

# DiHO: A Distributed Higher-Order Association Rule Miner

Shenzhi Li

Department of Computer Science and Engineering  
19 Memorial Drive West, Lehigh University  
Bethlehem, PA 18015, USA  
1-610-758-3737

shl3@lehigh.edu

William M. Pottenger

Department of Computer Science and Engineering  
19 Memorial Drive West, Lehigh University  
Bethlehem, PA 18015, USA  
1-610-758-3454

billp@cse.lehigh.edu

## ABSTRACT

Since its inception association rule mining (ARM) has been an important research area in the data mining field. To deal with ever increasing database sizes, parallel ARM variants that improve performance have been developed. More recently, as the need to mine patterns across distributed databases has emerged, distributed ARM algorithms have been developed. These algorithms, however, assume that the databases are horizontally distributed. As a result, it is not possible to use existing distributed ARM algorithms to discover rules based on itemsets that are formed from vertically fragmented data.

In this paper, we present the DiHO Miner, a novel Distributed Higher-Order association rule mining algorithm that mines association rules in vertically fragmented distributed databases. The DiHO Miner employs a mapping function that resolves local primary keys in each database and identifies globally unique keys. Higher order associations between vertically fragmented attribute values across distributed databases are discovered via these keys. Evaluation methods are proposed to incorporate the performance of the mapping function into the traditional ARM evaluation metric support.

## 1. INTRODUCTION

Association rule mining (ARM) is an important research area in the field of data mining. The goal of ARM is to discover associations between attribute values. Given two distinct sets of attribute values,  $X$  and  $Y$ , we say  $Y$  is associated with  $X$  if the appearance of  $X$  implies the appearance of  $Y$  in the same context. ARM outputs a list of association rules of the format  $X \Rightarrow Y$ , where  $X \Rightarrow Y$  has a predetermined support and confidence. The prototypical application of ARM is market-basket analysis in which items that are frequently purchased together are identified in order to aid in the layout of items in the store.

In recent years, database systems are not only geographically distributed but also vertically fragmented. One important example is the databases employed in law enforcement. For example, there are more than 1,260 police jurisdictions in the Commonwealth of Pennsylvania alone, many of which utilize different database products and schemas. As was made strikingly clear in the aftermath of the terrorist attack on September 11, different kinds of records on a given individual may exist in different databases – a type of vertically fragmented data. In fact, the United States Department of Homeland Security (DHS) recognizes that the proliferation of databases and schemas involving fragmented data poses a challenge. In response, the DHS is promulgating a “System

of Systems” approach that acknowledges the infeasibility of creating a single massive centralized database [5]. Indeed, the picture that is emerging in the US is basically a three-tier structure with some overlap between tiers: local databases, state databases and federal databases. The state collects information from local jurisdictions to form a state-level centralized database, and likewise for the federal government. However, due to the sheer volume of data, constraints on system interoperability as well as legal restrictions on data sharing, not all information is passed from local jurisdictions to the state-level. Likewise, the federal level captures only a modicum of the data available in state-level databases. In essence, the resulting data sharing structure is pyramidal in nature. The more centralized the database, the less information that is shared. Given this reality, the DHS as noted has acknowledged that it is simply not feasible to keep an all-in-one federal database. As a result, the DHS is promoting a “System of Systems” approach that is based initially on the creation of standards for interoperability and communication in areas where standards are currently lacking. Indeed, efforts are underway to establish standards in database schema integration (e.g., OWL [8], GJXDM [10], etc.). Nonetheless, even with the widespread acceptance of such standards, the need for distributed mining of vertically fragmented data in a “System of Systems” will remain.

Although important, law enforcement is but one application of distributed mining of vertically fragmented data. By and large, ARM algorithms mine association rules from data in a homogeneous database that may or may not be distributed. Even multi-relational ARM algorithms mine rules from multiple tables within a given homogeneous database. It is our contention that the restriction to a homogenous database source is unnecessary, and that useful rules can be mined from diverse databases with different schemas as long as records can be linked via, for example, a unique key. Many interesting applications emerge if one considers this approach, which we term *higher-order* distributed association rule mining. *Higher-order* implies that rules may be inferred between attributes that do not naturally occur in the same database schema (regardless of whether the schema itself is distributed or not). In other words, rules can be inferred based on attributes that never occur together in any record in any of the distributed databases being mined.

To clarify this important point, consider as an example two or more patent information records in distinct databases. There are many such databases, each recording slightly different information for a given patent. For example, the United States Patent and Trademark Office (USPTO) database keeps the full text of United States patents online. Each patent is classified using the United States classification system. On the other hand, the commercial intellectual property management firm Thomson Derwent is a leading value-added reseller of patent-related information. As such, many patent intelligence experts prefer the Derwent classification to the US classification. Derwent, however, does not maintain the full text of patents nor a linkage from their coding system to the US coding system. As a result, experts in patent intelligence would welcome higher-order association mining rules linking the

Derwent classification to the US classification, and, consequently, to the full-text of patents online at the USPTO.

There are many other practical examples of how distributed higher-order rules can be mined from multiple diverse databases. The common advantage of employing such technology is the promise of discovery of higher-order associations in data sources that cannot be easily centralized. Thus the area of distributed association rule mining in general, and higher-order distributed mining in particular, is an intriguing and promising area of research. Existing distributed ARM algorithms, however, assume that the distributed databases are homogenous; i.e., that the data is not vertically fragmented. In this paper, we present a novel ARM algorithm, the DiHO Miner, which supports mining of higher-order rules from distributed databases that contain vertically fragmented data.

The article is organized as follows. Section 2 summarizes the related work in distributed ARM. In section 3, background knowledge introducing multi-relational ARM is provided. Our DiHO algorithm is presented in section 4, followed in section 5 by a discussion of changes needed in the evaluation of support given the vertically fragmented, distributed nature of the data. We draw conclusions and discuss future work in section 6.

## 2. RELATED WORK IN PARALLEL & DISTRIBUTED ARM

Parallelism is an ideal way to scale ARM to large databases. There are two major approaches for using multiple processors: parallel ARM algorithms, in which all processors access shared memory, and distributed ARM algorithms, in which each processor accesses its own private memory and communication is accomplished via message passing. In general terms, communication between processors in parallel ARM algorithms is based on the use of variables allocated in shared memory, while communication in distributed ARM algorithms is based on message passing. Most existing parallel and distributed ARM algorithms are based on a kernel that employs the well-known Apriori algorithm [1,2].

