsg(tll, (f, q, k), self);
2. Bag  $\mathcal{B}' = \{\}$ ;
3. Peer peer = SAMPLE();
4. send qMsg to peer;
5. while( $\min_{1 \leq i \leq k} \phi_i < \tau_\phi$ )
6.   REPLY = WaitForMsg();
7.   if(REPLY is TTL_EXPIRED)
8.     peer = SAMPLE();
9.     send qMsg to peer;
10.  else if (REPLY is SimQueryHitMsg)
11.    merge REPLY. $\{(o_j, c_j)\}$  into  $\mathcal{B}'$ ;
12.     $S^* = \text{UpdateEffectiveSampleSize}(\text{REPLY.objNum},$ 
      REPLY. $\bar{s}$ , REPLY. $v$ );
13.    ( $\{o_i\}, \{\phi_i, \tau_p\}$ ) = EstimateQuality( $\mathcal{B}'$ ,  $S^*$ ,  $\tau_p$ ,  $k$ );
14.    update interface;
15.  endIf
16. endWhile
```

Figure 4: The Main Thread on Querying Node

```
function: AnswerSimQuery
Input: SimQueryMsg (query)
Output: SimQueryHitMag (hit)
1. if ( query.ttl = 0 )
2.   send TTL_EXPIRED to query.root
3. else
4.   query.ttl --;
5.   Peer peer = SAMPLE();
6.   forward query to peer;
7.   count number of objects objNum;
8.   get  $k$  best distinct objects with # of copies  $\{(o_j, c_j)\}$ ;
11.   $\bar{s} = \frac{1}{\text{objNum}} \sum_{i=1}^{\text{objNum}} f(o_i)$ ;
12.   $v = \sum_{i=1}^{\text{objNum}} (f(o_i) - \bar{s})^2$ ;
13.  SimQueryHitMag hit( $\{(o_j, c_j)\}, \text{objNum}, \bar{s}, v$ );
14.  send hit to query.root;
15. endIf
```

Figure 5: Query Propagation

there exists a (logical) centralized entity that has global knowledge of the peers in the system. *SAMPLE()* can be implemented quite easily by a *Sampling Service* that returns, uniformly at random, one peer address from the global list. This takes only $O(1)$ messages (to contact the sampling service) to get one random sample.

In *Decentralized and Structured* P2P networks, such as Distributed-Hash-Table (DHT) based P2P systems³ (e.g. [20]), implementing *SAMPLE()* requires more sophistication. Theoretical properties guaranteed by these overlays must be used to derive the sampling scheme. We will describe two schemes to show the overhead involved (See [12, 14] for details). In [12], the authors present an algorithm which chooses a peer uniformly at random from the set of all peers in a DHT. In a network of n peers, a peer p is chosen with probability exactly $1/n$, sending, in the average case, $O(\log n)$ messages. In [14], the authors propose a distributed method to create and

³eDonkey, <http://www.edonkey2000.com>, is a commercial system based on this approach

dynamically maintain a random expander network. In this network, a random walk of $O(\log n)$ steps has almost equal probability of stopping in any of the n peers in the network. Thus, random sampling can be achieved with $O(\log n)$ messages.

In *Decentralized and Unstructured P2P* networks, there are neither central servers nor known theoretical overlay properties that can be applied directly for random sampling. However, in [10], the authors evaluate (empirically) several gossip policies, showing that an approximate random graph can be constructed from non-random graphs, by running a gossip protocol. Each peer maintains a cache of other peers and this cache is gossiped and updated. The graph of the peers’ caches is an approximate random graph. The *SAMPLE()* method can then be implemented via a simple lookup in the cache; no messaging cost is paid for several *SAMPLE()* calls which are locally handled, and the overhead is reduced to maintaining the cache of known peers.

6 Experiments

In this section we perform an empirical evaluation of the techniques developed in this paper. In Section 6.1 we discuss our methodology in terms of network topology, data sets, and performance measures. In Section 6.2 we discuss the results obtained from our experiments. Our goal is to show that a significant quality-performance trade-off can be realized when approximate similarity queries are executed over a P2P network, and high quality answers can be obtained at a fraction of the cost needed for exhaustive search.

6.1 Simulation Methodology

We study the effectiveness of our methods, varying P2P topology, data types, and performance metrics.

