Available online at www.sciencedirect.com
ScienceDirect
J. Parallel Distrib. Comput. 68 (2008) 64–77
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

Semantic indexing in structured peer-to-peer networks

Ronaldo A Ferreira[a,1], Mehmet Koyutürk[b,1], Suresh Jagannathan[c], Ananth Grama[c,*]

[a]*Department of Computer Science, Federal University of Mato Grosso do Sul, Brazil*
[b]*Department of Electrical Engineering & Computer Science, Case Western Reserve University, USA*
[c]*Department of Computer Science, Purdue University, USA*

Received 23 February 2007; accepted 12 June 2007
Available online 26 June 2007

Abstract

The past few years have seen tremendous advances in distributed storage infrastructure. Unstructured and structured overlay networks have been successfully used in a variety of applications, ranging from file-sharing to scientific data repositories. While unstructured networks benefit from low maintenance overhead, the associated search costs are high. On the other hand, structured networks have higher maintenance overheads, but facilitate bounded time search of installed keywords. When dealing with typical data sets, though, it is infeasible to install every possible search term as a keyword into the structured overlay.

State-of-the art semantic indexing techniques have been successfully integrated into peer-to-peer (P2P) systems using semantic overlays. However, exiting approaches are based on the premise that the fundamental ingredient of semantic indexing, a semantic basis for the underlying data, is globally available, which is not likely to be the case in practice. Therefore, development of techniques to efficiently compute basis vectors for data distributed across peers is important for large-scale deployment of semantic indexing in P2P systems.

In this paper, we present a novel structured overlay that integrates aspects of semantic indexing using non-orthogonal matrix decompositions, with the hash structure of the overlay. We adopt PROXIMUS, a recursive decomposition method for computing concise representations for binary data sets, to locally identify latent patterns in data distributed across peers. To enable efficient consolidation of patterns, we rely on distributed hash tables (DHT), commonly used in various applications in P2P networks. The discrete nature of non-orthogonal matrix decomposition is well suited to the binary key structure of DHTs, resulting in an indexing method, PMINER, that enables the network to deliver efficient and accurate responses to semantic queries. We present the algorithmic underpinnings of PMINER and demonstrate its excellent performance characteristics on real, as well as synthetic data sets.

Keywords: Peer-to-peer networks; Semantic indexing; Matrix decompositions; Information retrieval; Distributed hash table; Semantic overlay networks

1. Introduction

Developments in data-acquisition technologies, combined with advances in digital storage and high-speed communications, enable collection and archival of large distributed data sets. A highly visible example of such an infrastructure is in bioinformatics, where researchers accumulate terabytes of sequence, expression, and interaction data in publicly accessible repositories. In commercial enterprise, retailers such as Walmart record terabytes of customers' transactions daily. These large data sets are generally multi-attributed (high dimensional), discrete, and distributed. For typical applications, collecting all of the data to a single central location is infeasible, due to associated overheads, privacy concerns, and resource constraints. Therefore, querying and analyzing these data sets require the development of distributed techniques.

Peer-to-peer (P2P) systems have emerged as attractive solutions for a number of distributed applications, such as file sharing [3], archival storage [11], backup [7], web caching [14], and information retrieval systems [31], among others. More recently, researchers have explored the integration of data analysis and mining techniques into P2P systems [10]. Given the large amount of data already residing in P2P networks, decentralized solutions to data mining problems have the potential for new and exciting applications.

* Corresponding author.
E-mail addresses: raf@dct.ufms.br (R.A. Ferreira), koyuturk@eecs.case.edu (M. Koyutürk), suresh@cs.purdue.edu (S. Jagannathan), ayg@cs.purdue.edu (A. Grama).
[1] The work was performed while the authors were at Purdue University.

0743-7315/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2007.06.003

In this paper, we examine the problem of affecting semantic queries on structured P2P networks. Semantic indexing techniques can be integrated into P2P systems using distributed hash tables (DHTs), however, a key component of semantic indexing, computation of basis vectors, remains unresolved in P2P networks. The core idea of our approach is integrating semantic indexing techniques into a P2P system to enable smart query processing, while avoiding centralized mechanisms and global communication, with minimum *a priori* information that is globally available. Specifically, we integrate binary non-orthogonal matrix decompositions, which extract discrete latent semantic vectors from binary data sets, into a DHT to build a distributed storage infrastructure capable of supporting semantic queries. As we illustrate, the key advantage of binary decompositions in this application comes from their discrete nature, which ideally suits the distributed indexing techniques commonly used in P2P systems.

The proposed storage infrastructure, pMINER, builds on the ideas of PROXIMUS [18,19] and its coarse-grained parallelization, CONQUEST [6], to develop an efficient and completely decentralized P2P system for decomposing binary data sets distributed across peers. In pMINER, peers are organized into a semantic overlay to compute an approximate binary non-orthogonal matrix decomposition. To facilitate integrity between peers, we adopt a coarse-grained parallelization approach that relies on the principle of subsampling. This strategy has been shown to be effective in minimizing communication overhead while maintaining the quality of the solution. Peers compute rank-one approximations of the matrices that represent their local data and go through consolidation phases to achieve global low rank approximations.

In the consolidation phase, peers find patterns that are similar and can be merged together to compute a single rank-one approximation across multiple peers. This enables identification of global patterns in the network. The key issue in performing matrix decompositions in a P2P network is one of performing this computation without any centralized mechanism or global communication. We achieve this by using DHTs to index approximation vectors in such a way that similar vectors are hashed to neighboring peers in the overlay. Then, local computation groups are formed by peers whose identifiers are within a given bounded Hamming distance from each other. Peers consolidate their approximation vectors with those in their computation groups, to identify similar patterns.

DHTs provide the ideal substrate for this approach, since peers are identified by binary strings. We hash binary vectors to peers, i.e., map them to a lower dimensional binary space, to: (i) preserve proximity, i.e., ensure that vectors close to each other are hashed to keys (hence, peers) that are close to each other, and (ii) the keys are distributed uniformly. Consequently, approximation vectors are indexed efficiently, since a peer with a specific identifier can be found within a bounded number of hops (commonly $O(\log N)$) in an N-node network. Moreover, the first property provides accurate consolidation by ensuring that potentially similar vectors are processed together in a computation group, while the second property ensures that load balance is maintained.

We implement a prototype system for pMINER and evaluate its performance on a cluster of 18 workstations. To evaluate a larger P2P network, each workstation runs multiple processes representing different peers. In the evaluation of the system, we use both real-world and synthetically generated data sets. The real-world data set is a database of customer's transaction from Walmart. The quality of an approximation can be measured in terms of precision, recall, and number of patterns discovered (compression). Our experiments show that pMINER achieves levels of precision and recall that are close to that of the underlying serial algorithm. We also investigate distributed performance issues of our system and show that it achieves excellent speedup.

The rest of the paper is organized as follows. In the next section, we discuss related work, briefly describe semantic indexing techniques, and elaborate on PROXIMUS and CONQUEST. In Section 3, we describe our system and associated algorithms. We present experimental results in Section 4. Finally, in Section 5 we conclude our discussion.

2. Background and related work

2.1. Matrix decompositions and semantic indexing

Matrix transforms, such as truncated singular value decomposition (SVD) [12] and semi-discrete decomposition (SDD) [16], are used extensively for processing and analyzing high-dimensional data. The applications of these linear algebraic methods include identification of underlying patterns (e.g., principal component analysis [24]), classification, feature extraction, compression, noise elimination, and dimensionality reduction [30,34]. In information retrieval, the ability of matrix transforms to identify strong underlying patterns and eliminating noise, thereby capturing *latent* semantic information, is effectively used for processing semantic queries [2].

SVD decomposes a matrix A into two orthogonal matrices (U and V) and a diagonal matrix (Σ) of singular values, such that $A = U\Sigma V^T$. Columns of the two orthogonal matrices, known as singular vectors, characterize the dominant patterns (principal components) in the matrix. Each pair of singular vectors is associated with a (scalar) singular value, characterizing the strength of the pattern in the matrix. Therefore, truncating the decomposition to only a portion of singular vectors that correspond to the largest singular values, leads to an approximation ($\tilde{A} = U_k\Sigma_k V_k$) that captures semantic information accurately. This method is known as *latent semantic indexing* (LSI) in information retrieval.

A major problem with SVD is that its computation is quite expensive and the resulting approximation is dense. SDD alleviates these problems by computing a fast approximation to SVD. In this approximation, the entries of singular vectors are restricted to the set $\{-1, 0, 1\}$. This approximation has the benefit of compressing, significantly, the resulting matrices, since each value can be represented using only 1.5 bits. SDD is also shown to be effective in LSI. However, despite being a non-orthogonal matrix decomposition, SDD also suffers from a problem that is caused by the orthogonality constraint in SVD.

As in SVD, the most significant approximation vectors correspond to a rank-one approximation to the original matrix, and each subsequent pair of vectors approximate the *residual* matrix, which is the difference between the approximated matrix and the approximation matrix. Therefore, while decomposing a matrix with only non-negative entries, which is the case in most applications of semantic query processing, the negative values in approximation vectors correspond to *correction* of extra information introduced by more significant approximation vectors. This makes the analysis and use of discovered patterns difficult, and in some cases meaningless (e.g., what is the physical interpretation of a negative number in binary data?). PROXIMUS, the non-orthogonal matrix transform used in PMINER, alleviates this problem by computing binary approximations to binary data sets.