Parallelism in both shared-memory and distributed memory ARM algorithms can be categorized as *data-parallelism* or *task-parallelism* [3,7,20]. Data-parallelism logically partitions the database among processors. Each processor works on its local partition using the same computational model. Count distribution (CD) is a simple data-parallelism algorithm. Each processor generates the local candidate itemsets independently based on the local partition. Then the global counts are computed by sharing (or broadcasting) the local counts, and the global frequent candidate itemsets are generated. Data-parallelism exchanges only the counts among processors, which minimizes the communication cost. As a result, it seems ideal for use in a distributed environment. Data-parallelism is however only suitable for use on homogenous databases.

In task-parallelism, each processor performs different computations independently, yet all have access to the entire dataset. For example, the computation of candidate itemsets of different sizes could be distributed among processors in a parallel loop across itemsets. In this case, each processor generates global counts independently. As noted, this requires that each processor have access to the entire dataset. In a distributed environment, this can be accomplished by performing an initial ‘copy-in’ operation of the dataset.

These two types of parallelism serve to distinguish one of the tradeoffs between parallel and distributed data mining: in essence, it boils down to scalable processing power vs. network latency and bandwidth. On a low-bandwidth, high-latency network, however, it is unlikely that either form of parallelism will be practical.

Distributed ARM algorithms discover rules from distributed databases. Fast Distributed Mining (FDM) is based on count distribution [6]. The advantage of FDM over CD is that it reduces the communication cost by sending the local frequent candidate itemsets to a polling site instead of broadcasting. Also based on CD, Ashrafi, et al. [4] propose the Optimized Distributed Association Mining (ODAM) algorithm which both reduces the size of the average transaction and reduces the number of message exchanges in order to achieve better performance. The transactions are reduced by deleting the non-frequent items from the itemsets and merging several transactions with the same itemsets into one record. As for the message exchange, instead of using broadcast as with CD or polling sites like FDM, ODA just sends all local information to one site, called the receiver. The receiver then calculates the global frequent itemsets and sends them back.

Based on our survey of existing parallel and distributed ARM algorithms, we determined that all make the following basic assumptions: 1. the databases are homogenous; 2. the databases are horizontally fragmented; 3. the databases only contain one table (e.g., have been denormalized). All three assumptions prevent parallel or distributed algorithms from discovering higher-order associations directly from the database tables.

## 3. MULTI-RELATIONAL ARM

The parallel and distributed ARM algorithms discussed in the previous section make the assumption that an object (composed of a primary key and associated attributes) appears only once in only one of the databases. In many situations, however, different attributes of a single object may be recorded in more than one table in a single database and referenced through a shared primary key. The simplest case of this is a single (non-distributed) database with two tables, linked by a primary key. For example, suppose we have one table that records authors of a given document, and a second that records citations. The shared document ID links these two tables together. Modulo database denormalization, existing ARM algorithms are not designed to discover rules that cross table boundaries and, in this case, associate authors with citations<sup>1</sup>.

There is however a type of ARM algorithm designed specifically to mine rules across tables in a single database: multi-relational ARM. In fact, multi-relational data mining in general (not limited to ARM) is an emerging research area that enables the analysis of complex, structured types of data such as sequences in genome analysis. Similarly, there is a wealth of recent work concerned with enhancing existing data mining approaches to employ relational logic. WARMR, for example, is a multi-relational enhancement of Apriori first presented by Dehaspe et al. [9]. Later, Nijssen and Kok [14] proposed an algorithm dubbed

---

<sup>1</sup> We realize that simple rules like these could be ‘mined’ trivially by invoking, for example, an SQL query across tables – this example simply serves to highlight this characteristic of the ARM algorithms discussed.

FARMER which makes several modifications to WARMR to improve performance. Since an understanding of multi-relational ARM will be helpful in understanding DiHO ARM, the following subsection gives a brief introduction to these algorithms.

### 3.1 Multi-relational ARM: WARMR and FARMER

First, we will introduce some basic notation used in multi-relational ARM. WARMR, presented in [9], is an Apriori-based algorithm. The key difference between WARMR and Apriori is that WARMR uses sets of atoms based on first order logic as an extension to the itemsets used in Apriori. The atoms are also referred to as queries if all the variables are quantified and the set is ordered. An example query is:  $Key(X) \rightarrow Buys(X, Y, card), Property(X, loyal)$ . Here  $X$  is a variable that takes unique customer IDs as its range of values.  $Buys$  and  $Property$  are predicates, and can be thought of as different tables in a relational database. The  $Buys$  predicate records purchases – i.e., customer value( $X$ ) buys some item value( $Y$ ) using a particular credit card. The  $Property$  predicate specifies that customer value( $X$ ) has the property of being loyal. The key value( $X$ ) connects the two predicates. In this example, it is actually only the body of the Horn clause that is termed a query. The support of the query is defined as the number of variable bindings for the head of the Horn clause  $Key(X)$ .

The search space in multi-relational ARM is altered due to the change from itemsets to atom sets. To define the *bias* of the search space, WARMR uses a refinement operator based on *mode declarations*. Simply put, mode declarations specify how predicates can be connected to one another. As such, mode declarations guide the process of adding predicates to a query. Here is an example of a mode declaration:

$$Buys(+, -, \#)$$

This mode specifies that the predicate  $Buys$  can be added to a query when its first parameter is bound to an existing, or input, variable, its second parameter is bound to a free, or output, variable and the third parameter is bound to a constant (represented by the  $\#$  character). The parameters are called mode constraints. Often, an integer is associated with the mode to indicate the maximum number of times the mode can be used in a given query. In our example, a particular value of the customer ID variable  $X$  is input to the unification process, any value instantiated for the variable  $Y$  is considered output, and the nominal value ‘card’ is a constant. An example of the application of this process is a customer who makes a purchase using ‘card’ and is therefore considered loyal.

