JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 25, 1539-1553 (2009)

A Distributed Multi-Dimensional Publications Management System*

TSUNG-YUAN LIU AND YUH-JZER JOUNG+
Department of Computer and Information Science
National Chiao Tung University
Hsinchu, 300 Taiwan
+*Department of Information Management*
National Taiwan University
Taipei, 106 Taiwan

Conventional hierarchical organizations are inadequate in managing publications which are inherently multi-dimensional. The rigid hierarchical organization limits a user to a small number of navigable paths, and ambiguities arise on the preferred location of an object. Prior research on enhanced browsing or information visualization suffers from the curse of dimensionality and is limited by the interaction usability and/or the computational efforts required.

This paper proposes a hypercube object space for modeling an interactive browsing methodology, called *multi-dimensional browsing*, along with an integrated, powerful, yet simple interface for navigating the object space. The entire navigation control is displayed as one concept hierarchy tree, allowing a user to navigate the object space by refining and generalizing multi-dimensional search criteria incrementally with strong feedback – but without the major drawbacks of prior research. A cross-platform, peer-to-peer architecture and modular implementation is also prototyped in Java to support the concept and to demonstrate its feasibility in handling thousands of publications.

Keywords: multi-dimensional browsing, hierarchical directory file system, publication management system, interfaces, interactive IR, peer-to-peer architecture

1. INTRODUCTION

The evolution of computing devices and the proliferation of personal computing have placed vast amount of e-Data at our disposal. In terms of publications, there now exist hundreds of thousands of papers in the form of Postscript or Portable Document Format (PDF). However, as the amount of available content grows, locating the exact information required becomes increasingly difficult. Without an adequate publications management system, there exists the danger of duplicate effort when researchers are unaware of similar work done already.

One of the most widely used information management technique is the plain vanilla hierarchical directory file system. The concept of the hierarchical directory file system is simple and natural, and hence its popularity. However, this simple system is doomed to fail when storing large numbers of publication papers.

Received October 31, 2007; revised May 22, 2008; accepted October 16, 2008.
Communicated by Jonathan Lee.
* A preliminary version of this paper appeared as "Multi-dimension browse," in Proceedings of the 28th Annual IEEE International Computer Software and Applications Conference, Hong Kong, China, 2004, pp. 480-485, and was partially supported in part by the National Science Council of Taipei, under grants No. NSC 94-2422-H-002-014 and 95-2422-H-002-017.
+ Corresponding author.

First, a publication has many dimensions such as its title, authors, keywords, publication date and publication place, *etc*. If we organize the directory according to one of the dimensions, then search according to the other dimensions becomes difficult. For example, if the directory structure groups papers by subject areas such as *Ad hoc Network* and *Routing*, then there is no direct way to find papers written by a certain author, or papers written in a specific year. This is a dimensional problem since we are trying to store something that is multi-dimensional (paper attributes) into a single dimensional space (directory hierarchy).

Secondly, some paper attributes such as authors and subject areas are not mutually exclusive, so logically the paper should appear in more than one directory. For example, a paper on *Ad hoc Multicast Routing* should be in both the *Ad hoc Network* and *Routing* directories (see Fig. 1). Using techniques such as symbolic links or duplicate copies only solves part of the problem, yet brings along additional complications, such as dangling links and wastage of storage space.

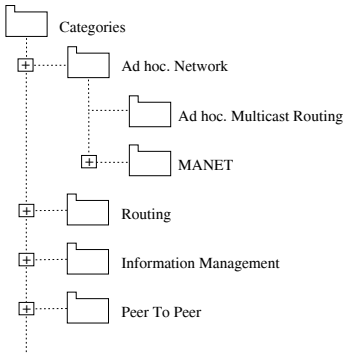

Fig. 1. Hierarchical directory file system.

Alternatively, papers can be stored in a database with text indices so that users can retrieve required information via queries. Search by querying requires some knowledge about the target object as well as the search engine so that appropriate queries can be formulated. Moreover, queries cannot be too general, or else thousands of search results could be returned. They may not be too specific either – unless users know exactly what they want; otherwise specific queries might bring nothing out. However, often times users cannot clearly describe what they want without clues, or cannot translate human language into system-understandable queries.

Browsing is an attractive retrieval technique for the casual user, or someone unfamiliar with the database contents or the query language. An interactive, iterative browsing interface should assist a user to find the required information quickly. This capitalizes on a well known human cognitive characteristic: it is easier to recognize some object than to describe it [1]. Browsing is especially desirable in a multi-user environment, as it is hard to discover what others have contributed without a browsable overview. The following quote by Mark Apperley [1] highlights the beauty of browsing:

Navigation is … about finding your way confidently and successfully to your goal while discovering fresh delights along the way.

This paper introduces an innovative and efficient browsing methodology that allows a user to find the required information quickly and easily via pointing and clicking operations. During the browsing process, the interface reveals summary information, giving the user quick insight of the scope and number of related papers in the system.

The rest of the paper is organized as follows: Section 2 discusses related work; section 3 presents the proposed multi-dimensional browsing scheme; section 4 briefs the system implementation; and finally, section 5 concludes and presents directions for future work.

