# Hypergraph-based Multilevel Matrix Approximation for Text Information Retrieval*

Haw-ren Fang
Dept. of Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455, USA
hrfang@cs.umn.edu

Yousef Saad
Dept. of Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455, USA
saad@cs.umn.edu

## ABSTRACT

In Latent Semantic Indexing (LSI), a collection of documents is often pre-processed to form a sparse term-document matrix, followed by a computation of a low-rank approximation to the data matrix. A multilevel framework based on hypergraph coarsening is presented which exploits the hypergraph that is canonically associated with the sparse term-document matrix representing the data. The main goal is to reduce the cost of the matrix approximation without sacrificing accuracy. Because coarsening by multilevel hypergraph techniques is a form of clustering, the proposed approach can be regarded as a hybrid of factorization-based LSI and clustering-based LSI. Experimental results indicate that our method achieves good improvement of the retrieval performance at a reduced cost.

## Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Indexing methods; G.2.2 [**Graph Theory**]: Hypergraphs; F.2.1 [**Numerical Algorithms and Problems**]: Computations on matrices

## General Terms

Algorithms, Experimentation

## Keywords

Multilevel Hypergraph Partitioning, Low-rank Matrix Approximation, Latent Semantic Indexing, Text Information Retrieval

## 1. INTRODUCTION

Matrix approximation techniques appear in many fields, including data mining, machine learning, and computer vision, where a set of data is converted into numerical form as a matrix. The goal is to produce a low-rank matrix approximation, in order to extract the most important features of the data. On the other hand, there are instances such as in text mining when the original data itself is sparse. For such cases, there is a hypergraph canonically associated with the data. The hypergraph model combined with a multilevel approach, using coarsening among other tools, has had a remarkable success for scientific computing with sparse data, such as parallel sparse-matrix techniques [2, 16] and VLSI design [9, 10]. Motivated by these works, we explore multilevel matrix approximation methods for LSI, by exploiting hypergraph coarsening schemes.

The technique proposed in this paper recursively computes a coarsened version of the original hypergraph (i.e., one with fewer vertices) of the data set. This is performed with a method called *maximal-weight matching* which repeatedly merges pairs of vertices, e.g., [5]. The coarsened hypergraph is then processed with a term weighting scheme and the resulting matrix is approximated by a matrix factorization algorithm before the query matching is performed. We refer to the resulting method as *multilevel-LSI*.

The factorization performed at the last level can be any of the common factorizations used in the literature. Three such factorizations will be explored in this paper: the singular value decomposition (SVD), the semi-discrete decomposition (SDD) [11, 12], and the non-negative matrix factorization (NMF) [13, 14].

## 2. BACKGROUND

In the vector space model (VSM), a collection of $n$ documents indexed by $m$ terms is usually pre-processed to form an $m$-by-$n$ term-document matrix [1, 7]. In a simple form, the $(i, j)$-th entry of the matrix is the number of occurrences of term $i$ in document $j$, called term frequency. In information retrieval, a query is a column vector of size $m$ reflecting the relevance of the query to the $m$ terms. The objective is to retrieve the relevant documents from the collection.

Term weighting schemes are normally applied in order to enhance the retrieval effectiveness [3, 15]. We use $f_{ij}$ to denote the term frequency, the number of occurrences of term $i$ in document $j$, and $s(f_{ij})$ is a boolean set to one if term $i$ occurs in document $j$ ($f_{ij} > 0$), and to zero otherwise. The 'weighted' term-document matrix $A = [a_{ij}] \in \mathbb{R}^{m \times n}$ has $a_{ij}$ in the form $a_{ij} = t_{ij} g_i d_j$, representing some weight of term $i$ in document $j$. The three components $g_i$, $t_{ij}$, and $d_j$ stand for collection frequency (global term weight), term frequency (local term weight), and normalization, respectively. A term weighting scheme is represented by three letters. For exam-

*This work was supported by NSF grants DMS 0510131 and DMS 0528492 and by the Minnesota Supercomputing Institute.


*CIKM'10,* October 26–30, 2010, Toronto, Ontario, Canada.

ple, the classical TF-IDF (term frequency, inverse document frequency) weight with normalization is coded as `tfn`, corresponding to the formula $a_{ij} = f_{ij} \log(\frac{n}{\sum_j s(f_{ij})}) d_j$, where $d_j$ is chosen to make $\sum_{i=1}^m a_{ij} = 1$. The code `cfx` stands for the weighting scheme $a_{ij} = \frac{1}{2}(s(f_{ij}) + \frac{f_{ij}}{\max_k f_{kj}}) \log(\frac{n}{\sum_j s(f_{ij})})$.