Using atom sets in multi-relational ARM leads to a change in the relation used for counting and evaluation as well. Apriori makes use of the subset relation in itemset counting and evaluation. In multi-relational ARM, however, the subset relation cannot be used to express the relations among atom sets. Instead, WARMR uses  $\theta$ -subsumption. An atom set  $C$  subsumes an atom set  $D$ , denoted as  $C \geq D$ , if there is a substitution  $\theta$  such that  $C \theta \subseteq D$ . The  $\theta$ -subsumption relation induces an equivalence relation  $\sim$ , which is defined as follows:  $C \sim D$  iff  $C \geq D$  and  $D \geq C$ . For example, given the two queries  $Q_1$ :  $buys(X, Y), likes(X, cola)$  and  $Q_2$ :  $buys(Z, Z), likes(Z, cola)$ ,  $Q_2$  subsumes  $Q_1$  with  $\theta = \{X/Z, Y/Z\}$ , while  $Q_1$  does not subsume  $Q_2$ .

Although WARMR provides a sound theoretical basis for multi-relational ARM, it does not seriously address the efficiency of computation. In fact the runtime performance of WARMR depends heavily on the implementation of  $\theta$ -subsumption, and because  $\theta$ -subsumption is NP-complete, performance is poor [13].

FARMER makes two modifications to enhance the runtime performance of WARMR, the first to the data representation and the second by avoiding the need for  $\theta$ -subsumption. The first modification is to bind all mode declarations to, for example, Boolean procedures that return true or false for a given input. For example, consider the following facts:  $Fact(1, a, c)$  and  $Fact(1, j, e)$ . Given the mode  $Fact(+, \#, +)$ , a Boolean procedure would return true for the input  $(1, j, e)$  and  $(1, a, c)$ , and false for  $(1, a, e)$  and  $(1, j, c)$ . Furthermore, a mode could also be associated with a procedure that returns an output vector. For example, given the mode  $Fact(+, -, -)$  and the same input as above, the vector  $\{(a,c), (j,e)\}$  would be returned given the input ‘1’. Essentially, this supports optimization of the query unification process by predetermining the results of certain queries. Consequently, FARMER uses an entry in a multidimensional matrix for each mode declaration. Each element in the matrix corresponds to a set of input values and contains either a truth value or a list of output values. When a fact for a predicate is read, all corresponding mode elements are updated. The advantage of this mechanism lies in the constant time to determine the truth of an atom. The disadvantage is the memory requirement.

The second modification is in search. Under certain reasonable constraints, Nijssen and Kok [14] show that  $\theta$ -subsumption is unnecessary. As a result, FARMER does not perform  $\theta$ -subsumption. To search the knowledge base, FARMER maintains a trie structure to generate queries. Each path from the root to a node corresponds to a query. An example is given in Figure 1. There are for example seven 2-atom-set queries represented in this trie, each with three predicates (modes) in the Horn clause, including the common head,  $K(X)$ .

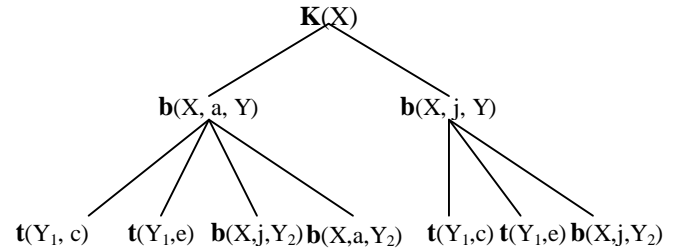


Figure 1: A trie in FARMER (adapted from [13])

The experimental results reported for FARMER in [14] indicate an improvement of a factor of 100 in runtime performance over WARMR.

In this section we have briefly introduced multi-relational association rule mining. With this background in hand, we are now prepared to introduce the DiHO Miner, a novel distributed multi-relational association rule mining algorithm.

## 4. The DiHO Miner

In this section, we present our DiHO ARM algorithm which makes use of the same kind of higher-order relations (used in multi-relational ARM algorithms) to mine rules from multiple distributed heterogeneous databases. The key to understanding the DiHO Miner is to view the distributed databases logically as a single relational database with multiple tables that have been distributed across multiple sites.

The DiHO Miner is a level-wise, bottom-up association rule mining algorithm. A global trie structure is maintained at each site and queries and their frequencies are communicated between sites. The algorithm is depicted in Figure 2 below.

- (1) Read local knowledge base
- (2) Broadcast type, predicate and mode declarations
- (3)  $T$  = a trie with only the key atom as the root
- (4) Repeat
- (5) Count (local freq)
- (6) Broadcast the frequencies and the key lists
- (7) Update ( $T$ )
- (8) Until  $T$  contains no uncounted queries
- (9) Generate rules from global frequent queries

**Figure 2: The DiHO ARM Algorithm**

In step 1, the local knowledge base, including threshold settings, site configuration information, predicates, modes and facts, is input by the algorithm. As DiHO is a CD-based algorithm, each site keeps a list of all possible query modes regardless of whether a particular mode exists locally. Thus, the trie structure is identical on all sites.

The identical trie structures are built based on the input predicate, mode and type information. This is why this information is broadcast in step 2. After forming the trie with the key atom as its root, iteration begins with the Repeat-Until loop in step 4. The next step is local counting of frequencies for those portions of the queries (i.e., modes) that exist on each respective local site in step 5. The frequencies for the queries are then broadcast to other sites in step 6 along with key lists, and each site updates its own identical trie structure  $T$  with the received frequencies in step 7. This loop repeats until all queries have been counted, at which point each site contains the global frequent queries with their supports and can proceed to generate rules. In the following subsection, we take a closer look at the communication entailed by DiHO ARM.

### 4.1 Issues in Message Passing

In CD-based distributed algorithms, each local site generates support counts and broadcasts them to all other sites, thereby enabling each site to calculate globally frequent itemsets. In DiHO ARM, however, broadcasting frequencies alone is insufficient. This is because queries cannot be resolved without knowledge of the key that links modes – in other words, a shared key is needed to make the higher-order ‘hop’ between distributed attribute values. As noted, the support of a query is calculated as the number of variable bindings of the key predicate. Thus, not only the count of bindings, but also the keys themselves are broadcast. For example, if the modes in query  $q$  that exist locally on site  $i$  are frequent, and the local frequency is  $f$ , then, the message to be passed for query  $q$  is:

$$M_q = \{q, f, \{id_1, id_2, \dots id_r\}\}$$

The minimization of communication costs is a major aim of distributed data mining. As a result, the DiHO Miner makes use of a key-bounding method in all operations with queries such as counting, evaluation, expansion, etc. This key-bounding method reduces communication by employing a monitor that tracks IDs associated with each mode in the trie. For a given mode on a given site, a list of IDs that match other sites is maintained. As messages arrive, any new IDs are matched against the list of existing IDs. IDs that fail to match are returned to the sender so that the sender’s monitor can prune those IDs from future messages sent to the site. In this way, the number of IDs communicated in any given message is minimized. In addition, a side effect of this key-bounding method is that messages are only broadcast among those sites that have data for the query associated with the message.