2. RELATED WORK

Doyle [2] was among the first to propose an interactive browsing environment based on a graph structure for information retrieval purposes. Many variations have appeared since, and more modern interactive user interfaces based on *concept lattices* were proposed by Godin *et al*. [3] and Lindig [4]. However, the proposed interfaces are somewhat cluttered. Nevertheless, concept lattices do offer more degree of freedom in browsing than the conventional hierarchical organization, as concepts in the lattice are independent and users can browse along every possible path within the lattice. Our multi-dimensional browsing was in fact based on concept lattices. However, rather than using lattices, we use *hypercubes* to represent our browsing space so as to give a finer distinction of our navigation philosophy. A hypercube is geometrically symmetric, representing that at any point of browsing, a user can go to any dimension to view the document collection, henceforth the term "*multi-dimensional browsing*".

Browsing from multiple dimensions has also been highly appraised for image-based content. For example, the Flamenco system [5, 6] uses faceted metadata to describe images, and a user can select any arbitrary facet to navigate the art gallery. The Endeca information access platform[1] has also successfully applied this multi-dimensional browsing concept to manage large inventories for e-commerce companies (*e.g*., Wal-Mart, Barnes and Noble, IBM, and Home Depot). Compared to these systems, our work differs from them dramatically in user interface and in the browsing target. More importantly, we have proposed a formal model for this multi-dimensional browsing concept. Such a model is useful not only in capturing the browsing space and user interaction, but can also serve as a basis for measuring the computational complexity as well as for further possible optimization in the implementation.

Also related to our work are document management systems and knowledge management systems, where many commercial systems are now available. The focus of the commercial document management systems, *e.g*., Domino.Doc [7] and Onbase [8], is often on scalability and flexibility [9]. Scalability ensures that the stored contents are readily accessible by the users. Flexibility refers to the types of purposes the system can serve, and the extent to which the system can be customized and extended. Most document management systems offer only a simple text search and a hierarchical categorization of documents.

Knowledge management systems, *e.g.*, IBM Lotus Knowledge Discovery System [10], on the other hand, aim to track organizational knowledge and expertise that exist throughout the organization, and present it to the end user in a meaningful context, with

[1] http://www.endeca.com.

the effect of improving an organization's responsiveness, innovation, competency and efficiency. Contents and resources that a user requires are context dependent and forever changing. A successful knowledge management system has to separate contents that have value from those that do not for a particular context via automated discovery, knowledge mining or manual means.

Despite the efforts, one must admit that the intelligence and knowledge of present day systems is still no match for an experienced veteran. Our research philosophy is to separate the expertise of man and computer where their strengths lie. Man is responsible for maintaining a well-managed, deterministic browsing structure that is easy and efficient for fellow researchers to navigate. The computer is responsible for straightforward calculations, such as mass summarization, set manipulation, and interactive visualizations.

Moreover, although our publications management system is no match against the scope of current document and knowledge management systems that exhibit features such as automatic text analysis, automatic categorizing, version control, integrated workflow, compound documents, full text search, consistent replication and flexible access control, *etc*. [11], our interactive and efficient browsing methodology is unseen in these systems. The ad hoc browsing proposed in this paper is attractive for users unfamiliar with the stored contents, and allows such users to find the required information quickly via the interactive, iterative browsing interface. Existing systems should benefit with the addition of the proposed browsing methodology.

3. HYPERCUBE-BASED MULTI-DIMENSIONAL BROWSING

We propose an innovative and efficient browsing methodology – *multi-dimensional browsing* – which allows a user to navigate in a *hypercube object space* that is familiar with the way how humans locate information. List of symbols used in this section is tabulated in Table 1.

Table 1. List of symbols used and their description.

A: Number of objects in the database.
a_i: A particular object in the database, $0 \le i < A$.
F: Total number of distinct object features.
f_k: A particular object feature, $0 \le k < F$.
$\mathbb{F}a_i$: Set of logical features that are true for object a_i.
N: Number of nodes in an F-dimensional hypercube.
n_j: A particular node in the hypercube, $0 \le j < N$.
b^j_k: Bit k in the address of node n_j.
$\mathbb{F}n_j$: Set of features that are true for node n_j.
$\mathbb{A}_{n_j}$: Set of objects that have the same features as that of the node n_j, that is, $a_i \in \mathbb{A}_{n_j} \Leftrightarrow \mathbb{F}a_i = \mathbb{F}n_j$.
$\mathbb{A}^*_{n_j}$: Set of objects that contain features of node n_j, that is, $a_i \in \mathbb{A}^*_{n_j} \Leftrightarrow \mathbb{F}n_j \subseteq \mathbb{F}a_i$.

3.1 Hypercube Object Space

Consider a database with A objects $a_0, a_1, \ldots, a_{A-1}$. We characterize the objects with F features $f_0, f_1, \ldots, f_{F-1}$ where each feature is either *true* or *false* for a particular object

(or equivalently, whether an object exhibits that particular feature), such as "published in year 2002 or 2003", "object is a master thesis", and "keywords of object include *information retrieval*," *etc*. We define $\mathbb{F}_{a_i}$ as the set of logical features that are true for object a_i, that is, $f_k \in \mathbb{F}_{a_i}$ if and only if feature f_k is true for object a_i. Our model graphically corresponds to the hypercube structure. A d-dimensional hypercube is an undirected graph of 2^d nodes, and each node has degree d. Examples of hypercubes of dimensions 1, 2, 3, and 4 are shown in Fig. 2.