6.1.1 P2P Network Topology

As seen previously, the way in which peers are sampled randomly depends on the network structure. We deal with both structured and unstructured decentralized networks. We have also performed experiments for a centralized network, in which sampling is implemented with a single message to a central sampling service; results can be found in [1]. Unless stated otherwise, we use networks of size $N = 10,000$ in our experiments.

(1) **Unstructured:** We implemented a gossip protocol with the (rand, rand, pushpull) policy which was shown to perform well in the empirical evaluation presented in [10]. We started with a Power-Law network and ran the gossip protocol for 100 rounds: this populates the local caches in all the peers of the network. The *SAMPLE()* function is then implemented as a lookup in each node’s cache as previously explained.

(2) **Structured:** We built a Random Expander Graph [14]. Each node has exactly $2d$ edges, and duplicated edges are allowed. The bag of edges is composed

of d Hamilton cycles: every node has d successive and d preceding nodes, one for each of these cycles. Such a graph is known as a $H_{N,2d}$ -graph [14]; we follow the protocol described in [14] to construct it.

A random walk of length $t = 4\log_{d/2}N + 4$ is guaranteed [14] to reach peer v with probability $|\Pr\{\text{reach } v\} - \frac{1}{N}| \leq \frac{1}{N^2}$. We set $d = 20$ in our experiments, and therefore a random walk of length 20 will generate random peers with high probability.

Network topology changes as peers join or leave the network. We assume that such topological evolution takes place at a pace slow enough not to affect a particular query’s execution.

6.1.2 Data types

Both synthetic and real data are used in our experiments:

(1) **Synthetic data:** The similarity score of objects is randomly allocated. For simplicity, every peer is assumed to host the same number of objects. Different distributions of scores may be observed in practice, and we examine two of them, *Uniform Allocation* and *Cluster Allocation*, which serve to test our algorithms in the case where scores within each peer are not correlated or are similar to each other respectively.

(1.1) *Random Allocation:* 20 scores uniformly drawn from $[0, 10000]$ are assigned to each peer.

(1.2) *Clustered Allocation:* For each node n_i , we first pick its mean score $\mu_i \sim N(\mu, a)$. Then 20 scores are generated for n_i , following a Gaussian distribution $N(\mu_i, b)$. We set $\mu = 5,000$, $a = 500$, and vary b : this controls the “tightness” of the clusters, with $\frac{b}{a}$ capturing the intensity of correlation. All scores are limited to be in $[0, 10,000]$.

(2) **Real data:** 68,040 images in 708 categories from the Corel Image Features data set in UCI KDD Archive [8] were used. Each image is represented as a 32-d color histogram [21], one of the most widely used visual features in image retrieval and is provided with the data set. If $H(o)$ is the color histogram of image o and $H(q)$ is the color histogram specified in the query, the similarity between q and o is defined as:

$$f(q, o) = \sum_{i=1}^{32} \min(H_i(q), H_i(o)) \quad (22)$$

where H_i is the i^{th} value of vector H . As with our synthetic data, we use two methods to allocate images:

(2.1) *Random Allocation:* Each peer is assigned 20 images randomly from the full dataset.

(2.2) *Clustered Allocation:* For each node, a category is first randomly selected, and then images from that category are randomly assigned to it. This results in similar images being assigned to each node.

We assume that the distribution of objects is fixed during query evaluation. In practice, objects are added and deleted from peers, but this is assumed to be negligible during the short time of query execution.