In particular, suppose that the trie in Figure 1 is formed from the mode declarations for two distributed databases. During the first level query (e.g., the level of the trie in Figure 1 immediately below the root  $K(X)$ ), only a single mode is resolved at each local site – these individual modes are each resolved locally. In Figure 1, this corresponds to the resolution of mode  $b(X,a,Y)$  on site 1 and mode  $b(X,j,Y)$  on site 2, both of which are children of the root  $K(X)$ . (Recall that after step 2 in the DiHO algorithm in Figure 2 completes, an identical copy of the trie exists at each distributed database location.) As execution proceeds along possible paths to the second level modes (e.g.,  $b(X,j,Y_2)$  in the left branch below  $b(X,a,Y)$  in Figure 1), a message  $M_q$  where  $q$  is the query “ $b(X,a,Y), b(X,j,Y_2)$ ” must be broadcast by site 1 to communicate to site 2 the frequent atoms and keys matching mode  $b(X,a,Y)$ . Site 2 resolves the query using its locally frequent atoms that matched mode  $b(X,j,Y)$ . Site 2 then responds by sending site 1 any frequent 2-atom-sets that satisfy both modes  $b(X,a,Y)$  and  $b(X,j,Y)$ . Execution proceeds in this way along all paths from root to leaves for each distributed database. Some of the modes may be found locally, some of them may not. If a given mode is locally available, the local site counts it and shares the resulting frequencies and keys with other sites as requested. If the local site is the only site where the mode exists, then no message passing is needed. For modes not locally available, the local site must broadcast the mode to obtain the frequencies and matching keys.

Other distributed ARM algorithms such as FDM and ODAM reduce communication costs by assigning one site to act as a server to calculate global frequencies. Up to the point where the central server becomes the bottleneck, the more sites participating in the computation, the greater the savings in communication. This kind of mechanism requires, however, that some sites rely on others (usually only one) to obtain the final ruleset. This dependency is avoided in our distributed algorithm as given in Figure 2. However, it is straightforward to design a version of DiHO ARM that employs a single server. In fact the choice of DiHO variant depends on the application – i.e., do all sites or only a few need the final ruleset? In the law enforcement example, all sites need the final ruleset.

### 4.2 Resolution of Object Identifiers

If the variables of the key predicate bound with the query are broadcast, our algorithm must guarantee that the variables are globally unique and consistent. To understand the need for uniqueness, consider for example the online paper databases

Axiom (Compendex®, INSPEC®) and Citeseer. In many cases these two databases will have the same object (a reference to a scholarly research article) with different attributes. To illustrate, consider this simple example. Suppose the Citeseer database has a table: {ID, title, year, citenum} where ID is a unique ID assigned to the article (e.g., a DOI), title is the title of the publication, year is the year of publication, and citenum is the number of citations to the article. Likewise, suppose that the Axiom database has a table: {ID, title, year, code} where the first three fields are similar in meaning to those in Citeseer, and the fourth contains one of the classification codes assigned to the article by abstract and indexing personnel. Suppose an information scientist wishes to know the citation rate of articles containing certain Axiom classification codes. If we assume that the two databases use the same global ID for articles, then in DiHO ARM each site will generate the local frequent 1-atom-set with the associated frequencies and exchange atom-sets and frequencies. The 2-atom-set {code citenum} can then be calculated using the global ID to match records in the two different databases – a higher-order association between code and citenum via the global ID.

A problem arises, however, when there is no guaranteed unique global ID for objects. In this particular example, in the absence of a DOI or URN, there may be no failsafe method to resolve object identity. It is reasonable to expect this kind of situation to arise fairly often, and steps must be taken to deal with it when it does. First, in the absence of a guaranteed globally unique ID, a decision must be made to approximate the ID. In this particular case, author names can be concatenated with the title to form a reasonable approximation of a globally unique ID. As a general solution we employ an edit-distance algorithm to match IDs formed in this way. Other special-purpose matching algorithms can be employed as well, such as that described in [17]. This does not guarantee, however, that for a given user-defined threshold, either the edit-distance or other special-purpose matching algorithms will predict matches correctly – both false positive and false negative mismatches are possible. Such mismatches will have an impact on both the support and confidence of the final rules. We deal with this issue further in section 5.

### 4.3 An Example Application of DiHO ARM

To further illustrate our algorithm, here is a simple example. Consider a situation in the law enforcement domain where multiple investigative reports from different jurisdictions detail different crimes committed by the same person. In this case, the criminal is the primary key (perhaps identified by SSN), and the various facts such as modus operandi that surround different crimes become the (vertically fragmented) attributes associated with the key. Let's suppose that our goal is to learn association rules that link the type of crime committed by an individual with some aspect of the modus operandi used in committing the crime (e.g., the type of weapon used). This kind of association rule can be very useful in narrowing the list of possible suspects to question about new criminal incidents<sup>2</sup>. However, as noted earlier, we have no guarantee in this case that both the crime type and weapon used will be recorded in a given investigator's record of an incident. This can result, for example, from incomplete (or inaccurate) testimony from witnesses. Thus DiHO ARM is applied to discover associations between crime type and weapon used in multiple jurisdictions' distributed databases.

<sup>2</sup> Because we are all creatures of habit, some good, some bad.