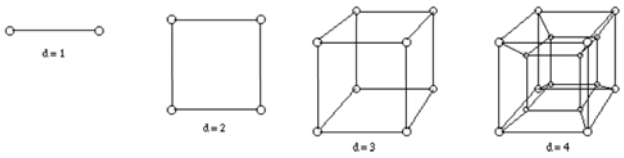

Fig. 2. Hypercubes of dimensions 1, 2, 3, and 4.

We use a logical F-dimensional hypercube, called *Hypercube Object Space* (*HOS*), to represent the object database. The hypercube consists of $N = 2^F$ nodes $n_0, n_1, \ldots, n_{N-1}$, where each node is identified by an F-bit binary number $b_0b_1\ldots b_{F-1}$. We associate each node with a set of features by letting a node's identity $b_0b_1\ldots b_{F-1}$ represent its features. That is, bit b_k indicates if (1 = *true*, 0 = *false*) feature f_k is associated with the node. We use $\mathbb{F}_{n_j}$ to denote the set of features that are associated with node n_j. Thus, a node with binary address $b^j{}_0b^j{}_1\ldots b^j{}_{F-1}$ has the set $\mathbb{F}_{n_j}$ such that $f_k \in \mathbb{F}_{n_j} \Leftrightarrow b^j{}_k = 1$. We populate the object space by placing the objects into the corresponding nodes in the hypercube. Each node n_j then contains a set of objects $\mathbb{A}_{n_j}$ such that $a_i \in \mathbb{A}_{n_j} \Leftrightarrow \mathbb{F}_{a_i} = \mathbb{F}_{n_j}$. We browse the HOS by navigating from node to node along the edges of the hypercube. The object set $\mathbb{A}_{n_j}$ is presented when visiting node n_j, and in each step we can follow any one of the F edges out of the node to explore other object sets.

The definition of a hypercube states that two nodes are linked with an edge if and only if their binary addresses differ in precisely one bit [12, section 3.1.1]. Thus traversing an edge directly translates to toggling the *true*/*false* state of a logical feature. Although visualization of a high dimensional hypercube is difficult, the features of the hypercube make a flat representation of the navigational interface possible, such as that shown in Fig. 3. The currently visiting node is presented by the on/off state of the list of features. The figure shows that the current node has feature 1 being *true* while the other features *false*. In this interface, we navigate to the next node by checking or unchecking any one of the F features, which corresponds to traversing over one of the F edges linked to the current node.

3.2 Object Aggregation

We introduce an object aggregation $\mathbb{A}^*_{n_j}$ that is the set of objects containing features of node n_j; that is, $\mathbb{A}^*_{n_j} = \cup_x \mathbb{A}_{n_x}$, where x is such that $b^x{}_k \geq b^j{}_k$ for all $0 \leq k < F$. In terms of object features, the aggregation can alternatively be defined as follows: $a_i \in \mathbb{A}^*_{n_j} \Leftrightarrow \mathbb{F}_{n_j} \subseteq \mathbb{F}_{a_i}$. The object aggregation is useful for object retrieval – it alleviates the need to fully specify an object's characteristics. Specifying a subset of features will return a set of

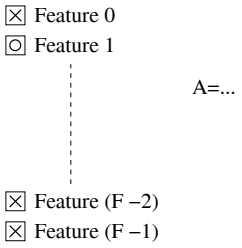

Fig. 3. Navigation interface in a hypercube object space.

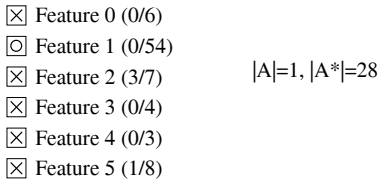

Fig. 4. Navigation interface in a 6-dimensional hypercube object space with 54 objects.

objects containing those required. The size of the returned set $|\mathbb{A}^*_{n_j}|$ decreases with increasing specificity of n_j (that is, $\Sigma_i b^j_i$).

An example browsing interface with addition of the aggregation is shown in Fig. 4. The figure shows a 6-dimensional HOS with 54 objects. Conceptually, a user is at a node of the hypercube object space in each iteration, where the current node is reflected by the on-off state of the features as before. The object sets $\mathbb{A}_{n_c}$ and $\mathbb{A}^*_{n_c}$ for the currently visiting node n_c are displayed. In addition, the sizes of the object sets $\mathbb{A}_{n_j}$ and $\mathbb{A}^*_{n_j}$ for the node n_j that is to be navigated when selecting one of the features are shown next to that feature. For example, the figure shows that the current node n_c has feature 1 being *true*, while the rest of the five features being *false*. There is exactly one object ($|\mathbb{A}_{n_c}| = 1$) with this characteristic, while there are 28 objects ($|\mathbb{A}^*_{n_c}| = 28$) in total that have feature 1 being *true*. The information (3/7) next to feature 2 indicates that there are 7 objects in our database with features 1 and 2, and among them, 3 objects have exactly the two features being *true* but the rest of the features being *false*. Similarly, the information (0/4) next to feature 3 tells us that if the feature is selected, then we are expected to see a total of 4 objects in our database that have features 1 and 3, but all of the 4 objects have some other features with them (as there is no object having just features 1 and 3).