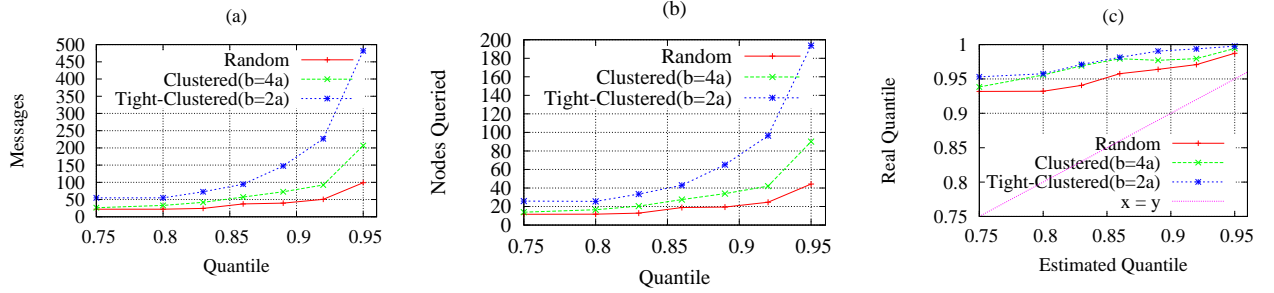


Figure 6: Cost in Power-Law Random Graph with Synthetic Data ($\tau_p = 0.95$)

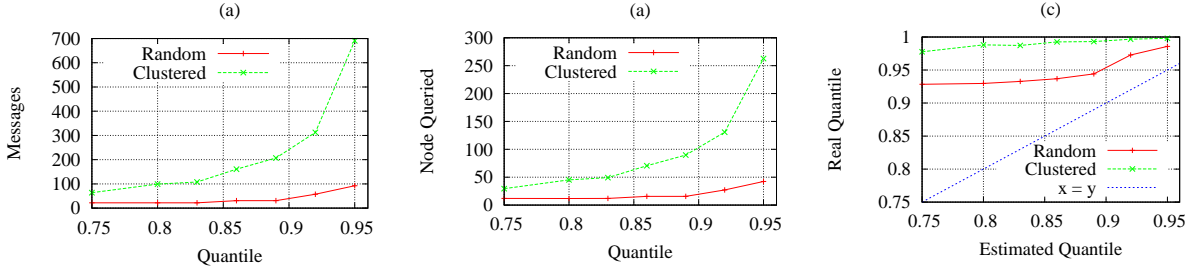


Figure 7: Cost in Power-Law Random Graph with Image Data ($\tau_p = 0.95$)

6.1.3 Query Formulation and Performance Metrics

For each simulation run, we pick a node as the source of the query. The scores are generated either artificially in our synthetic data, or using Equation 22 with one randomly selected image in the dataset as the query. The search algorithm is terminated when the quality requirement (τ_ϕ, τ_p) is reached. The confidence τ_p is always fixed to be 0.95 and the quantile τ_ϕ varies from 0.75 to 0.95. We invoke the quality estimation module after the 5th peer is accessed, so as to build a non-trivial initial sample size. For each τ_ϕ level, the following performance measures are included:

Number of Messages: The total number of messages generated during the search, including SimQueryMsg, SimQueryHitMsg, TTL_EXPIRED and SAMPLE messages. This is a measure of the network activity generated by the query.

Number of Peers Queried: The total number of peers that have processed the query locally. This measures the number of other computers which must do some work in response to the query.

Real Quantile: The quantile of the k^{th} best object in the approximate answer, in terms of all objects in the network.

To assess the quality of our approximate answer we use the real quantile instead of a measure such as recall, i.e., the fraction of our best k objects that are also in the best k over all the objects of the network. This is a better measure, because it captures the proximity of scores in the approximate answer to those in the exact one. Recall cannot capture this, as it penalizes all objects which are

not in the global best k , irrespective of their closeness to the best ones.

Each query is repeated 20 times with different starting peers, and results are averaged over these runs. We use the performance of an exact similarity query answering system as a comparative baseline. This must send and receive a message from each peer in the network. So, in a network of N peers, all N must be accessed and $2N$ messages must be transmitted.⁴

6.2 Experimental Results

The results for the unstructured-network can be seen in Figures 6, 7. The quantile τ_ϕ is shown on the horizontal axis and different performance metrics are plotted on the vertical axis.

Figures 6a,b and 7a,b show the cost for varying threshold τ_ϕ . Better quality requires higher cost, but as τ_ϕ approaches unity, the cost starts to rise rapidly, indicating that increasingly lower gains are expected for the same additional effort. The cost of obtaining an exact answer would be 10,000 peers and at least 20,000 messages, which greatly exceeds the range shown for the approximate evaluation. These results demonstrate the great potential performance benefits that approximate answering can realize in this setting, and the smooth improvement in quality as more effort is expended. It also confirms our intuition that, after a certain number of peers have been retrieved, quality improves more slowly, and the user will be inclined to terminate the query.

⁴In a real network even more messages must be exchanged, since the querying node cannot contact all the N nodes, and hence some nodes may be contacted more than once.