2.2. Data mining, analysis, and consolidation in P2P networks

Data mining in P2P networks is still in relative infancy. Datta et al. [10] present a survey of current work in this area, highlighting the main differences between centralized and P2P environments. They also specify a set of requirements for P2P solutions, such as scalability, decentralization, privacy, and asynchrony, among others. Recent efforts in P2P data mining concentrate on the computation of basic statistics, such as averages, sum, maximum, and minimum [15,17]. Kowalczyk [17] proposes a method, called NEWSCAST, for computing the mean of data distributed in a P2P network using an epidemic protocol. The protocol tries to preserve data privacy, while handling a high transient population of nodes. An evolving P2P solution for the k-means clustering problem is proposed in [33]. In this algorithm, peers communicate only locally with their neighbors and use majority voting to achieve consensus on the solution. Application of these mining, analysis, and consolidation techniques to information retrieval in P2P systems is promising for efficient and effective query processing.

2.3. Semantic overlay networks

A recent approach that facilitates processing, analysis, and exchange of semantic information in P2P networks is based on semantic overlay networks [1,22,31]. The general idea in semantic overlay networks is to group peers with semantically similar content together. In PSEARCH [31], a P2P information retrieval system based on a semantic overlay network, documents and queries are represented as vectors in a multi-dimensional Cartesian space. The basis vectors for this *semantic space* are computed using LSI. Each document corresponds to a point in this space, indexing of which is implemented using the DHT system CAN. This DHT implementation represents the *logical space* of peers as a multi-dimensional Cartesian space as well. Therefore, the resulting placement function for indexing documents provides the advantage that documents with similar contents are placed close to each other in the network. A query is routed toward the point in the semantic space represented by its respective vector. Consequently, while processing a query, only the part of the network that contains vectors that are close to the query vector in the semantic space is searched. This technique effectively localizes and decentralizes query processing.

One major problem for practical deployment of pSearch, however, is the computation of the basis for the semantic space. PSEARCH assumes that the peers in the network share some global document statistics that allow the computation of a common basis. However, computing a global basis that captures the global semantic information is not a straightforward task in a P2P network. In this paper, we show how the basis vectors can be computed without requiring decentralized mechanisms and global communication. We achieve this by taking advantage of the discrete and hierarchical nature of binary non-orthogonal decompositions. As in PSEARCH, PMINER also explores the geometry of the DHT, but the logical space is in the form of a *binary hypercube*, which suits the discrete nature of our binary approximations. Furthermore, hierarchical decomposition does not suffer from numerical deficiencies posed by orthogonality constraints, facilitating efficient and accurate consolidation of basis vectors.

2.4. Non-orthogonal decomposition of binary matrices

PROXIMUS [18,19] is an algebraic framework, composed of novel algorithms and data structures, to compute error-bounded approximations for binary attributed data sets. It relies on the idea of non-orthogonal matrix decomposition, which recursively extracts patterns and partitions the matrix accordingly. At each step, it computes a binary rank-one approximation for the matrix, to maximize the number of non-zeros captured by the approximation, while minimizing those that are introduced. The binary rank-one approximation problem is formulated as follows.

Definition 1 (*Rank-one approximation*). Given a matrix $A \in \{0,1\}^m \times \{0,1\}^n$, find $x \in \{0,1\}^m$ and $y \in \{0,1\}^n$ that minimize the error:

$$\|A - xy^T\|_F^2 = |\{a_{ij} \in (A - xy^T) : |a_{ij}| = 1\}|. \quad (1)$$

It can be easily shown that minimizing the error in a rank-one approximation is equivalent to maximizing [19]

$$C(A, x, y) = 2x^T Ay - \|x\|_2^2 \|y\|_2^2. \quad (2)$$

A heuristic solution to this problem relies on an iterative improvement strategy. The idea is to find an optimal solution to x for a fixed y, and then use this solution to x to find a new y. Clearly, each such iteration only improves the solution, and the process eventually converges to a local maximum. This results in an alternating iterative method, that continues until no improvement on the objective function is possible. The time complexity of this algorithm is linear in the number of non-zeros of A. A key issue in converging to a *good* local maximum is initialization. We present several initialization strategies and discuss their effectiveness in [19].

The recursive structure of PROXIMUS is illustrated in Fig. 1. A rank-one approximation captures the dominant pattern in a matrix A, and produces *pattern vector* y and a corresponding

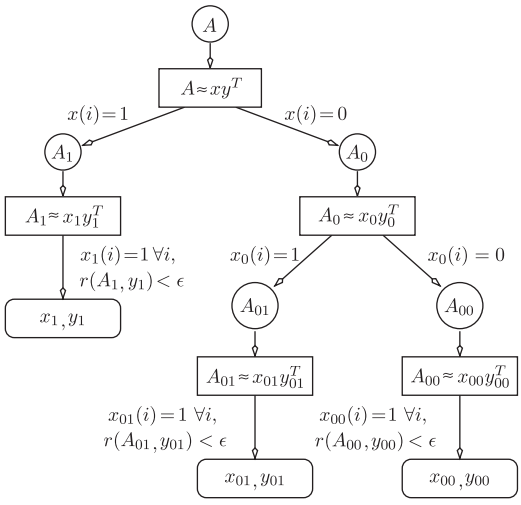

Fig. 1. Recursive structure of PROXIMUS. The tree shows the recursive decomposition of a matrix. Each rectangular internal node is a rank-one approximation and two circular children of these nodes are the matrices that result from partitioning of parent matrix based on this approximation. The leaves of the tree correspond to the final decomposition.

presence vector x. These vectors are used to partition A into two submatrices A_1 and A_0. A_1 contains the rows of A that contain the pattern characterized by y, i.e., those that correspond to ones in x. A_0 contains the remaining rows of A (zeros in x). Observe that rows of A_1 share some a common pattern characterized by y, i.e., they are centered around y in the binary hyperspace, where each dimension corresponds to a column of A. What we learn from the rank-one approximation about the rows of A_0, however, is only that they are of certain distance from y. Therefore, PROXIMUS computes a new rank-one approximation for A_0 and partitions this matrix recursively. On the other hand, if x and y adequately approximate the non-zero structure of A_1, i.e., the binary hyper-sphere consisting of rows of A_1 centered around y is sufficiently small, then y is identified as an approximation vector and the recursion stops for A_1. Otherwise, A_1 is also recursively partitioned following the same strategy. To evaluate the adequacy of an approximation, PROXIMUS uses a threshold on Hamming radius, denoted ε, as the stopping criterion. The Hamming radius $\rho(A, y)$ of matrix A, around a vector y, is defined as the maximum of the Hamming distances of A's rows to y. The recursive algorithm does not partition a sub-matrix A_i if the following conditions hold for the rank-one approximation $A_i \approx x_i y_i^T$:

(1) $\rho(A_{i1}, y_i) < \varepsilon$, where ε is the bound on Hamming radius.
(2) $x_i(j) = 1 \forall j$, i.e., all rows of A_i are present in A_{i1}.

If the second condition holds, but not the first, only the rows that are within ε distance from the pattern vector are retained in the final approximation and the recursion continues for the rest of the rows. Consequently, in the final decomposition $A \approx XY^T$, for every row $A(i)$ of A, there is at least one row $Y(j)$ of Y such that the Hamming distance between $A(i)$ and within $Y(j)$ is at most ε. The corresponding entry of X, $X(i, j)$, is equal to 1.

2.5. Coarse-grained parallel algorithms for non-orthogonal decomposition of binary matrices

CONQUEST [6] is a coarse-grained parallel formulation of PROXIMUS that relies on the principle of subsampling to reduce communication overhead. In CONQUEST, processors compute their local patterns independently and communicate among themselves to consolidate their patterns. The communication overhead is minimized using work groups, which are aggregates of processors that are working on data with similar patterns. Work groups are formed adaptively according to the patterns identified by local approximations, where similar patterns are identified according to being within a bounded Hamming distance of each other. Patterns are consolidated within each work group by computing a rank-one approximation to the matrix composed of approximation vectors, each of which belong to one of the processors in the work group. More specifically, all processors compute a rank-one approximation of their matrices and broadcast the resulting pattern vectors in the network. Upon receiving patterns from all processors, a processor identifies the processors that contain patterns similar to that of itself, and forms a work group to consolidate their patterns. Processors within the same work group continue working together recursively for the rows of their matrix that is represented by the pattern that characterizes this work group. Therefore, at each rank-one approximation and subsequent partitioning step, the work groups are also partitioned along with the matrices. Note that this generates a tree of work groups as well, and a processor gets into various work groups throughout the course of recursive decomposition, to decompose different parts of its matrix.

We adapt the coarse-grained parallel formulation of CONQUEST in PMINER as well. However, it is not straightforward to apply this algorithm to P2P platforms, because of the requirement for global communication to broadcast approximation vectors. Nevertheless, the coarse-grained nature of this approach allows application of DHT-based consolidation schemes, by indexing patterns in peers to preserve their proximity, and forming work groups with respect to locality in the logical space of peers characterized by the overlay.

3. PMINER: A framework for computing global bases for binary datasets in P2P systems

PMINER integrates a structured overlay with PROXIMUS to architect a true semantic overlay network. In PMINER, peers organized into a semantic overlay compute local rank-one approximations and go through consolidation phases to achieve global rank-one approximations. In the consolidation phase, we aim to find patterns that are similar and that can be merged together to achieve a common rank-one approximation across multiple peers. The consolidation phase is performed in multiple rounds of communication by peer groups.

3.1. Problem definition

Consider the following scenario: there are p peers in a P2P network, with peer i hosting m_i documents to share with other

peers. Here, we use the term document to specify a data element that is of interest in its entirety, which, for example, may correspond to a scientific article, a transaction, or a biological sequence in a practical application. The documents are composed of terms, dereferenced by a common dictionary of n terms. In practice, the terms may correspond to words, items, or features. In total, there are $m = \sum_{i=1}^{p} m_i$ documents in the network.