Intuitively, the document aggregation $\mathbb{A}^*_{n_j}$ is the set of documents navigable if we are only allowed to change the state of logical properties from *false* to *true*. Thus, as we specify more and more logical properties, we narrow down the browsing region, with each navigated document set a subset of the previous navigated set. A user thus browses the database iteratively by specifying a document property (asserting the property to be *true*) in each iteration. Our browsing scheme can be further refined by hiding properties that leads to nodes with $|\mathbb{A}^*| = 0$. This allows us to choose only relevant sets of document properties pertaining to the current document database.

We note that the browsing methodology proposed can achieve the same effect as concept lattice browsing from prior researches. Essentially, the “more specific”, “current” and “more general” browsing features in concept lattice [3] are united into the sin-

gle interface of our scheme, and the document sets stored in vertices of the concept lattice browsing correspond to the document set $\mathbb{A}^*$ in our multi-dimensional browsing.

3.3 Multi-Dimensional Browsing

The multi-dimensional browsing interface we propose is shown in Fig. 5. Every attribute in every document stored in the system is summarized and displayed in a familiar hierarchical style. We refer to this structure as a *concept hierarchy tree*. The number in brackets before an attribute gives the number of documents containing that attribute (plus the attributes selected so far).[2] This summary offers a quick overview of the scope and amount of stored contents available in the system.

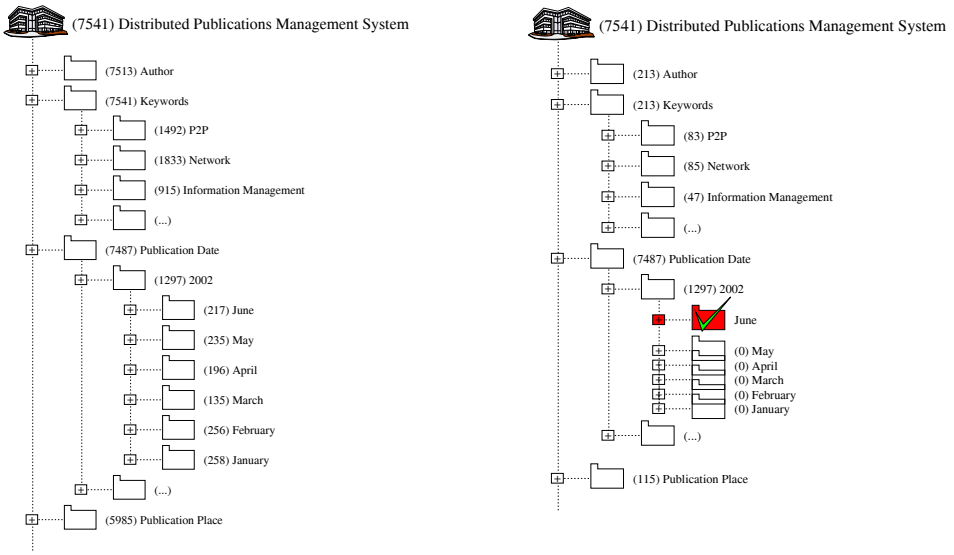

Fig. 5. An example of concept hierarchy tree with object counts. The right one shows the result after selecting publication date 2002/June in the left.

Although similar in view to a conventional directory tree, the user interface interaction is radically different. The user browses the stored contents by selecting and de-selecting object features in the concept hierarchy tree. For example, say we are looking for papers published in June 2002, we simply check the feature *2002/June*. Note that with our object summarization feature, we already know that there are 217 such papers before making the selection.

Similar to a conventional file management system, the selected objects (synonymous to files in a directory) are shown in a contents panel in list, icon or thumbnail view (not shown). Our system differs in that the user can opt to select or de-select any number of features in any dimension, and in any order at will. The effect of the selection is that it

[2] More precisely, it is the number of documents that will be in the browsing pool after selecting that attribute. This number is $|\mathbb{A}^*_{n_j}|$ for the to be visited node n_j. Also note that in the example shown in Fig. 9, a document may not have all the necessary attributes. For example, among the 7,541 documents, only 7,513 of them have an author attribute.

allows a user to browse different sections of the stored database, or in view of the HOS a node in that space. This truly free and multi-dimensional browsing capability lured us to coin the term *ad hoc browsing* in our project. This term reflects on the end-user's perspective better than the theoretical term, *multi-dimensional browsing*, and is used throughout the user interface that is shown in the following section.

The right one in Fig. 5 shows the resulting concept hierarchy tree after we have selected the feature *2002*/*June* in the left. Examining the concept hierarchy tree we see that the object counts next to each feature have been updated – they are filtered by the current object selection. The 83 next to the feature *P*2*P* means that there are 83 papers on *P*2*P* published in *2002*/*June*. Thus, even if we are not looking for peer-to-peer papers in particular, we can still learn the scope and amount of papers available in the sea of 217 publications, which could be useful. Additionally, we can choose to hide features that lead to nodes with $\mathbb{A}^* = 0$. This allows us to choose only relevant sets of object features pertaining to the current object database. For example, since a paper can be published at most once, no paper can contain both the features *2002*/*May* and *2002*/*June*. This is reflected in the figure by an object count of zero next to *2002*/*May*. Concepts that are irrelevant such as *2002*/*May* in this case are conveniently compressed as shown in the figure, and can be completely removed from the interface.