**Table 1. Relational Database on Site 1**

Criminal		Weapon	
ID		ID	Weapon
Allen		Allen	gun
Jack		Jack	knife
Carol		Carol	knife
Diana		Diana	hands
John		John	gun

**Table 2. Relational Database on Site 2**

Criminal		Crime	
ID		ID	Crime
Allen		Allen	Robbery
Jack		Jack	Robbery
Carol		Carol	Mugging
Bill		Bill	Burglary
John		John	Kidnapping

Given the two distributed databases depicted in tables 1 and 2 above, the site identifiers, key and mode declarations shown in table 3 are sufficient to launch the DiHO Miner. Table 4 is simply a view of the contents of the two distributed databases in the form of predicates.

**Table 3. Distributed Pattern Data**

Site 1	Site 2
SiteID (1).	SiteID (2).
Key (Criminal (-)).	Key (Criminal (-)).
Predicate (Criminal (ID)).	Predicate (Criminal (ID)).
Predicate (Weapon(ID, weapon)).	Predicate (Crime (ID, crime)).
Mode (Weapon (+, gun)).	Mode (Crime (+, Robbery)).
Mode (Weapon (+, knife)).	Mode (Crime (+, Kidnapping)).
Mode (Weapon (+, hands)).	Mode (Crime (+, Mugging)).
	Mode (Crime (+, Burglary)).

**Table 4. Distributed Databases**

Site 1	Site 2
Weapon (Allen, gun).	Crime (Allen, Robbery).
Weapon (Carol, knife).	Crime (Carol, Mugging).
Weapon (Diana, hands).	Crime (John, Kidnapping).
Weapon (John, gun).	Crime (Jack, Robbery).
Weapon (Jack, knife).	Crime (Bill, Burglary).

After all sites exchange local pattern data, each site will have a copy of the global pattern data which is the union of the local pattern data. This is depicted in table 5. Each site builds an identical trie structure based upon the global pattern data.

**Table 5. Global Pattern Data**

Site n	
SiteID (n)	
Key (Criminal (-))	Predicate(Weapon(ID,weapon)
Predicate(Criminal(ID))	Predicate (Crime (ID, crime))
Mode (Weapon (+, gun))	Mode (Crime (+, Robbery))
Mode (Weapon (+, knife))	Mode (Crime (+, Kidnapping))

Mode (Weapon (+, hands))	Mode (Crime (+, Mugging))
--------------------------	---------------------------

In this example, during the first iteration of the DiHO Miner, Site 1 generates frequent queries such as Criminal(A), Weapon (A, gun), while Site 2 generates frequent queries such as Criminal (A'), Crime (A', Kidnapping), etc. Table 6 depicts examples of the rules discovered on both sites.

**Table 6. Queries with frequency greater than 0.1**

Query begin with Criminal(A)	Support
Weapon (A, gun)	0.333333
Weapon (A, gun), Crime (A, Robbery)	0.166666
Weapon (A, gun), Crime (A, Kidnapping)	0.166666
Weapon (A, knife)	0.333333
Weapon (A, knife), Crime (A, Robbery)	0.166666
Weapon (A, knife), Crime (A, Mugging)	0.166666
Weapon (A, hands)	0.166666
Crime (A, Robbery)	0.333333
Crime (A, Mugging)	0.166666
Crime (A, Kidnapping)	0.166666
Crime (A, Burglary)	0.166666

Although this example is contrived, it is in fact based on work funded by the National Institute of Justice in modus operandi search of narrative textual police reports being conducted at Lehigh University [18]. We discuss this further in section 6.

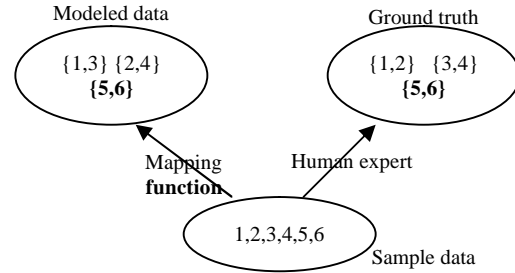
## 5. METRICS FOR EVALUATION

The methods described above for resolving object identifiers lead to another challenge – evaluation. One of the most important metrics used in ARM is *support*, which is defined as the frequency of an itemset divided by the total number of objects (instances). In distributed databases, the total number of unique objects can be calculated by counting the number of unique global IDs. As noted in section 4.2, the function used to map local keys to a unique global identifier is not guaranteed to be 100% accurate. Different source records can be incorrectly mapped to the same global ID; likewise, records that should be mapped to a single global ID can be mapped to different IDs. The error rate of the mapping function will thus influence both the support and confidence metrics. It will not suffice to calculate support and confidence in the traditional way employed in existing ARM algorithms. The error rate of the mapping function must be considered in the calculation of these metrics. To our knowledge, no similar work has been conducted that addresses this issue. In what follows, we present a novel evaluation method that incorporates an error rate into support. To simplify the presentation we use upper case letters to represent sets and lower case letters to represent single elements or sizes.

In a realistic ‘real-world’ data mining scenario where databases reach terabytes in size, it is infeasible to obtain the true error rate of the mapping function. Hence, we must rely on an estimate of the error rate obtained from a sample of the data for which all errors have been manually identified. This sample data is termed a *ground truth* because, for the sample, the actual error rate is known. As a result, in the analysis that follows we estimate the

true error rate as the upper bound of an assumed normally distributed observed error rate for a given confidence level<sup>3</sup>.

Given a sample data set  $R=\{r_1, r_2, \dots, r_k\}$  where  $r_i$  is a local database record, the ground truth data set  $G=\{g_1, g_2, \dots, g_n\}$  is a partition on  $R$ , where  $g_i \subseteq R$ ,  $\cap g_i = \mathbf{f}$  and each element  $g_i$  in  $G$  contains the records which map to a single object. On the other hand, the ID mapping performed on  $R$  will result in a partition  $P=\{p_1, p_2, \dots, p_n\}$ , where  $p_i \subseteq R$ ,  $\cap p_i = \mathbf{f}$  and each element  $p_i$  in  $P$  contains the records which map to a single object. This second mapping might incorrectly map two different records which are in fact unique objects into a single group, or conversely fail to map the records representing the same object into a single group.