A semantic query by a peer specifies a set of terms and aims to locate and possibly retrieve all documents in the network that are relevant to these terms. The purpose of pMINER is to provide an indexing mechanism that efficiently and accurately identifies relevant documents for a query, by finding a compact representation of documents that is consistent throughout the network, with minimum requirement on *a priori* knowledge about the global statistics of the documents in the network. In algebraic terms, this representation is characterized by a set of basis vectors.

We model documents and queries as vectors in n-dimensional binary hyperspace, such that each term corresponds to a dimension. If a document contains a term, then the corresponding entry of the document vector is equal to one; it is zero otherwise. Based on this model, the non-orthogonal decomposition problem in P2P systems is formulated as follows:

Definition 2 (*Binary non-orthogonal matrix decomposition in a P2P network*). Given m binary vectors $v \in \{0, 1\}^n$, distributed across p peers, find a set of k binary approximation vectors $w \in \{0, 1\}^n$, $k \ll m$, such that for any input vector v, there is an output vector w, with $d(v, w) \leqslant \varepsilon$. Here $d(v, w) = \|v - w\|^2$ denotes the Hamming distance between v and w, which is equal to the number of bits they differ at.

This problem is one of identifying the basis vectors that are used for indexing the documents. The above definition does not explicitly specify where the approximation vectors are to be located in the network, since this is handled by the indexing scheme. However, it should be noted that the aim here is to provide a global view of the approximation; that is, each peer has quick access to the basis vectors, as facilitated by the indexing scheme. Indeed, in pMINER, the DHT implementation used for indexing documents is also used for efficiently consolidating approximation vectors, resulting implicitly in a global set of basis vectors.

3.2. Distributed hash table systems

DHTs are used in a number of P2P systems to provide placement and location of objects in a network of cooperating peers. The main goal of such systems is to provide two basic operations, namely `PUT` and `GET`. `PUT` takes a key and an item and places them at a specific peer in the network, and `GET` takes a key and retrieves the value previously stored under that key from the network. Peers are generally identified by binary strings of fixed length. A key is computed using a hash function shared by the peers and the item is placed at a peer whose identifier is closest to the key. The notion of closeness varies from system to system. Chord [29], for instance, places a key at the peer whose ID follows the numeric value of the key in the virtual space, while Pastry [27] places a key at the peer whose id is numerically closest to the key. Other systems of interest, such as CAN [25] and Kademlia [23], map the peer IDs to a logical space and locate keys in peers based on proximity in this space. DHTs provide the infrastructure for building more complex systems, such as distributed file systems [9,20,28], name services [8], web caching [14], information retrieval systems [31], and publish-subscribe systems [4]. The organization of peers in a DHT, based on their identifiers, provides an ideal substrate for processing binary patterns resulting from local matrix approximation.

3.3. Computing global approximation vectors in a P2P system

The process of computing basis vectors and processing queries in pMINER is illustrated in Fig. 2. On the right-hand side of the figure, the flow of processing a semantic query is shown. This approach mirrors methods that are successfully used in semantic overlay based information retrieval in P2P systems, such as pSEARCH. However, existing approaches assume that a set of basis vectors that are used to process queries in semantic indexing schemes, are globally available. This task, which is one of computing a global approximation to a collection of matrices that are distributed across peers, is not straightforward. However, as shown on the left-hand side of Fig. 2, use of binary non-orthogonal matrix decompositions makes it possible to use DHTs effectively to perform this task efficiently, providing high levels of accuracy. In this section, we present an algorithm that achieves this task by integrating binary matrix decompositions with a DHT implementation based on a representation of the logical space of peers as a binary hypercube. The main contributions of this paper are highlighted (shaded) in Fig. 2.

In pMINER, we use a variant of Kademlia to organize nodes into a semantic overlay, in which peer identifiers and hash keys are represented as binary vectors. The logical space of peers in Kademlia is shown in Fig. 3. In the figure, each leaf corresponds to a peer with the specified identifier. As in the hypercube interconnect [21], messages are routed by *fixing* a bit of the identifier at each hop. For example, when a message is being routed from peer 0010 to 1011, 0010 fixes the first unmatched bit between the source and target identifiers, and sends its message to peer 1010, which is a logical neighbor in the overlay, as shown in the figure. Then, peer 1010 fixes the next bit and routes the message to its final destination, 1011. The distance between any two peers in the overlay is specified by the `XOR` of their identifiers, which is indeed equal to the Hamming distance between them. Since hash keys in the DHT implementation are also binary vectors, this provides an efficient way for routing key-item pairs to their destination.

In pMINER, every peer has its own binary matrix A_i of size $m_i \times n$. Each row of this matrix corresponds to a document hosted by the peer. Our approach for computing a global

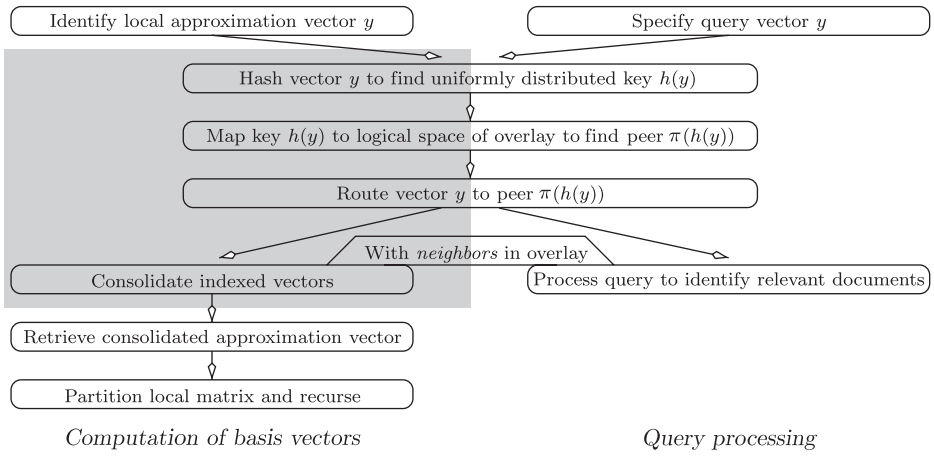

Fig. 2. The flow of basis computation and query processing in pMiner. This chart illustrates that the indexing mechanism used for processing queries also allows computation of basis vectors, which in turn is necessary to effectively process semantic queries. In this paper, we focus on efficient computation of basis vectors in a P2P system, as highlighted by the shaded region.

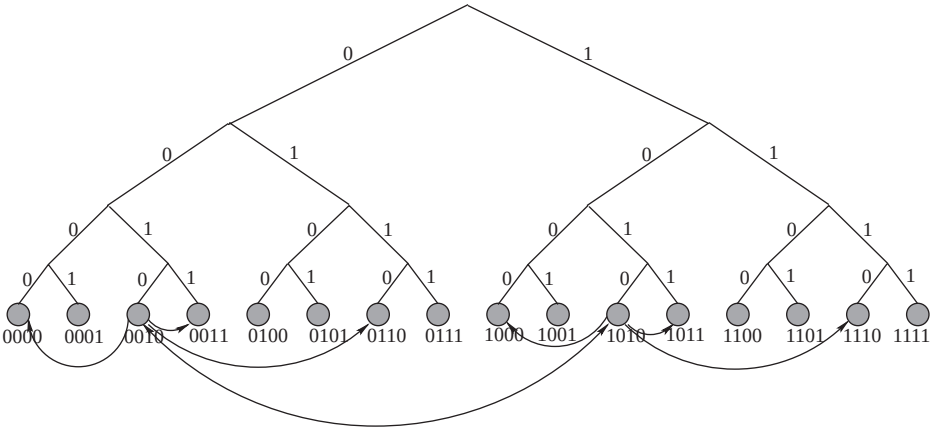

Fig. 3. Peer groups of nodes 0010 and 1010: arrows point to the computation neighbors of peers 0010 and 1010. The leaves of the tree represent the peers. The internal nodes are used to better illustrate the embedded tree in the id space.

approximation to the matrix

$$A = \begin{bmatrix} A_0 \\ A_1 \\ \cdots \\ A_{p-1} \end{bmatrix} \tag{3}$$

is based on a coarse-grained distributed formulation, in which each peer first computes its local rank-one approximation $A_i \approx x_i y_i^T$. Then, as in Conquest, the peers cooperate to consolidate their approximation vectors, so that patterns that are common to several pairs will be identified. To avoid broadcast of approximation vectors, which is expensive in a P2P network, we use a DHT implementation specified by Kademlia. The idea is to index similar approximation vectors (i.e., those with small Hamming distance) close to each other in the overlay, so that consolidation can be performed by cooperating only with neighbors in the overlay. For this reason, for each pattern vector y, we compute a hash key $h(y) \in \{0,1\}^r$, which is an r-dimensional binary string, satisfying the following properties:

(1) *Accuracy*: The hash function is *proximity preserving*, i.e., for any $y, y' \in \{0,1\}^n$, $d(h(y), h(y')) < d(y, y')$. This property ensures that pattern vectors that are of at most ε Hamming distance from each other are indexed at peers that are of at most ε hops away from each other in the overlay. Consequently, if the bound on Hamming radius is ε, then it is possible to identify pattern vectors that can be merged with each other by consolidating with only peers that are in the ε-neighborhood of each peer.
(2) *Load balance*: The hash function $h(y)$ is *uniformly distributed*, i.e., for any $b \in \{0,1\}^r$ and $y \in \{0,1\}^n$, $P[h(y) = b] \approx \frac{1}{2^r}$. This property ensures that the expected number of pattern vectors indexed at each peer is approximately equal to that at other peers in the network, balancing the load across peers.