A minor benefit of our user interaction is the close correlation with traditional hierarchical organizations. In fact, with slight modifications – allowing a user to select one concept only, and do not sum document sets from child nodes for a parent node – our system can be used like a normal file system. This can be beneficial for a smooth transition, in that a backward compatibility mode can be provided for newcomers.

3.4 Data Structure and Algorithm

The data structure used to implement multi-dimensional browsing is as follows:

- Each concept (feature) c_i in the concept hierarchy tree is an object that contains a reference to its parent concept and references to its child concepts.
- Each concept c_i also contains a hash set $\mathsf{A}^*_{c_i}$ that stores the set of objects that contain c_i.
- A hash set S^c is used to store the currently selected set of concepts.
- A hash set S^a is used to store the currently selected set of objects.

In the initialization step, the set $\mathsf{A}^*_{c_i}$ for each concept c_i in the concept hierarchy tree is computed, and S^c and S^a are set to the empty set and all objects set respectively. At the end of the initialization step, the concept hierarchy tree along with the object counts (size of $\mathsf{A}^*_{c_i}$) is displayed for user navigation.

In the multi-dimensional browsing stage, whenever a user selects a concept c_j from the concept hierarchy tree, that concept is added to S^c and the set S^a is intersected with the set of objects $\mathsf{A}^*_{c_j}$ that contain the concept c_j. If a concept c_j is de-selected, that concept is removed from S^c and the set S^a is recomputed from $\mathsf{A}^*_{c_i}$ for all $c_i \in S^c$. Whenever the set S^a changes, each visible concept node object in the concept hierarchy tree is redrawn to reflect the change in object count. The intersection size between the selected object set S^a and the object set in the concept node ($\mathsf{A}^*_{c_i}$) is displayed next to the tree node, and nodes that have a zero intersection size are compressed (by setting the node's row height to a small value).

3.5 Complexity Analysis

In this subsection we analyze the time complexity required in order to achieve the interface interaction as described. We list the notations used in our analysis:

N: The number of objects stored in the system.
C: Total number of concepts.
s: (= $|S^c|$): The size of the currently selected set of concepts.
$\hat{a}$: the depth of the concept hierarchy tree.
$\hat{c}$: the maximum number of concepts an object can contain.

In the initialization step, we scan the concepts that an object contains and place the reference of the object under the appropriate concept nodes of the concept hierarchy tree. The hierarchy requires us to place the object under the concept node's object set and all parent concepts' object sets. With the (hash) data container proposed, such operations require $O(\hat{a})$ time for each concept an object contains. Processing all concepts of an object thus requires $O(\hat{c}\hat{a})$ time, and the total time complexity for initialization is $O(N\hat{c}\hat{a})$. In practical cases, $\hat{c}$ and $\hat{a}$ are much smaller than C, so the time complexity for initialization is loosely bounded by $O(NC)$. Object insertions, deletions and updates after initialization all have cost $O(\hat{c}\hat{a})$, which is $O(C)$ and much better in real cases. Concept modifications cost $N\hat{a}$ at worst, which is $O(N)$ in practice.

The set S^a is calculated from $\mathsf{A}^*_{c_i}$ for all $c_i \in S^c$, and thus requires worst case time complexity $O(sN)$. Updating S^c is a trivial $O(1)$ operation. To display the object counts next to the concepts in the concept hierarchy tree, we need to calculate the intersection count between the currently selected object set and the object set for that concept, which is an $O(N)$ time operation for each node. However, since only the object counts fall within the currently viewable region in the interface need to be calculated, given a fixed bounded screen size, the time complexity to calculate the counts is only $O(N)$. Even if irrelevant concepts are compressed, the "hidden" concepts still occupy finite space, and thus for any fixed view region, the time complexity required is still $O(N)$. As the user scrolls the viewable region or expands contracted nodes, the pending counts are calculated interactively on demand. The total time complexity for the multi-dimensional browsing action is thus only $O(sN)$ and, with the small s expected in practical usage, the complexity reduces to $O(N)$.

More efficient data structures and algorithms, perhaps those from that of data cube researches [13-15], will be beneficial in future work.

3.6 Feature Extensions

We note that the features in multi-dimensional browsing are generic, and thus multi-dimensional browsing can be easily complemented by other search techniques, such as full-text search or search by similarity. The resultant object set from other search techniques can be regarded as another "feature", and is fed back into the multi-dimensional browsing interaction to obtain a quantitative/qualitative overview and for further iterative interactions. Multiple resultant object sets from the extended searches will result in additional feature nodes displayed in the concept hierarchy tree, and these features are avail-

able to the user for further multi-dimensional browsing. We see that multi-dimensional browsing is indeed unlimited in terms of the number and nature of dimensions/features to browse against.