A weighting scheme is also applied to a query, but the collection frequency component (global weight) is from the collection of documents. The overall weighting is specified by a six-letter string, three for document weights and three for query weights. In the experiments reported in Section 4, we used the `tfn.cfx` term weighting, which is an effective method for vector space model [15]. Some other term weighting formulas can be found in [15].

Query matching is the process of finding the documents in the collection relevant to a given query. The similarity of two vectors, a query $q \in \mathbb{R}^m$ and a document $a \in \mathbb{R}^m$, is measured by the cosine distance $q^T a/(\|q\|_2 \|a\|_2)$, the cosine of the acute angle between them.

Another approach for information retrieval is by hierarchical agglomerative clustering, which forms a binary tree with leaves associated with documents and internal nodes representing clusters. To be precise, a node represents the cluster containing all the leaves. The retrieval is performed by a bottom-up search, until a preset number of documents are retrieved, e.g., [8].

In text mining, common word usage may affect the retrieval performance of the vector space model. Two well-known issues are *synonymy* (two or more words with one meaning) and *polysemy* (one word with more than one meaning). The main assumption of Latent Semantic Indexing (LSI) is that there is an underlying latent semantic structure that represents the contextual meaning of words rather than their literal usage. LSI uses the singular value decomposition (SVD)[1] of the term-document matrix to extract this intrinsic, or latent, structure [4]. Attempts have been made to replace the SVD by other factorizations, such as semi-discrete decomposition (SDD) [11]. In either case, the approximation of the term-document matrix can be expressed in the form $A \approx A_d = U_d \Sigma_d V_d^T \in \mathbb{R}^{m \times n}$, where $\Sigma_d \in \mathbb{R}^{d \times d}$ is non-negative and diagonal, $U \in \mathbb{R}^{m \times d}$, $V \in \mathbb{R}^{n \times d}$, with $d$ a certain desired number of vectors to approximate $A$. For SVD, $U$ and $V$ consist of orthogonal columns. For SDD, the entries of $U$ and $V$ are discrete values from $\{-1, 0, 1\}$.

When the document vectors (columns in $A \in \mathbb{R}^{m \times n}$) and the query vector $q \in \mathbb{R}^m$ are normalized, or when the inner product measure instead of cosine distance measure is utilized, the similarity scores are the entries of $q^T A$ in the vector space model. Replacing $A$ by $A_d = U_d \Sigma_d V_d^T$,

$$q^T A_d = q^T U_d \Sigma_d V_d^T = (q^T U_d \Sigma_d^\alpha)(\Sigma_d^{1-\alpha} V_d^T) = \widehat{q}^T \widehat{A}, \quad (1)$$

where $\alpha$ is a scalar which controls the splitting of the matrix approximation [11]. The reduced representations of query $\widehat{q}$ and document vectors (columns in $\widehat{A}$) are defined by

$$\widehat{q} = \Sigma_d^\alpha U_d^T q \in \mathbb{R}^d, \qquad \widehat{A} = \Sigma_d^{1-\alpha} V_d^T \in \mathbb{R}^{d \times n}. \quad (2)$$

Assuming $U_d$ consists of linearly independent columns, $\widehat{A} \in \mathbb{R}^{d \times n}$ is a linear projection of $A_d \in \mathbb{R}^{m \times n}$:

$$\widehat{A} = \Sigma_d^{1-\alpha} V_d^T = \Sigma_d^{-\alpha} U_d^+ A_d, \quad (3)$$

[1]Strictly, we mean 'truncated' singular value decomposition. The word 'truncated' is omitted as there is no ambiguity.

where $U_d^+ \in \mathbb{R}^{d \times m}$ is the pseudo-inverse of $U_d \in \mathbb{R}^{m \times d}$. Another way to obtain the reduced representation of a query $q \in \mathbb{R}^m$ is by applying the projector $P = \Sigma_d^{-\alpha} U_d^+$ in (3) to $q$. The resulting formula is

$$\widehat{q} = \Sigma_d^{-\alpha} U_d^+ q \in \mathbb{R}^d, \qquad \widehat{A} = \Sigma_d^{1-\alpha} V_d^T \in \mathbb{R}^{d \times n}. \quad (4)$$

We call (4) the Type-I LSI and (2) the Type-II LSI. In both cases, one can optionally renormalize $\widehat{q}$ and the columns of $\widehat{A}$. Renormalization usually slightly improves the retrieval performance [11]. As in the case of the vector space model, a document is deemed relevant to a query if their similarity score is larger than some pre-defined threshold.