We now propose a hash function that generates an r-dimensional binary key for an n-dimensional binary vector, which satisfies these two properties.

3.3.1. Key computation and placement

In most applications, the underlying data matrix is highly sparse and the distribution of terms in documents is skewed. Consequently, the approximation vectors are sparse and the distribution of non-zeros in entries is non-uniform. To deal with the uneven distribution of non-zeros, we propose a proximity preserving hash function that takes into account the distribution of non-zeros in the underlying data.

Let $y = [y(0)\ y(1)\ \ldots\ y(n-1)]^T$ be an n-dimensional binary vector. The hash function $h(y)$ computes an r-dimensional binary vector $b = h(y) = [b(0)\ b(1)\ \cdots\ b(r-1)]^T$. This hash function aggregates multiple bits in y into a single bit in b. To ensure uniform distribution, these bits are chosen adaptively such that the probability that the aggregated bit is set will be as close as possible to $\frac{1}{2}$, i.e., $P[b(i) = 0] \approx \frac{1}{2}$ for each $1 \leqslant i \leqslant r$. Without loss of generality, the aggregated bits are chosen to be consecutive.

Let s_i be the number of bits in y that are aggregated into the ith bit of b and f_i be the index of the first of these bits. Then each bit of the hash function $b = h(y)$ is defined as follows:

$$b(i) = \begin{cases} 0 & \text{if } \sum_{j=f_i}^{f_{i+1}-1} y(j) = 0; \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

for $1 \leqslant i \leqslant r$, where

$$\begin{aligned} &f_0 = 0, \\ &f_i = f_{i-1} + s_{i-1} \quad \text{for } 1 \leqslant i < r. \end{aligned} \tag{5}$$

The ith bit of $h(y)$ is therefore equal to the OR of s_i bits in y, starting from the f_ith bit. Consequently, the choice of s_i, for $1 \leqslant i \leqslant r$, is critical to ensure uniform distribution of the keys.

Let the relative frequency of the term j in the collection of documents be $q_j = \|A^T(j)\|^2/m$, where $A^T(j)$ denotes the jth column of matrix A. This simple statistic, which provides a reliable estimate of the probability that a given term exists in an arbitrary document, can be estimated and maintained using available procedures discussed in the previous section [17].

In pMiner, uniformity of the keys is achieved by choosing s_i as follows:

$$s_i = \underset{0 \leqslant \ell < n}{\operatorname{argmin}} \left| \frac{1}{2} - \prod_{j=f_{i-1}}^{f_{i-1}+\ell-1} (1 - q_j) \right|. \tag{6}$$

In other words, s_i specifies the size of the set of consecutive bits in y, starting from f_i, which minimizes the difference between $\frac{1}{2}$ and the probability that all of the bits in the set will be zero. This ensures that all binary vectors of length r are equally likely. The shortcoming here is the possibility that either: (i) $s_i = 0$ for $i \geqslant j$ for some j, where $\sum_{i=0}^{j} s_i = n$, or (ii) $\sum_{i=0}^{r-1} s_i < n$. Case (i) corresponds to the situation where there are more bits than necessary in the key space, which is easily alleviated, as we discuss below. Case (ii), on the other hand, means that the number of dimensions in the key space is not sufficient to capture the variance in the semantic space. This case, however, is unlikely since the documents are highly sparse. Moreover, it is possible to choose r adaptively to alleviate this problem.

Once vectors in semantic space are hashed to the r-dimensional binary space, it is necessary to further map the keys to peers. In pMiner, the prefix of a key is used to determine the peer in which it is indexed. We denote the identifier of this peer as $\pi(h(y))$. Observe that this choice implicitly handles case (i) above by disregarding the s_i's that are equal to zero. Solution to case (ii) requires that there are enough peers to capture the dimensionality of documents in the network, which is true for most large scale systems.

We now show that the hash function $h(y)$ preserves proximity between binary vectors, which is required for accurately consolidating patterns.

Theorem 1. *The hash function* h: $\{0, 1\}^n \to \{0, 1\}^r$, *given by Eqs.* (4)–(6), *is proximity preserving. In other words, for any* $y, y' \in \{0, 1\}^n$ $d(h(y), h(y')) \leqslant d(y, y')$.

Proof. For arbitrary i, $0 \leqslant i < r$, let $b(i)$ and $b'(i)$ be the ith bit of $h(y)$ and $h(y')$, respectively. Let $z_i = [y(f_i)\ y(f_i + 1)\ \ldots\ y(f_i + s_i)]$ and $z'_i = [y'(f_i)\ y'(f_i + 1)\ \ldots\ y'(f_i + s_i)]$ be the s_i-dimensional binary vectors that generate $b(i)$ and $b'(i)$, respectively. Clearly, $d(b(i), b'(i)) \leqslant 1$. Moreover, if $d(z_i, z'_i) = 0$, then $y(j) = y'(j)$ for $f_i \leqslant j \leqslant f_i + s_i$, so we have $d(b(i), b'(i)) = 0$. Hence, $d(b(i), b'(i)) \leqslant h(z_i, z'_i)$ always holds. Therefore, $d(h(y), h(y')) = \sum_{i=0}^{r-1} d(b(i), b'(i)) \leqslant \sum_{i=0}^{r-1} d(z_i, z'_i) = d(y, y')$. □

Note that $h(y)$ does not necessarily preserve distance, i.e., there is no guarantee that patterns that are distant from each other will be indexed to distant peers. However, for our purposes, this case is not relevant, since consolidation is performed on the pattern vectors, not keys. Furthermore, uniformity of distribution of keys ensures that, even though a peer may be responsible for consolidating multiple unrelated patterns, this load is distributed evenly throughout the network.

3.3.2. Pattern consolidation

The actions performed by peers in pMiner can be divided into three parts: (i) group formation; (ii) local matrix approximation and partitioning, and (iii) consolidation within the neighborhood. Group formation is performed when peers join the network, in order to recognize the peers that will cooperate to consolidate pattern vectors. Upon joining the network, a peer uses its identifier to generate a set of keys whose values are at Hamming distance one from its identifier. It then routes these keys in the DHT and requests all peers responsible for the keys to recognize itself as their immediate neighbor in the overlay. These immediate neighbors form the *computation group* for that particular peer. In, Fig. 3 the neighboring peers of peers 0010 and 1010 are shown. Observe that, other than itself, the neighbors of 0010 are all at Hamming distance two from peer 1010.

The peer with ID $\pi(h(y))$ is the *rendezvous point* (target of the distributed hash) for pattern vector y. Observe that, once all peers with pattern vector y route their pattern to peer $\pi(h(y))$, this peer has all the information about the patterns that are at

procedure CONSOLIDATE
receive $y_1, y_2, ..., y_\ell$ and $\sigma_1, \sigma_2, ..., \sigma_\ell$, such that $\pi(h(y_j)) = i$ for $1 \le j \le \ell$
build matrix B that contains σ_j rows equal to y_j for $1 \le j \le \ell$
for $round = 1$ **to** ϵ **do**
decompose B into X and Y such that $XY^T \approx B$ using PROXIMUS
send all rows $Y(j)$ of Y and $e_1 X^T(j)$ to communication group
receive $\hat{y}_1, \hat{y}_2, ..., \hat{y}_k$ and $\hat{\sigma}_1, \hat{\sigma}_2, ..., \hat{\sigma}_k$ from communication group
concatenate XY^T and $\sigma_j\ \hat{y}_j^T$s for $1 \le j \le k$ to obtain B
for $j = 1$ **to** ℓ **do**
send the row of Y that is of minimum distance from y_j to peer p_j

Fig. 4. Consolidation algorithm executed by all peers. Note that in practice, all σ_i rows are not replicated but they are accounted for implicitly.

distance zero from y. Therefore, if the bound on Hamming radius, ε, is equal to zero, then this peer can safely consolidate the patterns by itself, without communicating with any other peer in the network. This ensures that all pattern vectors that are within the specified radius (zero) are considered for being merged. Now consider the case $\varepsilon = 1$. Then, by Theorem 1, all pattern vectors that could potentially be merged with y reside either at peer $\pi(h(y))$ or its immediate neighbors in the overlay. Generalizing this observation, we conclude that, for given bound on Hamming radius, ε, all pattern vectors that need to be considered for consolidation with y reside at most ε hops from peer $\pi(h(y))$.

The consolidation phase consists of running PROXIMUS on the matrix composed of pattern vectors that are indexed within the computation group of a peer. This allows consolidation of the peer's pattern vectors with those that differ by at most one bit from these vectors. This procedure is repeated in rounds. In each round, pattern vectors that potentially differ by one more bit are also propagated. Therefore, at the ith round, a pattern vector is consolidated with all pattern vectors in the network that are at most at Hamming distance i from itself. Consequently, if the bound on Hamming radius is set to ε, consolidation is performed in ε rounds.

The steps a peer p_i executes in PMINER can be summarized as follows:

(1) Peer i computes a rank-one approximation of its local matrix A_i, producing a presence vector x_i and a pattern vector y_i.
(2) Peer i routes y_i and the number of ones in x_i, $\sigma_i = e_1^T x_i$, to $\pi(h(y_i))$. The number of ones in the presence vector indicates the *strength* of y_i in A_i, i.e., how many rows it approximates. This information is necessary to accurately weigh the strengths of pattern vectors in the global matrix A.
(3) Peer i receives all pattern vectors y_j such that $\pi(h(y_j)) = i$. It runs the consolidation procedure shown in Fig. 4, and routes the consolidated pattern vectors back to their owners.
(4) Upon receiving its own consolidated pattern vector, peer i partitions the rows of A_i, using the presence vector x_i and the consolidated pattern $\hat{y}_i$ corresponding to y_i. It then recursively decomposes the two resulting matrices, as shown in Fig. 1.