4. SYSTEM IMPLEMENTATION

4.1 Overview

A multi-dimensional publications management system with the browsing interactions described is implemented over a cross-platform, peer-to-peer architecture. The design choice of this system architecture is to reflect a typical decentralized publications management system within a mid-size organization.[3] The peers are interconnected by some communications medium, and have the ability to communicate with each other via message passing. They can be partitioned into different groups based on, say, their interests or organizational structure. Each peer can be in several groups, and for each group it maintains a list of peers within the group for communication. Optional rendezvous nodes may be employed to assist peer and group discovery.

Each peer stores some publications and publishes the index to other peers. Collecting the indices allows a user to browse documents that are physically stored on distributed machines off-line. Actual file contents are downloaded on demand by the user, and document updates are sent asynchronously as inter-server gossips.

Each publication contains a number of multi-dimensional attributes, and has possibly one or more content formats – the actual informational content of the document in a digital format. The document attributes may be automatically deduced from the document contents by automatic text analysis using techniques such as natural language processing [16], or managed manually. Manual management is sometimes desirable as proper administration of document attributes can filter out irrelevant information accurately and quickly. Although it costs more than automatic text processing, it is justified for our purpose since the frequency of document retrieval is expected to be much more than that of document insertion or modification in a multi-user environment.

4.2 Graphical User Interface

The main user interface, shown in Fig. 6, consists of action controls (menu and tool bar) and a tabbed pane. Each tab (except for the first, which show messages useful for debugging) in the tabbed pane corresponds to a group that the user has joined or created. Information about the peers of a group and the last interaction with the peers is tabulated in the tab panel. The user can aid peer discovery by informing the system of known peers, or requesting the system to perform a bandwidth-consuming flooding discovery. There is also an option to disable data synchronization within a group to conserve bandwidth. Disabling data synchronization is not desired if there are pending document modifications not yet distributed to remote peers, as this will raise the likelihood of conflicting updates.

[3] For small organizations, a centralized server would suffice to manage the publications, while for large organizations that involve thousands of peers, a more deliberate design in the peer-to-peer architecture may be needed to facilitate a more efficient communication and peer management within the network.

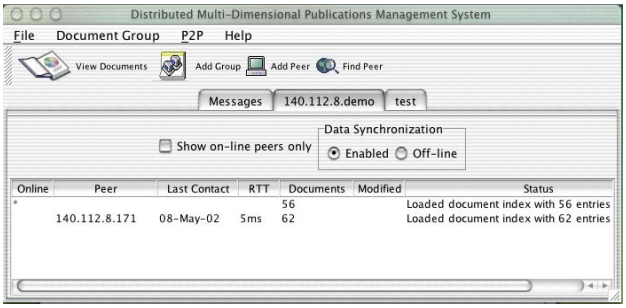

Fig. 6. Main frame of the system.

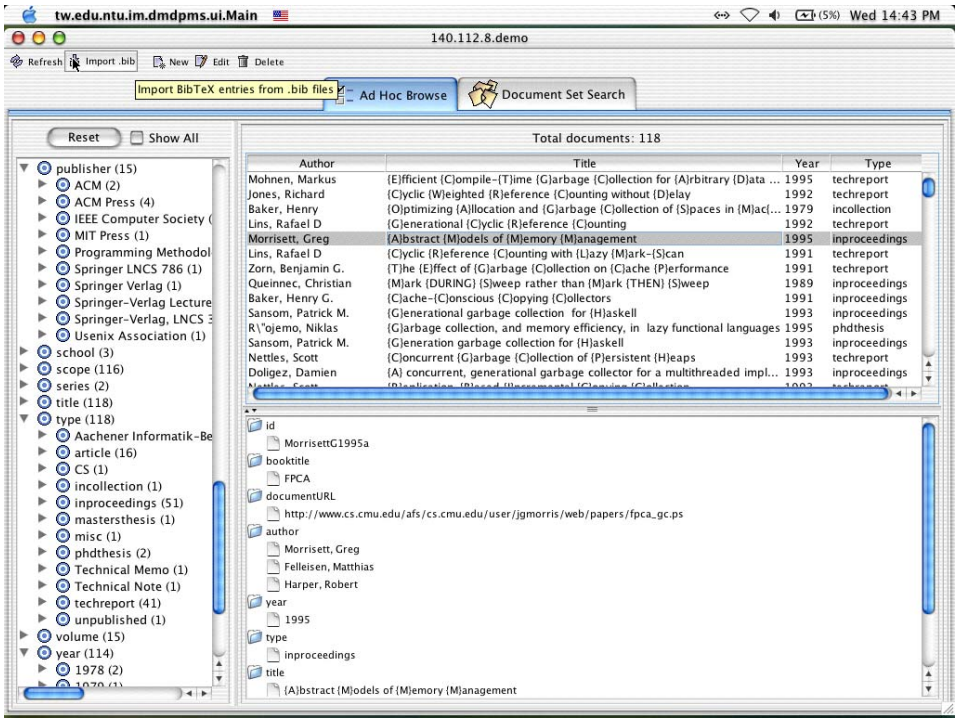

Fig. 7. Ad hoc browsing user interface.