# 3. MULTILEVEL TECHNIQUES

Consider the concept decomposition [6] in the form $C_k \widehat{Z}_k$ from clustering the columns of $A \in \mathbb{R}^{m \times n}$, where $k$ is the number of clusters, each column of $C_k \in \mathbb{R}^{m \times k}$ is the centroid of a cluster, and $\widehat{Z}_k \in \mathbb{R}^{k \times n}$ is chosen to minimize $\|A - C_k \widehat{Z}_k\|_F^2$. One way to further reduce the dimensionality is to compute a low-rank approximation of $C_k$. Following the notation in Section 2, we denote this decomposition by $C_k \approx \widehat{U}_d \widehat{\Sigma}_d \widehat{V}_d^T$. Replacing $C_k$ by $\widehat{U}_d \widehat{\Sigma}_d \widehat{V}_d^T$, we minimize

$$\min_{\widehat{Z}_k \in \mathbb{R}^{k \times n}} \|A - \widehat{U}_d \widehat{\Sigma}_d \widehat{V}_d^T \widehat{Z}_k\|_F = \min_{Z_d \in \mathbb{R}^{d \times n}} \|A - \widehat{U}_d Z_d\|_F, \quad (5)$$

where we set $Z_d = \widehat{\Sigma}_d \widehat{V}_d^T \widehat{Z}_k$.

When $\widehat{U}_d$ is of full column rank, the minimizer of (5) is

$$Z_d = (\widehat{U}_d^T \widehat{U}_d)^{-1} \widehat{U}_d^T A = \widehat{U}_d^+ A,$$

where $\widehat{U}_d^+$ is the pseudo-inverse of $\widehat{U}_d$. If $\widehat{U}_d$ is rank-deficient, $Z_d = \widehat{U}_d^+ A$ is still a minimizer of (5) but not a unique one. The term-document matrix approximation is then

$$A \approx \widehat{U}_d Z_d = \widehat{U}_d \widehat{U}_d^+ A.$$

The similarity scores of a query $q$ to the documents of $A$ are computed by $q^T(\widehat{U}_d \widehat{U}_d^+ A) = (q^T \widehat{U}_d \widehat{\Sigma}_d^\alpha)(\widehat{\Sigma}_d^{-\alpha} \widehat{U}_d^+ A)$, from which we obtain the reduced representations of $q$ and $A$,

$$\widehat{q} = \widehat{\Sigma}_d^\alpha \widehat{U}_d^T q, \qquad \widehat{A} = \widehat{\Sigma}_d^{-\alpha} \widehat{U}_d^+ A. \quad (6)$$

Alternatively, we can apply the same linear transformation $\widehat{\Sigma}_d^{-\alpha} \widehat{U}_d^+$ to $q$ and obtain the formulas

$$\widehat{q} = \widehat{\Sigma}_d^{-\alpha} \widehat{U}_d^+ q, \qquad \widehat{A} = \widehat{\Sigma}_d^{-\alpha} \widehat{U}_d^+ A. \quad (7)$$

In both (6) and (7), one can optionally renormalize the projected document vectors for similarity score computations.

Two considerations of the hybrid approach presented here are as follows. First, the clustering result depends on the term weighting scheme applied to the term-document matrix. The centroid of documents scaled by term weighting loses some intrinsic information, specifically the count of occurrences of each term. Second, some clustering algorithms, such as the spherical $k$-means clustering, need the number of clusters $k$ in advance, which may not be easy to determine in practice. Also, changing the value of $k$ may require recomputing the approximation. We present a multilevel technique using hypergraphs for matrix approximation to address these two issues.

A hypergraph $H = (V, E)$ consists of a set of vertices $V$ and a set of hyperedges $E$. Each hyperedge, also called a *net*, is a non-empty subset of $V$; the size of this subset is called

the *degree* of this hyperedge. Likewise, the *degree* of a vertex is the number of hyperedges which include it. Two vertices are called *neighbors* if there is a hyperedge connecting them.