**Figure 3: An Example for Mapping**

Figure 3 depicts a simple example that reveals the problem context. The sample data set is  $R=\{1,2,3,4,5,6\}$ , where records 1 and 2 represent the same object (e.g., have the same primary key), records 3 and 4 represent another object, and records 5 and 6 represent a third object. Thus, the ground truth data set  $G$  has 3 elements which are  $\{1,2\}$ ,  $\{3,4\}$  and  $\{5,6\}$ . Suppose the ID mapping function in this case also results in the three elements  $P=\{\{1,3\}, \{2,4\}, \{5,6\}\}$ . Clearly only one element in  $P$  is correct,  $\{5,6\}$ . The observed error rate is defined as the number of objects (i.e., objects in the ground truth data set) absent in the modeled data divided by the total number of objects (the size of the ground truth data set). In symbolic form the observed error rate  $f$  is:

$$f = 1 - \frac{|G \cap P|}{n}$$

We also define the observed difference degree  $t'$  as:

$$t' = \frac{n'}{n}, \quad t' > 0 \quad (1)$$

where  $n'$  is the size of the modeled data set and  $n$  is the size of the ground truth. Assuming that  $f$  is normally distributed, given a confidence level  $1-c$ , the probability that the normalized error rate is greater than  $z$  is:

$$\Pr \left[ \frac{f - e}{\sqrt{e(1-e)/n}} > z \right] = c,$$

where  $e$  is the true population error rate. We use the upper bound of the confidence interval to estimate the true population error rate  $e$  of the data set as follows:

<sup>3</sup> This technique is employed, for example, in the post-pruning stage of the popular C4.5 decision tree induction algorithm [20].

$$e = \frac{f + \frac{z^2}{2n} + z\sqrt{\frac{f}{n} - \frac{f^2}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

Similarly, the difference degree  $t$  is estimated in the same way. Having estimated both  $e$  and  $t$  based on sample data compared to the ground truth, the affect of the ID mapping function on support can be logically deduced as follows.

Given a set of real application data, the modeled data set  $D'$  can be obtained by applying the ID mapping function to the application data. The support rate for a given atom set on  $D'$  is denoted as  $s'$ .

Suppose  $D$  is the ground truth for the application data, which is unknown. To estimate the true support error rate  $s$ , we must estimate both the size of the ground truth data set as well as the number of objects supported (i.e., the support number).

Based on the definition of difference degree  $t$ , the size of the ground truth data set can be denoted as:

$$m = \frac{m'}{t}, \quad m = |D| \quad (2)$$

Consider the modeled application data  $D'$  as two parts: one part – denoted as  $T$  – containing all the correctly mapped objects; while the other part – denoted as  $E$  – is an incorrect partition over a set of records that represents the incorrectly mapped objects. Thus:

$$D' = T + E, \quad T \cap E = \mathbf{f}$$

The ground truth data can be represented in a similar way:

$$D = T + W, \quad T \cap W = \mathbf{f}$$

where  $W$  is the correct partition for the same set of records partition in  $E$ .

For the example shown in Figure 3, the modeled data set  $D'$  has three objects:  $\{1,2\}$ ,  $\{3,4\}$  and  $\{5,6\}$ , while the ground truth  $D = \{\{1,3\}, \{2,4\}, \{5,6\}\}$ .  $T = \{\{5,6\}\}$  because the modeled data correctly group the two records 5 and 6 into one object, as with in the ground truth.  $E = \{\{1,3\}, \{2,4\}\}$  and  $W = \{\{1,2\}, \{3,4\}\}$ . Clearly  $E$  and  $W$  are two different partitions over the same four records, and  $E \cap W = \mathbf{f}$ .

**Theorem 1** Given a set  $X = \{x_1, x_2, \dots, x_n\}$  where  $X \subseteq E$ ,  $|x_i| = 1$ ; set  $Y = \{y_1, y_2, \dots, y_v\}$  where  $Y \subseteq W$  and  $\cup x_i = \cup y_j$ , for  $\forall (y_j \in Y) |y_j| \geq 2$ .

**Proof.** Suppose  $\exists (y_j \in Y)$ ,  $y_j = \{r: |y_j| = 1\}$ . Since  $Y$  and  $X$  are two partitions on the same data set,  $\exists (x_i \in X)$  where  $r \in x_i$ . Since  $|x_i| = 1$ ,  $x_i = \{r: |x_i| = 1\}$ . We thus conclude that  $x_i = y_j$ , which is correctly mapped. This however violates the original assumption.

Let  $S'$  be the subset supporting a given atom set in  $D'$  (i.e.,  $s' = \frac{|S'|}{m'}$ ,  $|S'| \geq 1$ ), and  $S$  be the supporting subset in  $D$  (i.e.,  $s = \frac{|S|}{m}$ ). Furthermore, let

$$S' = S_T + S_E, \quad S_T \subseteq T, S_E \subseteq E$$

$$S = S_T + S_W, \quad S_T \subseteq T, S_W \subseteq W$$

where  $S_W$  is the correct partition over a subset of the data while  $S_E$  is an incorrect partition. Let's consider the following cases.

**Case 1.**  $S_E = \mathbf{f}$  and  $S_T = S'$

$S_E = \mathbf{f}$  means that all records in the application data are correctly mapped by the mapping function. Since  $S_W$  partitions the same data set as  $S_E$ ,  $S_W = \mathbf{f}$ . Thus, the true supporting data set can be derived as:

$$S = S_T + S_W = S' + \mathbf{f} = S'$$

As a result, the true support rate in this case is

$$s = \frac{|S_T|}{m} = \frac{|S'|}{m} = \frac{s' \cdot m'}{m} = t \cdot s'$$

**Case 2.**  $S_T = \mathbf{f}$  and  $S_E = S'$ .

$$S_E = S' \implies |S_E| = |S'| = s' \cdot m'. \quad (3)$$

Assuming that the estimated error rate  $e$  is the true error rate for the ground truth data set  $D$ , then we have:

$$|W| = e \cdot m \quad (4)$$

From equations (2) and (4), the size of  $E$  can be derived as:

$$\begin{aligned} |E| &= m' - |T| \\ &= m' - (m - |W|) \\ &= tm - (m - e \cdot m) \\ &= (t + e - 1)m, \quad t + e - 1 > 0. \end{aligned}$$