We illustrate pattern consolidation using a simple example. Let ε, the bound on Hamming radius, be set to one. Assume that a peer r, with identifier 01110111, received the following patterns to be consolidated:

$$\begin{bmatrix} y_1 \mid 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1 \\ y_2 \mid 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \\ y_3 \mid 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \\ y_4 \mid 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1 \end{bmatrix}$$

Each row in the matrix represents a pattern discovered by some peer in the network. Observe that r is the rendezvous point for y_1 and that $y_2 \ldots y_4$ were sent to r by its computation neighbors. The rank-one approximation of the matrix produces the vectors $x = [1\ 1\ 1\ 1]$ and $y = [0\ 1\ 1\ 1\ 0\ 0\ 1\ 1]$. The original matrix cannot be represented by these two vectors only and needs to be partitioned, since y is at Hamming distance two from y_4 and ε is set to one. After a new partition, the resulting patterns are:

$$\begin{bmatrix} 0\ 1\ 1\ 1\ 0\ 0\ 1\ 1 \\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 1 \end{bmatrix}.$$

In this case, peer r returns the first row of the matrix as the consolidated pattern for y_1. One point not considered in the above discussion is the strength of the patterns. The example assumes that all patterns have the same strength. If we assume that y_1 comes from a larger matrix and its strength is larger than the sum of the strengths of y_2 and y_3, the consolidation procedure would return the following pattern:

$$\begin{bmatrix} 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1 \end{bmatrix}.$$

Observe that this last pattern is at Hamming distance one from all patterns in the original matrix. Pattern strengths are not accounted for in CONQUEST. In Section 4, we show that one of the reasons PMINER achieves better results than CONQUEST is the inclusion of pattern strengths in the consolidation phase.

4. Experimental results

In this section, we present experimental results to investigate various performance aspects of PMINER. The main goal of the experiments is to compare results obtained by PMINER with the results obtained by PROXIMUS (serial implementation). More specifically, we investigate aspects of performance and quality of results, such as speedup, precision, recall, and compression ratio. We also investigate system parameters exclusive to PMINER, such as load balancing and messages processed per node. A qualitative comparison of PMINER with CONQUEST is also presented.

4.1. System setup and datasets

PMINER uses a modified version of Bamboo [26,32] as its underlying DHT. As discussed in Section 3, a key k in PMINER is installed at the node whose id shares the longest prefix with k. Bamboo, on the other hand, installs a key k at a node that is numerically closest to k. We modify Bamboo to take into account PMINER's key placement. The rank-one approximation and partition of matrices are implemented in C. The experiments are performed on Purdue/CERIASs Reassure testbed [5],

Table 1
Parameters used to generate the synthetic traces

Number of transactions (-ntrans)	3,000,000
Average items per transaction (-tlen)	10
Number of different items (-nitems)	100,000
Average length of maximal pattern (-patlen)	10
Correlation between patterns (-corr)	0.1–0.5
Average confidence in a rule (-conf)	0.6–0.9
Number of patterns (-npats)	2500–4000

Please, refer to [13] for a description of the generator and its parameters.

Table 2
PROXIMUS's parameters used in the serial experiments

Algorithm for rank-one approximation (-a)	Discrete (1)
Initialization strategy (-i)	Partition (4)
Minimum size of a cluster (-c)	1
Bound on normalized hamming radius (-e)	1–7

For more details about these parameters, please, refer to [19].

a cluster of 18 dual Athlon PCs networked over a Gbit Ethernet switch. This setup allows us to investigate more precisely, metrics of parallel performance, such as speedup. We run 15 peers on each machine for a total of 270 peers. Unless otherwise stated, we use this many peers for the experiments, the exception is the experiment that investigates load balancing.

We use two different types of data sets in the experiments, real-world data and synthetically generated data. The first data set is a database of customer's transactions (suitably anonymized) from Walmart. The database is translated into a matrix with 34,239 columns (items) and 974,158 rows (transactions). The matrix is very sparse, with an average number of items per transaction equal to 6.3. The second set of inputs is generated using the synthetic data generator developed by the IBM Quest Research Group [13]. We generate multiple synthetic data sets by varying the number of underlying patterns, correlation between patterns, and confidence of a pattern. The average number of items per transaction is set to 10. The average correlation between pairs of patterns is varied from 0.1 to 0.5 and the confidence of a pattern is varied from 0.6 to 0.9. Since the results obtained when the average correlation is varied does not present significant differences, we show only the results for a correlation of 0.1. Table 1 summarizes the parameters used in different synthetic data sets. These choices are by no means exhaustive, but they cover scenarios observed to be representative of general performance results; similar parameters are used in [6].

For the serial results, we run PROXIMUS with the parameters described in Table 2. We use the discrete rank-one approximation, the partition initialization scheme, and we vary the bounds on the Hamming radius to investigate the levels of compression, precision, and recall.

4.2. Quality of results

The main idea of PROXIMUS, and consequently PMINER, is to construct an approximation $\tilde{A} = XY^T$ of a matrix A, such that $\|A - \tilde{A}\|_F^2$ is minimized. X and Y are matrices representing the presence and pattern vectors. A natural measure of the quality of the approximation is the error in the approximation. The error is defined as the number of ones in the residual matrix, given by the following expression:

$$error = \|A - \tilde{A}\|_F^2 = |\{a_{ij} \in (A - \tilde{A}) : |a_{ij} = 1|\}|.$$

Since the error depends on the size of the input matrix, we use normalized metrics to evaluate the quality of the results. In data mining, *precision* and *recall* are well accepted metrics for evaluating search strategies. Precision is the ratio of the number of relevant records retrieved to the total number of records (relevant and irrelevant) retrieved. Recall is the ratio of the number of relevant records retrieved to the total number of relevant records present in a data repository. In our context, we define precision as the percentage of ones in the approximation matrix that are also present in the original matrix. Precision is given by the following expression:

$$precision = \frac{\|A\&\tilde{A}\|_F^2}{\|\tilde{A}\|_F^2} = \frac{|\{(i,j) : A(i,j) = \tilde{A}(i,j) = 1\}|}{|\{(i,j) : \tilde{A}(i,j) = 1\}|}.$$

We define recall as the percentage of ones in the original matrix that are also present in the approximation matrix. Recall is given by the following expression:

$$recall = \frac{\|A\&\tilde{A}\|_F^2}{\|A\|_F^2} = \frac{|\{(i,j) : A(i,j) = \tilde{A}(i,j) = 1\}|}{|\{(i,j) : A(i,j) = 1\}|}.$$

Fig. 5 shows the results for precision and recall for the Walmart data set as the bound on the Hamming radius is varied. As can be noted, PMINER achieves precision results slightly lower than PROXIMUS. The differences can be considered negligible, with the results of PMINER all well above 90%. Recall, on the other hand, is slightly better in PMINER. This small improvement in recall can be explained by the extra patterns identified by PMINER. Since, PMINER discovers more patterns than PROXIMUS, some patterns that are compressed, and consequently altered, in PROXIMUS, are left unchanged in PMINER.

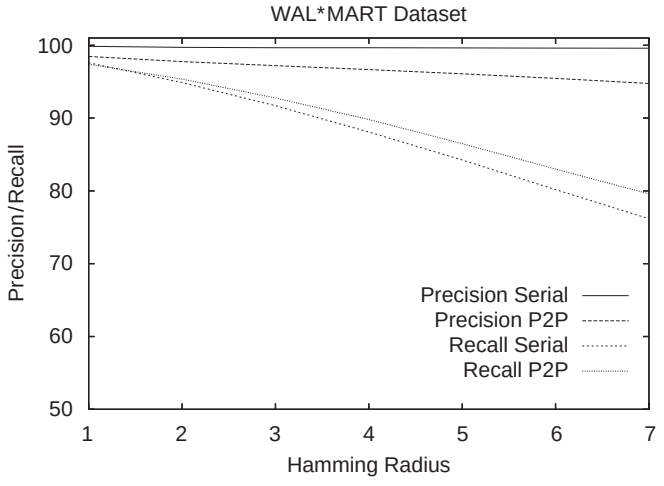

Fig. 5. Precision and recall for the Walmart data set.