A hypergraph $H = (V, E)$ can be canonically represented by a boolean matrix $A$, where the vertices in $V$ and hyperedges (nets) in $E$ are represented by the columns and rows of $A$, respectively. This is called the *row-net model*. Each hyperedge, a row of $A$, connects the vertices whose corresponding entries in that row are non-zero. For example, $V = \{1, \ldots, 9\}$ and $E = \{a, \ldots, e\}$ with $a = \{1, 2, 3, 4\}$, $b = \{3, 5, 6, 7\}$, $c = \{4, 7, 8, 9\}$, $d = \{6, 7, 8\}$, and $e = \{2, 9\}$. The boolean matrix representation of this hypergraph is

$$\text{A} = \begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \\ \hline & 1 & 1 & 1 & 1 & & & & & & \text{a} \\ & & & 1 & & 1 & 1 & 1 & & & \text{b} \\ & & & & 1 & & & 1 & 1 & 1 & \text{c} \\ & & & & & & 1 & 1 & 1 & & \text{d} \\ & & 1 & & & & & & & 1 & \text{e} \end{array}$$

Coarsening a hypergraph $H = (V, E)$ means finding a 'coarse' approximation $\bar{H} = (\bar{V}, \bar{E})$ to $H$ with $|\bar{V}| < |V|$, which is a reduced representation of the original hypergraph $H$, in that it retains as much of the structure of the original hypergraph as possible. By recursively coarsening we obtain a succession of smaller hypergraphs which approximate the original graph. Several methods exist for coarsening hypergraphs [2, 10]. The method used in this paper is based on merging pairs of vertices.

In order to select which pairs of vertices to merge in a hypergraph, we consider the *maximum-weight matching* problem, e.g., [2, 5]. Pairing two vertices is termed *matching*. The edge weight between two vertices is the number of hyperedges connecting them. In the hyperedge-vertex matrix representation (a boolean matrix), the weight of the pair $i, j$ (vertices) is the inner product of the two columns $i$ and $j$.

This inner-product weight is adopted as a similarity metric in two software packages for hypergraph partitioning, hMETIS [9] and Mondriaan [16]. The maximum-weight matching problem consists of finding a matching that maximizes the sum of edge weights of the vertex pairs. In practice, it is not necessary to find the optimal solution, as suboptimal greedy approaches yield satisfactory results [5].

A greedy algorithm for maximum-weight matching is used in our experiments. For each unmatched vertex $v$, all the unmatched neighbor vertices $u$ are considered, and the inner product between $v$ and each $u$ is computed. The vertex $u$ with the highest non-zero inner product $\langle u, v \rangle$ is matched with $v$ and the procedure is repeated until all vertices have been matched. The computed matching is a coarse representation of the hypergraph, with the coarse hyperedges inherited from the fine hypergraph. More precisely, the coarse vertex set consists of matched pairs of fine vertices. A pair of fine vertices is in a coarse hyperedge if any of the two vertices is in the corresponding fine hyperedge. It is convenient to present the hypergraph coarsening procedure in matrix form as in Algorithm 1.

Given a sparse term-document matrix $A$, we can use Algorithm 1 to recursively coarsen the corresponding hypergraph in the row-net model level by level, and obtain a sequence of sparse matrices $A_1, A_2, \ldots, A_r$ with $A_1 = A$, where $A_i$ corresponds to the coarse graph $H_i$ of level $i$.

For the quality of the clusters, a matrix of term frequencies is usually preprocessed by a term weighting scheme before applying a clustering algorithm, such as the spherical $k$-means algorithm. However, the clustering result by multilevel hypergraph coarsening is independent of term weighting, since only the sparsity patterns of matrices are taken into account during hypergraph coarsening.

{Coarsen a hypergraph $A \in \mathbb{R}^{m \times n}$. }
$p := 0$ ▷ # vertices already in the coarsened graph
**repeat**
  $p := p + 1$
  Randomly pick $j \in S$; $S := S - \{j\}$.
  Set ip$[k] := 0$ for $k = 1, \ldots, n$.
  **for all** $i$ with $a_{ij} \neq 0$ **do**
    **for all** $k$ with $a_{ik} \neq 0$ **do**
      ip$[k]$ := ip$[k] + 1$
    **end for**
  **end for**
  $i :=$ argmax{ip$[k] : k \in S$}
  **if** ip$[i] = 0$ **then**
    $\bar{A}(:, p) := A(:, j)$
  **else**
    $\bar{A}(:, p) := A(:, i) + A(:, j)$
    $S := S - \{i\}$
  **end if**
**until** $S = \emptyset$
{The coarse hypergraph is represented by $\bar{A} \in \mathbb{R}^{m \times p}$.}