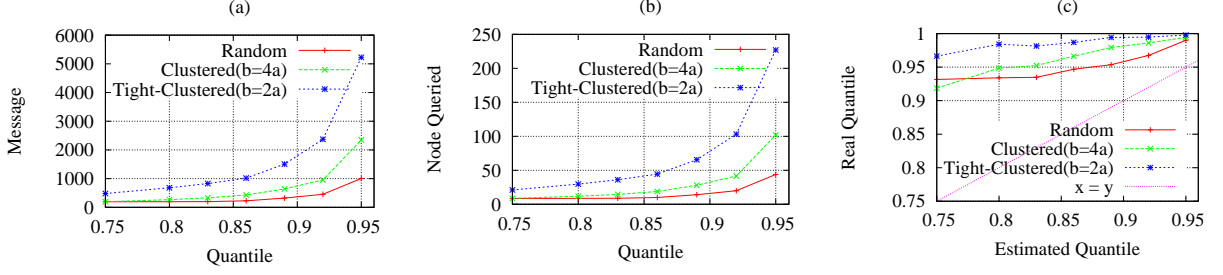


Figure 8: Cost in Expander Graph with Synthetic Dataset ($\tau_p = 0.95$)

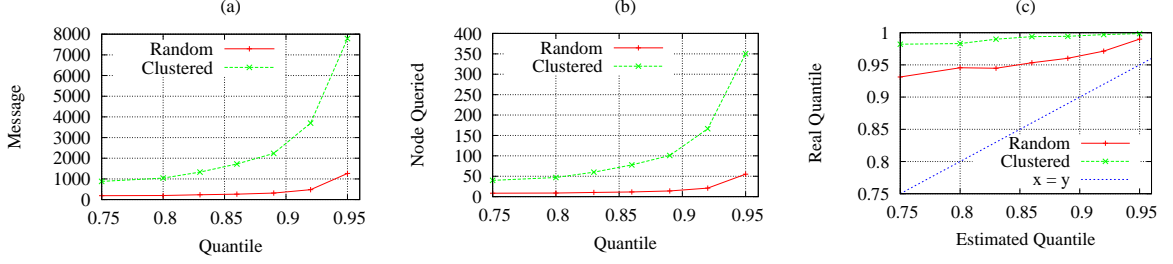


Figure 9: Cost in Expander Graph with Image Dataset ($\tau_p = 0.95$)

Furthermore, we observe that cost is related to the “correlation,” with higher cost corresponding to more correlated distributions. Our effective random sample size factor λ increases when within-peer correlation exists, and this results in more peers being sampled to achieve the same quality guarantee.

Figures 6c, 7c demonstrate the relationship between the real and estimated quantiles. All curves are above line $x = y$, indicating that the real quality of the approximate answer is actually higher than its reported value. Thus, if a user accepts an approximate answer and terminates his search, he will obtain an answer which is actually even better than promised by our system.

In Figures 8a, b and 9a, b, we study the cost-quality tradeoff in the expander network. Since we have to rely on a random walk to obtain a random peer, the number of messages is higher than in the previous case. To retrieve a single random sample peer, a random walk of length $O(\log N)$ must be performed and thus $O(\log N)$ SAMPLE messages are sent during sampling. However, the number of nodes needed to process the query remains relatively small compared to the total number of nodes in the network, since peers on the random walk path only need to relay SAMPLE messages instead of processing the query. Figures 8c and 9c verify that our quantile estimate is the lower bound of the real quantile in this case as well.

An interesting observation, applicable to both network topologies, is the high cost when peers contain objects from a single category, a case of extreme correlation. In real-life settings, peers will probably contain objects from multiple categories. Thus, the cost will be between that of random assignment and the very clustered single-

category assignment used in our experiments.

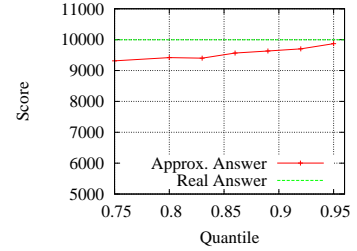


Figure 10: Real and approximate score threshold.

We also test the closeness of our approximate answer to the real one using synthetic data in Power-law network; results were similar for the other tested cases. In Figure 10 the score of the k^{th} object in the approximate answer is compared to the score of the real k^{th} best object. This shows how our answer progressively approaches the exact one. The score of the k^{th} object is 9720 when ϕ is 0.92, very close to the exact score which happens to be 9994 in this case.