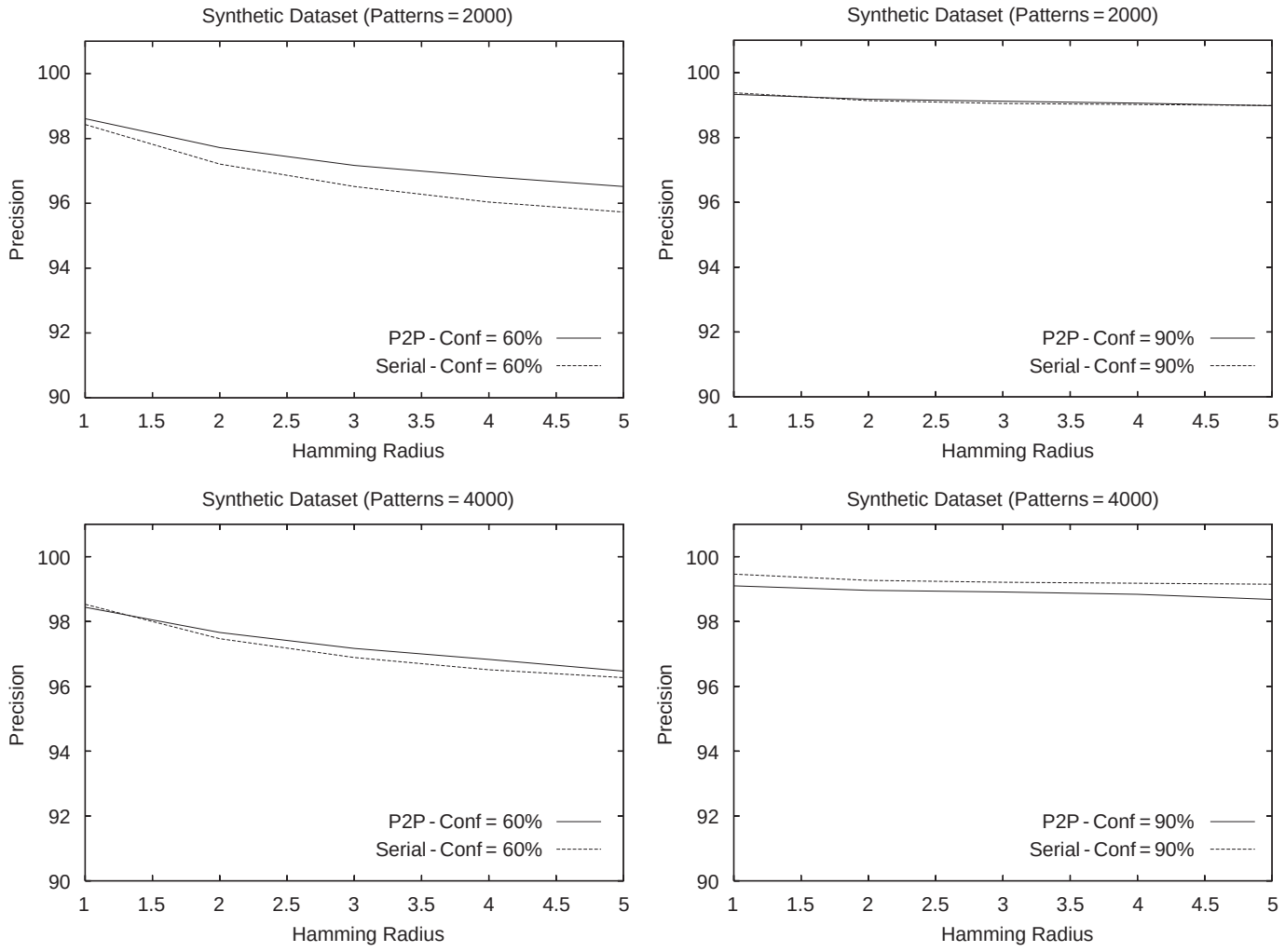

Fig. 6. Precision of various schemes for the synthetic data set.

In this particular data set, pMINER is able to better capture the entries of the original matrix (better recall), while inserting few extra ones (worse precision).

Fig. 6 shows the precision results for the synthetic data sets. We show results for two values of underlying patterns (2000 and 4000) and two values of pattern confidence (60% and 90%). Other parameter values show similar behavior. In these cases, the differences again can be considered negligible, with pMINER performing better in one situation (2000 and 60%) and PROXIMUS performing better in other situation (4000 and 90%). The other two cases do not show a clear difference.

Fig. 7 shows the recall results. The parameters used are the same as in the previous case. In these experiments, we observe the same behavior from the two approaches as the one observed in the case of precision.

4.3. Compression

The main goal of pMINER is to produce a small set of vectors that captures the dominant patterns present in the input matrix. Since the size of real data sets is expected to be large, significant levels of compression are required for a distributed solution to be useful. pMINER achieves between 61% and 80% compression for the synthetic data sets, and between 39% and 65% compression for the Walmart data set. Figs. 8 and 9 show the compression ratios achieved by pMINER and by PROXIMUS. As we can see, pMINER performs slightly worse than a centralized and serial implementation. In the worst case, pMINER finds 43.98% more patterns than the serial implementation for the synthetic data sets and 36.23% more patterns for the Walmart data set. These results are considerably better than the ones produced by CONQUEST. CONQUEST, in the worst case, discovers as many as 2.5 times more patterns than the serial implementation [6]. Moreover, the loss in compression does not affect the quality of the results, since the levels of precision and recall achieved by pMINER are similar to, and in some cases better than, the serial implementation. The loss in compression is a consequence of the design of the consolidation algorithm that tries to reduce the communication overhead as much as possible. In the consolidation phase, peers exchange just the rank-one approximation to consolidate local patterns with the global matrix.

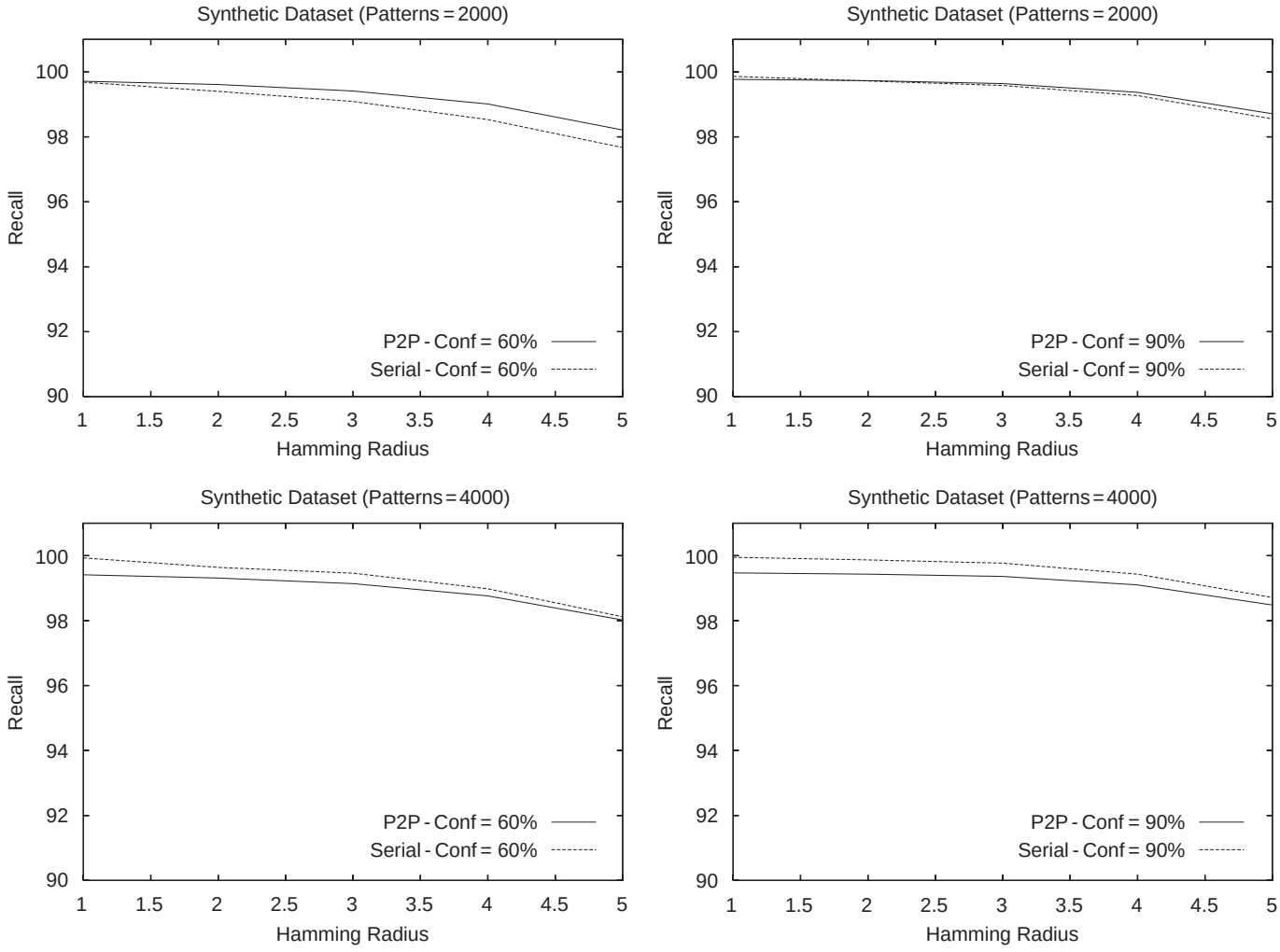

Fig. 7. Recall of various schemes for the synthetic data set.

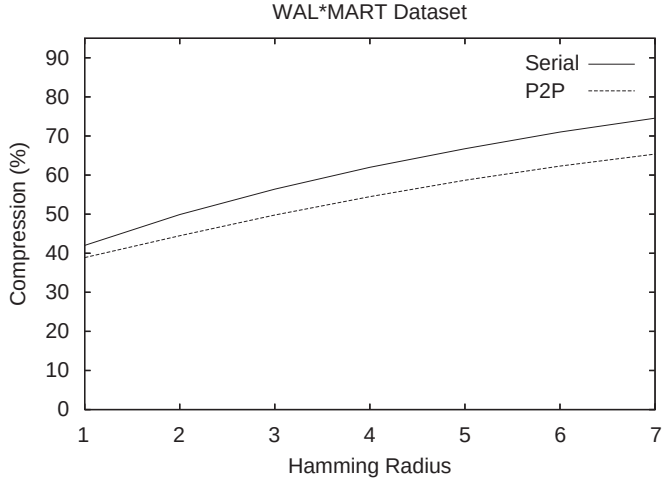

Fig. 8. Compression ratios achieved by PMINER and by PROXIMUS on the Walmart data set.