**Algorithm 1:** Hypergraph coarsening.

We take a matrix of term frequencies as input $A$. The coarsening result $A_r$ of the lowest level $r$ has entries still being term frequencies, since each column of $A_r$ is a sum of multiple columns in $A$. We then apply a term weighting scheme to $A_r$ to form $\widetilde{A}_r$, followed by a low-rank approximation $\widetilde{A}_r \approx \widetilde{U}_d \widetilde{\Sigma}_d \widetilde{V}_d^T = (\widetilde{U}_d \widetilde{\Sigma}_d^{\alpha})(\widetilde{\Sigma}_d^{1-\alpha} \widetilde{V}_d^T)$ by SVD, SDD, or NMF, where $d$ is the desired number of basis vectors, and $\alpha$ is the splitting scalar. Following the discussion leading to (6) and (7), we obtain the Type-I multilevel-LSI formula

$$\widehat{q} = \widetilde{\Sigma}_d^{-\alpha} \widetilde{U}_d^+ \widetilde{q} \in \mathbb{R}^d, \qquad \widehat{A} = \widetilde{\Sigma}_d^{-\alpha} \widetilde{U}_d^+ \widetilde{A} \in \mathbb{R}^{d \times n}, \quad (8)$$

and the Type-II multilevel-LSI formula

$$\widehat{q} = \widetilde{\Sigma}_d^{\alpha} \widetilde{U}_d^T \widetilde{q} \in \mathbb{R}^d, \qquad \widehat{A} = \widetilde{\Sigma}_d^{-\alpha} \widetilde{U}_d^+ \widetilde{A} \in \mathbb{R}^{d \times n}, \quad (9)$$

where a term weighting scheme has been applied to the query $\widetilde{q} \in \mathbb{R}^m$ and the term-document matrix $\widetilde{A} \in \mathbb{R}^{m \times n}$. For consistency, the global weights for $\widetilde{q}$ and $\widetilde{A}$, if any, follow those applied to $\widetilde{A}_r$.

Finally, the query matching is made by similarity scores between the query $\widehat{q}$ and documents as column vectors of $\widehat{A}$ in the reduced form. A key property of our multilevel-LSI is that the term weighting is applied posterior to multilevel hypergraph coarsening.

# 4. RETRIEVAL EXPERIMENTS

We used a Linux machine equipped with an Intel Core 2 Duo 2.4GHz CPU for our experiments. The codes are all in Matlab, including the public libraries SDD-PACK [12] and the NMF implementation [14]. We used the Lanczos algorithm with full reorthogonalization for SVD computation. The standard evaluation metric average precision is adopted to measure the retrieval performance.

Three public data sets were used in our experiments[2]: `Medline`, `Cranfield`, and `NPL`. The characteristics of these sets are listed in Table 1.

**Table 1: Characteristics of the test sets.**

| | `Medline` | `Cranfield` | `NPL` |
|---|---|---|---|
| # documents | 1033 | 1398 | 11429 |
| # terms | 3681 | 2331 | 4322 |
| # of terms/document | 48.36 | 52.11 | 19.68 |
| Sparsity (%) | 1.314% | 2.235% | 0.455% |
| # queries | 30 | 225 | 93 |
| # of terms/query | 9.90 | 9.12 | 7.14 |
| # of relevances/query | 23.20 | 8.17 | 22.40 |

To see the effect of our multilevel scheme on CPU time and retrieval performance, we report the hypergraph coarsening time, factorization time, and mean average precision using the `Medline`, `Cranfield`, and `NPL` data sets, with $d = 100, 200, 500$ in Tables 2–4, respectively. We adopted the term weighting scheme `tfn.cfx`, and used the Type-II methods with the splitting parameter $\alpha = 0$.

**Table 2: `Medline` results (Type-II, $\alpha = 0$, $d = 100$).**

| # levels | Coarsen. time | # docs | Factorization time SVD | SDD | NMF | Mean average precision SVD | SDD | NMF |
|---|---|---|---|---|---|---|---|---|
| #1 | N/A | 1033 | 1.44s | 9.62s | 64.16s | 70.9% | 51.8% | 66.5% |
| #2 | 0.11s | 517 | 0.66s | 7.70s | 47.98s | 70.7% | 59.6% | 69.4% |
| #3 | 0.06s | 259 | 0.38s | 6.76s | 35.97s | 70.6% | 64.9% | 68.7% |
| #4 | 0.04s | 130 | 0.09s | 6.38s | 20.66s | 66.4% | 57.5% | 66.3% |