Finally, we test the scalability of our system in Figure 11. Synthetic data with with power-law random networks of varying size are used. As expected, the quality-cost curves are very similar irrespective of the underlying network size. This is due to the nature of our quality estimation method which makes no assumptions about the network size, but treats objects as samples from an underlying distribution. This contrasts with the cost of exact search which increases with larger network size. Thus, our algorithm will remain useful even as P2P networks grow in size.

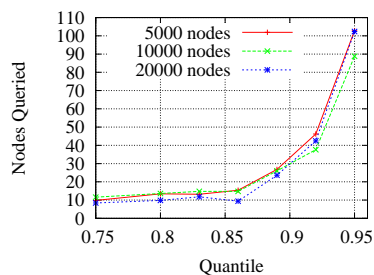


Figure 11: Network size impact on scalability.

6.3 Summary of Results

From our experiments, we conclude that: (i) our algorithm trades quality for cost efficiently; (ii) high quality is achieved with low cost irrespective of network size; (iii) correlation within peers results in higher cost; (iv) the real quality of the approximate answer given by the system is higher than our provable bounds.

7 Conclusions

This paper presents a framework to support approximate similarity queries in a P2P network. The advantage of such a system is that the user may monitor the progress of the query and tune its cost according to his quality needs. Since objects are grouped in peers, obtaining a random sample for the purpose of quality estimation is problematic, and we show how this can be produced, and how its quality can be assessed.

We assumed that the network topology and content distribution change slowly compared to the duration of query execution; an extension is to study similarity queries when this assumption does not hold, either because of a highly dynamic network or a long-standing continuous similarity query. We have not dealt with the interplay of multiple queries: caching of popular results may benefit in this case.

References

- [1] Supporting Approximate Similarity Queries with Quality Guarantees in P2P Systems (full), <http://www.ics.uci.edu/~qzhong/p2pfull.pdf>.
- [2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
- [3] S. Chaudhuri, V. Narasayya, and R. Ramamurthy. Estimating progress of long running sql queries. In *SIGMOD Conference*, 2004.
- [4] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of International Conference on Very Large Data Bases*, 1997.
- [5] W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, 1977.
- [6] M. Flickner, H. Sawhney, W. Niblack, and et al. Query by Image and Video Content: The QBIC System. In *IEEE Computer*, volume 28, 1995.

- [7] J. M. Hellerstein, P. J. Haas, and H. Wang. Online aggregation. In *SIGMOD Conference*, 1997.
- [8] S. Hettich and S. D. Bay. The uci kdd archive [<http://kdd.ics.uci.edu>], 1999.
- [9] W. Hoeffding. Probability inequalities for sums of bounded random variables. In *Journal of the American Statistical Association*, pages 13–30, 1963.
- [10] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In *5th Int'l Middleware Conf*, 2004.
- [11] I. King, C. H. Ng, and K. C. Sia. Distributed content-based visual information retrieval system on peer-to-peer networks. *ACM Transactions on Information Systems (TOIS)*, 22:231–262, 2004.
- [12] V. King and J. Saia. Choosing a random peer. In *PODC*, 2004.
- [13] M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li. *Applied Linear Statistical Models (fifth edition)*, chapter 25. McGraw-Hill, 2004.
- [14] C. Law and K.-Y. Siu. Distributed construction of random expander networks. In *INFOCOM*, 2003.
- [15] I. Lazaridis and S. Mehrotra. Progressive approximate aggregate queries with a multi-resolution tree structure. In *SIGMOD Conference*, 2001.
- [16] G. Luo, J. F. Naughton, C. J. Ellmann, and M. W. Watzke. Toward a progress indicator for database queries. In *SIGMOD Conference*, 2004.
- [17] A. N. Papadopoulos and Y. Manolopoulos. Distributed processing of similarity queries. *Distributed and Parallel Databases archive*, 9:231–262, 2001.
- [18] Y. Rui, T. S. Huang, and S. Mehrotra. Content-based Image Retrieval with Relevance Feedback in MARS. In *Proceedings of IEEE Int. Conf. on Image Processing (ICIP)*, 1997.
- [19] J. R. Smith and S.-F. Chang. VisualSEEK: A Fully Automated Content-Based Image Query System. In *Proceeding of ACM Multimedia*, 1996.
- [20] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. In *IEEE/ACM Trans. Netw.*, 2003.
- [21] M. Swain and D. Ballard. Color Indexing. *International Journal of Computer Vision*, 7, 1995.
- [22] B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *ICDCS Conference*, 2002.