From equations (2) and (3), the size of  $|S_E|$  can be derived as:

$$|S_E| = s' \cdot m' = s' \cdot t \cdot m$$

A special situation can occur in which each of the objects in  $S_E$  only contains one record. The number of records in  $S_E$  would thus equal the number of objects in  $S_E$ , which is  $s' \cdot t \cdot m$ . According to Theorem 1, the size of each object in  $S_W$  must be at least two. The largest possible size of any object in  $S_W$  is the total number of records in  $S_E$ . This occurs when every record in  $S_E$  represents the same object. Thus, we conclude that:

$$1 \leq |S_W| \leq \lfloor s' \cdot t \cdot m / 2 \rfloor \quad (5)$$

In the general case when objects in  $S_E$  contain more than a single record, the bounds for  $S_W$  are:

$$1 \leq |S_W| \leq em \quad (6)$$

The lower bound is achieved when the records contained in all the elements in  $S_E$  represent the same object. Assuming that the remaining objects in  $E$  map to  $z$  objects in  $W$ ,  $em - z$  objects in  $W$  correspond to the objects in  $S_E$ . When  $S_E$  equals  $E$ ,  $z$  is 0. Thus the upper bound  $em$  is achieved.

Combining equations (5) and (6), we conclude that:

$$\begin{aligned} 1 \leq |S_W| &\leq \max(em, \lfloor s' \cdot t \cdot m / 2 \rfloor) \\ \implies \frac{1}{m} \leq s &\leq \max\left(\frac{em}{m}, \frac{\lfloor s' \cdot t \cdot m / 2 \rfloor}{m}\right) \end{aligned}$$

$$\Rightarrow \begin{cases} \frac{1}{m} \leq s \leq \max(e, \frac{s't}{2}), \text{if } m \text{ is even} \\ \frac{1}{m} \leq s \leq \max(e, \frac{s't(m-1)}{2m}), \text{if } m \text{ is odd} \end{cases} \quad (7)$$

Since  $m$  is large, we assume that  $\frac{m-1}{m} \approx 1$ , and thus equation (7) becomes:

$$s \in [\frac{1}{m}, \max(e, \frac{s't}{2})]. \text{ for a given } |S_E| = s'tm \quad (8)$$

**Case 3** Based on the previous two cases, we can now explore the true error rate in the general situation where

$$S' = S_T + S_E, S_T \subseteq T, S_E \subseteq E.$$

Let  $\alpha = |S_E| / |S'|$  and  $1-\alpha = |S_T| / |S'|$  where  $\alpha \in [0,1]$ . This implies that  $|S_T| = (1-\alpha) \cdot |S'| = (1-\alpha) \cdot s'm' = (1-\alpha) \cdot s'tm$  and that  $|S_E| = \alpha \cdot |S'| = \alpha \cdot s'm' = \alpha \cdot s'tm = (\alpha \cdot s')tm$ .

The true support rate  $s$  can thus be represented as:

$$\begin{aligned} s &= \frac{|S|}{m} = \frac{|S_T| + |S_W|}{m} \\ &= \frac{|S_T|}{m} + \frac{|S_W|}{m} \\ &= (1-\alpha)s't + \frac{|S_W|}{m} \\ &= -s't \cdot \mathbf{a} + s't + \frac{|S_W|}{m}, \mathbf{a} \in [0,1] \end{aligned}$$

Obviously,  $s(\mathbf{a})$  is a linear function for which the upper bound is reached when  $\mathbf{a}=0$  and the lower bound when  $\mathbf{a}=1$ . Thus:

$$s \in \left[ \frac{|S_W|}{m}, s't + \frac{|S_W|}{m} \right] \quad (9)$$

From equation (8), given that  $|S_E| = (\alpha \cdot s')tm$ ,

$$\frac{|S_W|}{m} \in \left[ \frac{1}{m}, \max(e, \frac{\mathbf{a} \cdot s't}{2}) \right].$$

Equation (9) can thus be expressed as:

$$\begin{aligned} s &\in \left[ \frac{1}{m}, s't + e \right] \\ \Rightarrow s &\in \left[ \frac{t}{m'}, s't + e \right] \end{aligned}$$

The lower bound for the support  $s$  on the ground truth implies that there is only one object supporting the atom set. This situation occurs when the records contained in all the objects in  $S_E$  represent a single object, which results in only one object in  $S_W$ . The probability of this occurring is not large. One way to improve the utility of the lower bound is to substitute  $t/m'$  for  $1/m$  because  $m'$  is a known value and  $t/m'$  may be larger than one. In fact, the utility of these bounds depends on the application. If the application needs demand a conservative estimate for support, the lower bound may be appropriate. For applications involving

extremely large data sets, especially in distributed databases with relatively low bandwidth connections, choosing a higher support rate will be extremely helpful in reducing the size of the atom sets generated.

Although this analysis represents to the best of our knowledge the first attempt to incorporate global ID mapping errors into the evaluation of distributed multi-relational ARM, clearly there is much work that remains. Specifically, we have yet to deal with the impact of errors in the ID mapping on the confidence metric, as well as the impact on the resulting rules. The approach we have taken, however, serves to blaze a trail for such future work.

## 6. CONCLUSIONS AND FUTURE WORK

We have presented the DiHO Miner, a novel distributed higher-order association rule mining (ARM) algorithm that mines vertically distributed data. The DiHO Miner is a first step towards tackling the difficult challenge posed by heterogeneous distributed databases that cannot be easily centralized. We have provided a description of the theoretical underpinnings of DiHO ARM based on multi-relational ARM. This is the first work to address the complex issues surrounding the use of the traditional support metric in the context of distributed higher-order ARM. Even so, this is just the beginning of the research task at hand and much of real interest remains to be accomplished.

In no particular order, we plan to address both theoretical and practical issues in areas such as schema integration, evaluation metrics and workload balancing. Although the DiHO Miner applies a key resolution method to identify globally unique IDs, currently the algorithm relies on the user to identify the database fields to be used in generating IDs. This process could be partially automated given semantic mappings such as those supported by the recently released Web Ontology Language [8]. A related challenge is the need for further work in the adaptation of metrics for distributed mining of vertically fragmented data. Our initial foray into this area serves primarily to highlight the need to address these issues in a rigorous manner within an overall framework for evaluation. We have taken the first steps in creating such a framework in the Text Mining Infrastructure (TMI), a software infrastructure for sequential, parallel and distributed textual data mining [12]. The DiHO Miner will be released initially in a Linux/MPI version packaged with the existing TMI at [hddi.cse.lehigh.edu](http://hddi.cse.lehigh.edu).