4.4. Speedup

As discussed in the previous section, PMINER produces results with levels of accuracy similar to PROXIMUS. We now discuss the parallel performance of PMINER. A measure of quality of a parallel/distributed solution is the (fractional) reduction in execution time (*speedup*) that we obtain using a parallel/distributed solution. The speedup from PMINER varies significantly depending on the input parameters. Tables 3 and 4 show the speedup results for the two types of data sets. The best speedup results are achieved when the Hamming radius is set to small values and when the compression is low (the number of patterns increases). In some cases, we observe super-linear speedup. As discussed in [6], super-linear speedup is achieved because of the effects of subsampling. The amount of computation performed by the parallel solution is not the same as its serial counterpart. The side effect of this reduced computation is an increase in the number of patterns discovered, as discussed in Section 4.3. The speedups from PMINER reduce as the Hamming radius increases. This reduction can be attributed to two factors. First,

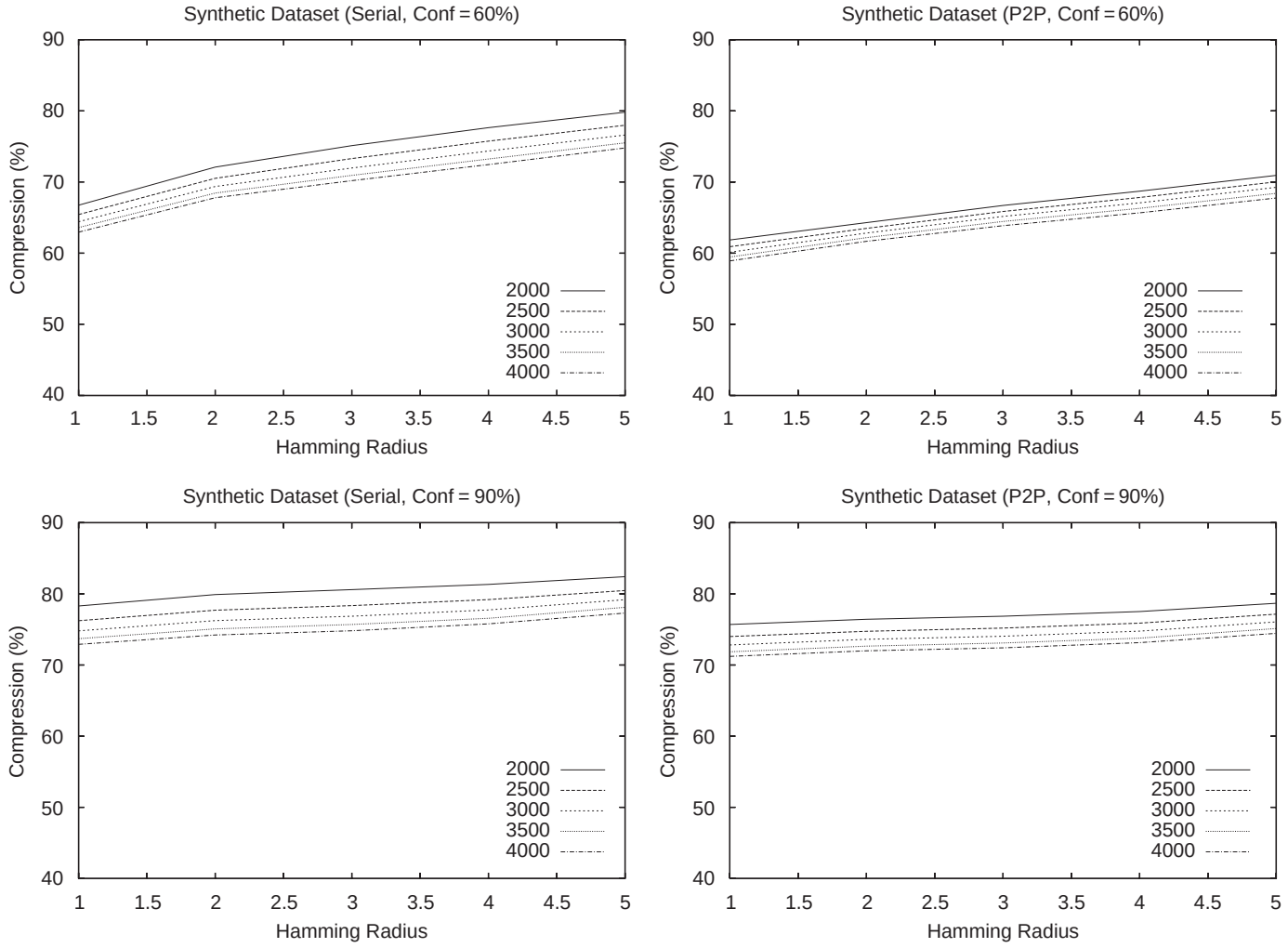

Fig. 9. Compression ratios achieved by pMiner and by Proximus on the synthetic data sets. Figures on the left show the results for Proximus and figures on the right show results for pMiner. The figures on top represent data sets generated with 60% of pattern confidence, and the figures at the bottom represent data sets generated with 90% of pattern confidence.

Table 3
Speedup results for the Walmart data set

Hamming radius	Serial time	P2P time	Speedup
1	61224.75	1075.71	56.92
2	59878.09	1792.97	33.40
3	55066.68	2453.70	22.44
4	51740.84	3438.02	15.05
5	49852.31	3486.22	14.30
6	48495.11	3989.33	12.16
7	47750.87	4555.83	10.48

The times above are expressed in seconds.

as the Hamming radius increases, pMiner spends more time synchronizing the peers. Each peer must synchronize with its neighbors before advancing to a new round of computation. Second, with the increase in the Hamming radius, Proximus spends significantly less time to converge to the final solution.

4.5. Load balancing

We now evaluate the ability of the hash function defined in Section 3.3.1 to balance the load among the peers. Given a set of K keys and N nodes, the expected number of keys per node is given by K/N. We adopt the definition of load balancing from Chord [29] and say that the load is balanced if each node receives at most $(1+\varepsilon)K/N$ keys, with $\varepsilon = O(\log N)$. Fig. 10 shows the number of messages each node receives. The middle line represents the average number of messages (K/N) and the top line represents $K/N \log N$. We observe an uneven distribution of keys per node in the left graphic of Fig. 10. This uneven distribution, however, is not severe, with all nodes receiving a number of keys well below the defined threshold for balanced load. We can better distribute the keys by utilizing virtual peers [29]. With virtual peers, each peer joins the network with multiple ids, i.e., it covers multiple ranges of the id space. The right graphic of Fig. 10 shows the results when each peer is responsible for two virtual peers. As

Table 4
Speedup for the synthetic data sets

HR	P2P time/speedup				
	2000	2500	3000	3500	4000
1	1562.8/27.61	1883.1/24.81	2236.1/21.89	2479.9/20.73	2650.6/20.15
2	2510.1/12.61	2974.7/11.91	3453.1/11.03	2878.3/14.08	4300.5/9.87
3	3466.6/7.52	4245.9/7.21	4760.7/7.06	5728.9/6.16	5959.4/6.40
4	4482.4/4.83	5223.5/4.96	5913.6/4.77	6770.4/4.56	7427.9/4.41
5	5599.4/3.23	6241.8/3.45	7046.0/3.42	8524.6/3.11	8833.6/3.20

The values 2000 to 4000 represent the number of underlying patterns in the data sets. The serial times (not shown) ranged from 18069.7 to 53385.3 s, and the P2P times ranged from 1562.8 to 8833.6 s.

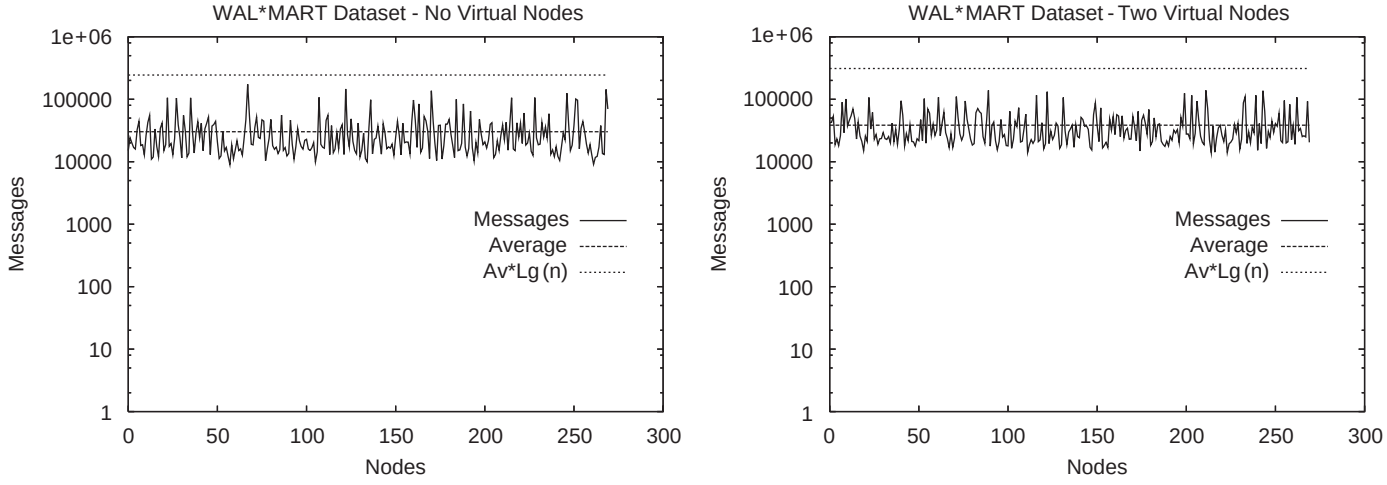

Fig. 10. Effect of virtual nodes on load balancing. The left figure shows the messages processed per peer when no virtual nodes are used. The right figure shows the results when two virtual nodes are used.