**Table 3: `Cranfield` results (Type-II, $\alpha = 0$, $d = 200$).**

| # levels | Coarsen. time | # docs | Factorization time SVD | SDD | NMF | Mean average precision SVD | SDD | NMF |
|---|---|---|---|---|---|---|---|---|
| #1 | N/A | 1398 | 6.44s | 23.55s | 95.84s | 42.8% | 27.3% | 34.9% |
| #2 | 0.25s | 699 | 3.41s | 12.12s | 89.08s | 42.8% | 41.6% | 39.3% |
| #3 | 0.17s | 350 | 0.98s | 6.97s | 42.25s | 41.5% | 36.1% | 38.5% |
| #4 | 0.11s | 175 | 0.18s | 6.41s | 22.34s | 39.2% | 38.5% | 37.1% |

**Table 4: `NPL` results (Type-II, $\alpha = 0$, $d = 500$).**

| # levels | Coarsen. time | # docs | Factorization time SVD | SDD | NMF | Mean average precision SVD | SDD | NMF |
|---|---|---|---|---|---|---|---|---|
| #1 | N/A | 11429 | 66.34s | 644.15s | 2617.3s | 21.9% | 13.5% | 21.8% |
| #2 | 3.16s | 5717 | 66.96s | 266.67s | 802.91s | 22.7% | 18.7% | 21.3% |
| #3 | 1.61s | 2861 | 44.84s | 242.21s | 509.62s | 23.4% | 23.3% | 23.2% |
| #4 | 0.91s | 1434 | 29.06s | 179.56s | 389.26s | 22.8% | 23.3% | 22.8% |

As shown in Tables 2–4, we obtained improved or comparable retrieval performance at a reduced cost. Our multilevel technique significantly reduced the CPU time for NMF-based LSI, and dramatically improved the retrieval result of the SDD-based LSI. Note however that we used the term weighting scheme `tfn.cfx` that involves global term weights, which are discouraged for SDD-based LSI [11].

# 5. CONCLUSION

The hypergraph-based multilevel framework presented in this paper is applicable to any situation where the data matrix is sparse, since the pattern of non-zero entries of the sparse matrix yields a hypergraph. The graph is coarsened, with a known scheme such as maximal-weight matching, which merges pairs of vertices. Then a matrix approximation technique, e.g., SVD, is then performed on the coarsened data matrix at the lowest (coarsest) level. The corresponding projection is then applied to the original matrix.

[2] `ftp://ftp.cs.cornell.edu/pub/smart`

In text mining, the term-document matrices are sparse. Therefore, this hypergraph-based multilevel technique is ideally suited for the latent semantic indexing (LSI) approach. We call the resulting method multilevel-LSI. The experiments showed good retrieval improvement at reduced cost.

# 6. REFERENCES


[1] M. W. Berry and M. Browne. *Understanding Search Engines*. SIAM Publiscations, 1999.

[2] U. V. Catalyurek and C. Aykanat. Hypergraph-partitioning based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transaction on Parallel and Distributed Systems*, 10(7):673–693, 1999.

[3] E. Chisholm and T. G. Kolda. New term weighting formulas for the vector space method in information retrieval. Technical Report ORNL-TM-13756, Oak Ridge National Laboratory, Oak Ridge, TN, 1999.

[4] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *J. Soc. Inf. Sci.*, 41(6):391–407, 1990.

[5] K. Devine, E. G. Boman, R. Heaphy, R. Bisseling, and U. V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *20th International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[6] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42:143–175, 2001.

[7] L. Eldén. *Matrix Methods in Data Mining and Pattern Recognition*. SIAM Publications, 2007.

[8] N. Jardine and C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7(5), 1971.

[9] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Trans. VLSI Sys.*, 7(1):69–79, 1999.

[10] G. Karypis and V. Kumar. Multilevel $k$-way hypergraph partitioning. *VLSI Design*, 11(3):285–300, 2000.

[11] T. G. Kolda and D. P. O'Leary. A semi-discrete matrix decomposition for latent semantic indexing in information retrieval. *ACM Trans. Info. Sys.*, 16(4):322–346, 1998.

[12] T. G. Kolda and D. P. O'Leary. Algorithm 805: Computation and uses of the semidiscrete matrix decomposition. *ACM Tran. Math. Soft.*, 26(3):415–435, 2000.

[13] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[14] C.-J. Lin. Projected gradient methods for non-negative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[15] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[16] B. Vastenhouw and R. H. Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM review*, 47(1):67–95, 2005.