When "*View Documents*" is clicked, a snapshot of the current document indices is taken and the main document browsing frame is displayed. A user is free to open multiple browsing frames to browse documents in different groups simultaneously. A screenshot of the document browse frame is shown in Fig. 7. It consists of a toolbar and a tabbed pane with two tabs, offering the *ad hoc browsing* and the *document set search* features respectively. The ad hoc browsing tab consists of three panels: The concept hierarchy tree on the left that shows all the concepts contained in all available documents. The user browses and selects the desired set of documents as that described in section 3.3. The current selected set of documents is listed in the top right panel, and clicking on a

document reveals its details in the bottom right panel. Document contents and URL fields can be clicked on in the bottom right document panel to bring up a browser.

A screenshot of the document set browsing user interface is shown in Fig. 8. The top half portion for document set selection allows a user to select and combine arbitrary sets of documents with graphical visualization. It consists of four functional panels: (1) tree panel that consists of a concept hierarchy tree to be browsed and selected; (2) selection panel that shows the currently selected sets of documents; (3) operations panel that allows a user to specify which set operation to perform, along with a graphical display of the selected set operation underneath the radio button; and (4) saved query panel that allows a user to save the current set operation result for further manipulations of compound sets, thus offering the user full expressivity of Boolean logic selections.

The lower half lists the documents currently selected. Clicking on a document reveals its details in the pop-up bottom right panel (not shown in the figure).

As an example, to browse technical reports that are published in either 1993 or 1994, we first drag the concepts *year/1993* and *year/1994* from the concept hierarchy tree and drop these selections onto the selection panel. The operations panel will list the available set operation pertaining to our selection. In this example, the two selected document sets are disjoint, so operations such as set intersection and subtraction will be appropriately dimmed. We inform the system to perform the union operation on the document sets by selecting the *A OR B* radio button in the operations panel. After saving the "*1993 or 1994*" query for further manipulation, we then drag the concept *type/techreport* and inform the system that we wish to perform an intersection (*A AND B*) operation. The result (shown in Fig. 8) is then all the technical reports that are published in 1993 or 1994.

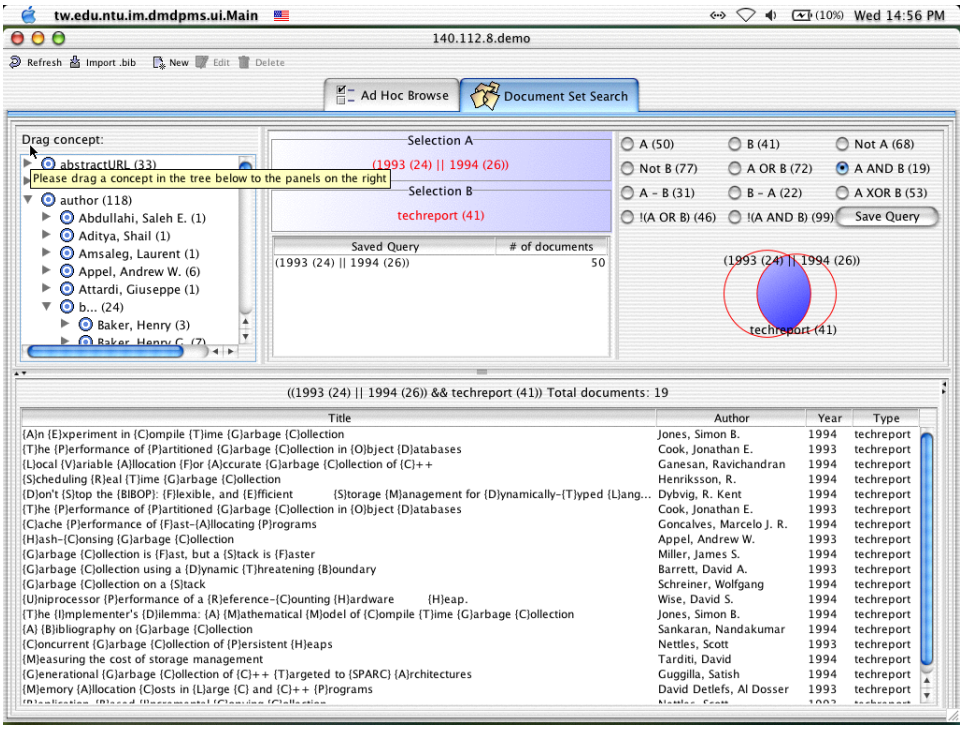

Fig. 8. Document set browsing user interface.

Apart from allowing a user to select any arbitrary set of documents based on the concepts they contain, it is worth to note that this interactive interface also allows a user to visualize the correlation between any two document sets. Specifically, the graphical drawing allows one to see the proportion of the overlap between two sets of documents, or whether a set completely embodies another. This knowledge is nontrivial to achieve in a conventional document management system.

The user can also freely contribute new documents or edit any document stored on the system (subject to security checks). Our implementation features direct importing of BibTeX bibliography database files. The user can select one or more BibTeX file for parsing, and the parsed BibTeX entries will be presented for further editing and refinement.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have pointed out the inadequacies of the conventional hierarchical organization in handling multi-dimensional objects such as publications. The rigid hierarchical organization limits a user to a small number of navigable paths, and ambiguities arise on the preferred location of an object. Prior research on enhanced browsing or information visualization suffers from the curse of dimensionality and is limited by the interaction usability and/or the computational efforts required.