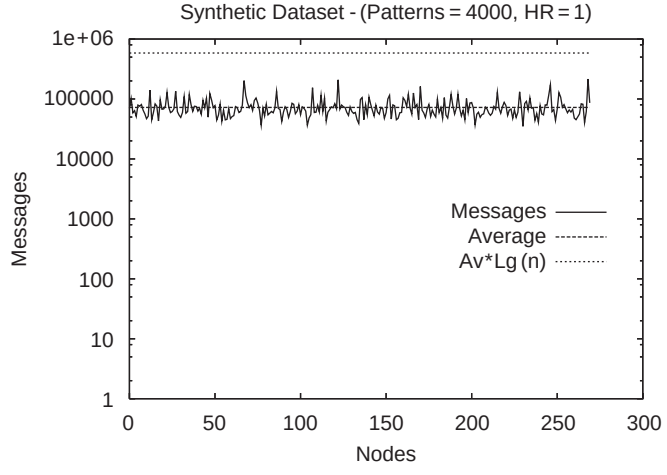

Fig. 11. Messages processed per node for the synthetic data set.

we can see, the curve is smoothed out considerably, with the maximum number of messages in a node smaller than half of the threshold.

Fig. 11 shows the number of messages processed per node for the synthetic data set. We show only one example here; other parameters produce similar results. In this case, the load is better distributed, even without the use of virtual nodes.

5. Conclusion

In this paper, we present a framework, pMINER for performing semantic queries in structured peer-to-peer (P2P) networks, using binary non-orthogonal matrix decompositions. We present algorithms for efficient computation of latent semantic vectors and associated search. We demonstrate excellent performance of our scheme in terms of precision and recall, resource requirement (load balancing), as well as speedups, in the context of real applications as well as synthetic workloads.

Acknowledgment

The authors would like to acknowledge Prof. Vipin Kumar for his significant suggestions and comments on this work. This work is funded by the National Science Foundation grant CNS 0509387.

References

[1] E. Anceaume, A.K. Datta, M. Gradinariu, G. Simon, A. Virgillito, A semantic overlay for self-* peer-to-peer publish/subscribe, in: Proceedings of the 26th IEEE International Conference on Peer-to-Peer Systems (ICDCS'06), Lisbon, Portugal, 2006, pp. 22–30.
[2] M.W. Berry, Z. Drmaic, E.R. Jessup, Matrices, vector spaces, and information retrieval, SIAM Rev. 41 (2) (1999) 335–362.
[3] BitTorrent. URL http://www.bittorrent.com/.

[4] M. Castro, P. Druschel, A.-M. Kermarrec, A. Rowstron, SCRIBE: a large-scale and decentralized application-level multicast infrastructure, IEEE Journal of Selected Areas Commun. (JSAC) 20 (8) (2002) 447–461.
[5] CERIAS's Reassure. URL ⟨http://www.reassure.cerias.purdue.edu/⟩.
[6] J. Chi, M. Koyutürk, A. Grama, Conquest: a coarse-grained algorithm for constructing summaries of distributed discrete datasets, Algorithmica 45 (3) (2006) 377–401.
[7] L. Cox, C. Murray, B. Noble, Pastiche: making backup cheap and easy, in: Proceedings of the Fifth Symposium on Operating System Design and Implementation (OSDI '02), Boston, MA, 2002.
[8] R. Cox, A. Muthitacharoen, R.T. Morris, Serving DNS using a peer-to-peer lookup service, in: Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, MA, 2002, pp. 155–165.
[9] F. Dabek, M. Kaashoek, D. Karger, R. Morris, I. Stoica, Wide-area cooperative storage with CFS, in: Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP), Lake Louise, Canada, 2001.
[10] S. Datta, K. Bhaduri, C. Giannella, H. Kargupta, R. Wolff, Distributed data mining in peer-to-peer networks, IEEE Internet Comput. 10 (4) (2006) 18–26.
[11] A. Goldberg, P. Yianilos, Towards an archival intermemory, in: Proceedings of IEEE Advances in Digital Libraries (ADL), 1998.
[12] G.H. Golub, C.V.V. Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, MD, 1996.
[13] IBM Almaden Research Center—Intelligent Information Systems. URL ⟨http://www.almaden.ibm.com/software/projects/iis/hdb/Projects/data_mining/mining.shtml⟩.
[14] S. Iyer, A. Rowstron, P. Druschel, SQUIRREL: a decentralized, peer-to-peer web cache, in: Proceedings of 21st ACM Symposium on Principles of Distributed Computing (PODC'02), Monterey, CA, 2002, pp. 213–222.
[15] D. Kempe, A. Dobra, J. Gehrke, Gossip-based computation of aggregate information, in: Proceedings of the 44th IEEE Symposium of Foundations of Computer Science (FOCS'03), Cambridge, MA, 2003, pp. 482–491.
[16] T.G. Kolda, D.P. O'Leary, Algorithm 805: computation and uses of the semidiscrete matrix decomposition, ACM Trans. Math. Software 26 (3) (2000) 415–435.
[17] W. Kowalczyk, M. Jelasity, A.E. Eiben, Towards data mining in large and fully distributed peer-to-peer overlay networks, in: Proceedings of BNAIC 2003, Nijmegen, Netherlands, 2003.
[18] M. Koyutürk, A. Grama, N. Ramakrishnan, Compression, clustering and pattern discovery in very high dimensional discrete-attribute datasets, IEEE Trans. Knowledge Data Eng. 17 (4) (2005) 447–461.
[19] M. Koyutürk, A. Grama, N. Ramakrishnan, Nonorthogonal decomposition of binary matrics for bounded-error data compression and analysis, ACM Trans. Math. Software 32 (1) (2006) 33–69.
[20] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao, OceanStore: an architecture for global-scale persistent storage, in: Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), Cambridge, MA, 2000.
[21] V. Kumar, A. Grama, A. Gupta, G. Karypis, Introduction to Parallel Computing: Design and Analysis of Algorithms, Benjamin-Cummings Publishing Co., Redwood City, CA, USA, 1994.
[22] M. Li, W.-C. Lee, A. Sivasubramaniam, Semantic small world: an overlay network for peer-to-peer search, in: Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04), Berlin, Germany, 2004.
[23] P. Maymounkov, D. Mazières, Kademlia: a peer-to-peer information system based on the XOR metric, in: Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, MA, 2002, pp. 53–65.
[24] C.R. Rao, The use and interpretation of principal components analysis in applied research, Sankhya Serie A 26 (1964) 329–358.
[25] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, in: Proceedings of the 2001 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, San Diego, CA, 2001, pp. 247–254.
[26] S. Rhea, D. Geels, T. Roscoe, J. Kubiatowicz, Handling churn in a DHT, in: Proceedings of the USENIX Annual Technical Conference, Boston, MA, 2004.
[27] A. Rowstron, P. Druschel, Pastry: scalable, , decentralized object location and routing for large-scale peer-to-peer systems, in: Proceedings of the 2001 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, San Diego, CA, 2001, pp. 247–254.
[28] A. Rowstron, P. Druschel, Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility, in: Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP), Lake Louise, Canada, 2001.
[29] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup service for internet applications, in: Proceedings of the 2001 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, San Diego, CA, 2001, pp. 149–160.
[30] P. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison Wesley, Reading, MA, 2005.
[31] C. Tang, Z. Xu, S. Dwarkadas, Peer-to-peer information retrieval using self-organizing semantic overlay networks, in: Proceedings of the 2003 ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Karlsruhe, Germany, 2003, pp. 175–186.
[32] M. Welsh, D. Culler, E. Brewer, SEDA: an architecture for highly concurrent server applications, in: Proceedings of the 18th Symposium on Operating Systems Principles (SOSP-18), Banff, Canada, 2001.
[33] R. Wolff, K. Bhaduri, H. Kargupta, Local L2-thresholding based data mining in peer-to-peer systems, in: Proceedings of the Sixth SIAM International Conference on Data Mining (SDM'06), Bethesda, MD, 2006.
[34] S. Zelikovitz, H. Hirsh, Using LSI for text classification in the presence of background text, in: Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM'01), ACM Press, New York, NY, USA, 2001, pp. 113–118.

Ronaldo Alves Ferreira received his Ph.D. from Purdue University in 2006. He is currently an Assistant Professor of Computer Science at Federal University of Mato Grosso do Sul, Brazil. His research interests are in networking and distributed systems.

Mehmet Koyuturk received his M.S. from Bilkent University in 2000 and Ph.D. form Purdue University in 2006. He is currently an Assistant Professor at the Electrical Engineering and Computer Science Department of Case Western Reserve University. Prior to joining CWRU in 2007, he was with the Computer Science Department of Purdue University as a post-doctoral researcher. His research interests include bioinformatics, data mining and knowledge discovery, and algorithms for distributed systems.

Suresh Jagannathan is a Professor of Computer Science at Purdue University. Prior to joining Purdue in 2002, he was a Senior Research Scientist at the NEC Research Institute. His interests are in programming languages and their implementation, especially with respect to concurrent and distributed systems. He received his M.S. and Ph.D. from MIT.

Ananth Grama is a Professor of Computer Science at Purdue University. He received his Ph.D. in Computer Science from the University of Minnesota. His primary areas of research include parallel and distributed computing, scientific computing, and large-scale data handling and analysis. On these topics, he has (co)authored several papers and textbooks. At Purdue, he is affiliated with the Coordinated Systems Lab.