The multi-dimensional browsing proposed in this paper navigates in a hypercube object space using an integrated, powerful, yet simple interface. The entire navigation control is displayed as one concept hierarchy tree, offering the user full control of the navigation process without getting disoriented, and at the same time requires low browsing as well as low data maintenance computational cost. The browsing methodology proposed also allows the user to begin the navigation process starting with any object set, thus enabling simple integration with other research efforts.

We have also prototyped a cross-platform, peer-to-peer architecture and modular implementation in the Java language as proof-of-concept and performed user evaluation. Preliminary user experience of the browsing methodology is positive, and our system loaded with thousands of bibliographical records and dimensions offers adequate performance on a moderate PC.

Future work will focus on production quality implementation with better crafted interfaces, security, and integration with researches from information retrieval and data mining. Moreover, we are planning a more comprehensive user experience study for this multi-dimensional publications management system. Interested readers may also refer to [17] for a related evaluation study of multi-dimensional browsing that additionally takes context into account.

REFERENCES

1. S. Jul and G. W. Furnas, “Navigation in electronic worlds: A CHI 97 workshop,” *ACM SIGCHI Bulletin*, Vol. 29, 1997, pp. 44-49.
2. L. B. Doyle, “Semantic road maps for literature searchers,” *Journal of the ACM*, Vol.

8, 1961, pp. 553-578.
3. R. Godin, J. Gecsei, and C. Pichet, "Design of a browsing interface for information retrieval," in *Proceedings of the 12th ACM SIGIR Conference on Research and Development in Information Retrieval*, 1989, pp. 32-39.
4. C. Lindig, "Concept-based component retrieval," *Working Notes of the IJCAI-95 Workshop*: *Formal Approaches to the Reuse of Plans*, *Proofs*, *and Programs*, 1995, pp. 21-25.
5. M. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen, and K. P. Yee, "Finding the flow in web site search," *Communications of the ACM*, Vol. 45, 2002, pp. 42-49.
6. K. P. Yee, K. Swearingen, K. Li, and M. Hearst, "Faceted metadata for image search and browsing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2003, pp. 401-408.
7. IBM Corporation, IBM Lotus Domino Document Manager, http://www-01.ibm.com/software/lotus/products/dominodocumentmgr/, accessed 2008.
8. Hyland Software, Inc., Enterprise content management – Integrated Document Management – OnBase by Hyland Software, http://www.onbase.com/, accessed 2006.
9. R. Allen, "White paper: Document management systems survey," Technical Report, Information Technology at Johns Hopkins, 2002, http://www.it.jhu.edu/nts/status/dmwp.pdf.
10. W. Pohs, G. Pinder, C. Dougherty, and M. White. "The lotus knowledge discovery system: Tools and experiences," *IBM Systems Journal*, Vol. 40, 2001, pp. 956-966.
11. R. Grutter and K. Stanoevska-Slabeva, "Document, communication and operating system standards: A survey on compound document standards, document management systems, and document database systems," Technical Report, Institute for Information Management, University of St. Gallen, 1997.
12. F. T. Leighton, *Introduction to Parallel Algorithms and Architectures*: *Arrays*, *Trees*, *Hypercubes*, Morgan Kaufmann Publishers, San Francisco, 1991.
13. S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi, "On the computation of multidimensional aggregates," in *Proceedings of the 22nd International Conference on Very Large Data Bases*, 1996, pp. 506-521.
14. T. Johnson, "Performance measurements of compressed bitmap indices," in *Proceedings of the 25th International Conference on Very Large Data Bases*, 1999, pp. 278-289.
15. M. Riedewald, D. Agrawal, and A. E. Abbadi, "Flexible data cubes for online aggregation," in *Proceedings of the 8th International Conference on Database Theory*, Vol. 1973, 2001, pp. 159-173.
16. C. Faloutsos and D. W. Oard, "A survey of information retrieval and filtering methods," Technical Report, No. CS-TR-3514, University of Maryland, College Park, U.S.A., 1995.
17. L. L. Wu, Y. L. Chuang, and Y. J. Joung, "Contextual multi-dimensional browsing," *Computers in Human Behavior*, Vol. 24, 2008, pp. 2873-2888.

Tsung-Yuan Liu (劉宗原) received his B.S. in Electrical Engineering from the University of the Witwatersrand, R.S.A., and M.B.A. from National Taiwan University. He is currently pursuing a Ph.D. from the Institute of Computer Science and Engineering at the National Chiao Tung University. His research interests include Internet technologies, information hiding, and artificial intelligence.

Yuh-Jzer Joung (莊裕澤) received his B.S. in Electrical Engineering from the National Taiwan University in 1984, and his M.S. and Ph.D. in Computer Science from the State University of New York at Stony Brook in 1988 and 1992, respectively. He is currently a Professor at the Department of Information Management in the National Taiwan University, where he has been a Faculty member since 1992. From 1999 to 2000, he was a visiting scientist at the Lab for Computer Science, Massachusetts Institute of Technology. He was the Chair of his Department from 2001 to 2005. His main research interests are in the area of distributed computing, with specific interests in multiparty interaction, fairness, (group) mutual exclusion, ad hoc and peer-to-peer computing, and personal data management.