One of the more interesting open problems that has not been addressed in any previous work that we are aware of in distributed ARM deals with the efficiency of computation. The DiHO Miner too as presented is just a skeleton in this regard – a great deal of work is needed to deal with workload balancing as well as optimization of both computation and communication. Unbalanced workloads coupled with the requirements of synchronization will heavily affect the efficiency of the algorithm. This can be mitigated in part by the fact that the DiHO Miner can be used to mine association rules from the results returned from a search (as opposed to mining rules from entire databases). For example, in the Axiom/CiteSeer example given earlier, a user may wish to find strong associations between Axiom's classification codes and the number of citations on CiteSeer for a particular set of documents returned by a query. Furthermore, from a different perspective, multilevel parallelism can be introduced to improve runtime performance: i.e., at each distributed database, a parallel ARM algorithm can be executed. This also leads to the need for time and space complexity analyses based on metrics such as isoefficiency.

Foremost in our thoughts, however, is a plan to deploy the DiHO Miner in a law enforcement environment. We have done much

work in preparation for such a deployment. In [18], we detail our work in information extraction from narrative textual data sources. In [19] we describe the design of a system based on the theory developed in [18] that has recently been deployed in the Bethlehem, Pennsylvania Police Department. This system uses advanced information extraction techniques to mine modus operandi data from the narrative text of police investigator's reports. The extracted data populates a relational database, thereby enabling straightforward modus operandi search.

The next phase of our research will build on this system. It involves the development and deployment of the DiHO Miner in the Northampton County, Pennsylvania region. Northampton County, Pennsylvania has 32 independent police jurisdictions, none of which share modus operandi data on a systematic basis. The County has, however, recently implemented a high-bandwidth networking infrastructure that supports secure communication amongst all 32 jurisdictions. It is our plan to deploy the DiHO Miner in this environment, with distributed higher-order modus operandi association rule mining and search as our first application. In this way we will take a step towards realizing the "System of Systems" envisioned by the US Department of Justice.

## 7. ACKNOWLEDGEMENTS

The authors wish to thank Lehigh University, the Pennsylvania State Police, the Lockheed-Martin Corporation, the City of Bethlehem Police Department and the National Institute of Justice, Office of Justice Programs, US Department of Justice. This work was supported in part by NIJ grant number 2003-IJ-CX-K003. Points of view in this document are those of the authors and do not necessarily represent the official position or policies of Lehigh University, the US Department of Justice, the Pennsylvania State Police or the Lockheed Martin Corporation.

We are also grateful for the help of Tianhao Wu, Tim Cunningham, other co-workers, family members and friends. We also gratefully acknowledge the continuing help of our Lord and Savior, Yeshua the Messiah (Jesus the Christ) in our lives and work. Amen.

## 8. REFERENCES

- [1] Agrawal R., Imielinske T., and Swami A. N. Mining association rules between sets of items in large databases. In *Proc. of the 1993 ACM SIGMOD Int'l. Conference on Management of Data*, pages 207-216, Washington, D.C., June 1993.
- [2] Agrawal R., Mannila H., Srikant R., Toivonen H., and Inkeri Verkamo A. Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307-328. AAAI/MIT Press, 1996.
- [3] Agrawal R. and Shafer J. C. Parallel Mining of Association Rules. *IEEE Trans. Knowledge and Data Eng.*, vol. 8, no. 6, 1996, pp. 962-969.
- [4] Ashrafi M. Z., Taniar D., and Smith K. ODAM: an Optimized Distributed Association Rule Mining Algorithm. *IEEE Distributed Systems*, vol. 5, No 3, March 2004.
- [5] Boyd D., Director of the Department of Homeland Security's new Office of Interoperability and Compatibility, in a presentation at the Technologies for Public Safety in Critical Incident Response Conference and Exposition 2004, New Orleans, LA, September.
- [6] Cheung D. W., Han J., Ng VAT., Fu AWE. and Fu Y. A Fast Distributed Algorithm for Mining Association Rules. *Proc. Parallel and Distributed Information Systems*, IEEE CS Press, 1996, pp. 31-42.
- [7] Cheung D. W., Ng V. T., Fu A. W. and Fu Y. Efficient Mining of Association Rules in Distributed Databases. *IEEE Trans. Knowledge and Data Eng.*, vol. 8, no. 6, 1996, pp. 911-922.
- [8] Dean M. and Schreiber G. OWL Web Ontology Language Reference. Editors, W3C Recommendation, 10 February 2004. [Online Article]. Retrieved Nov. 17, 2004 from the World Wide Web: <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>.
- [9] Dehaspe L. and Raedt L. D. Mining association rules in multiple relations. In S. Dzeroski and N. Lavrac, editors, *Proceedings of the 7<sup>th</sup> International Workshop on Inductive Logic Programming*, volume 1297, pages 125-132. Springer-Verlag, 1997.
- [10] GJXDM. Global Justice XML Data Model. [Online Article]. Retrieved Nov. 17, 2004 from the World Wide Web: <http://www.it.ojp.gov/gjxdm>
- [11] Grama A., Gupta A. and Kumar V. Isoefficiency function: a scalability metric for parallel algorithms and architectures. *IEEE Parallel and Distributed Technology*, vol. 1, no. 3, pp. 12-21, 1993.
- [12] Holzman, L.E., Fisher, T.A., Galitsky, L. M., Kontostathis, A., and Pottenger, W. M. A Software Infrastructure for Research in Textual Data Mining. *The International Journal on Artificial Intelligence Tools*. 2004. (In press)
- [13] Kietz J-U. and Lubbe M. An efficient subsumption algorithm for inductive logic programming. In S. Wrobel, editor, *Proceedings of the 4<sup>th</sup> International Workshop on Inductive Logic Programming*, volume 237, pages 97-106. Gesellschaft fur Mathematik und Datenverarbeitung MBH, 1994.
- [14] Nijssen S. and Kok J. Faster Association Rules for Multiple Relations. In *IJCAI01*, Seattle, Washington, USA, pages 891--896, 2001.
- [15] Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Francisco: Morgan Kaufmann.
- [16] Schuster A. and Wolff R. Communication-Efficient Distributed Mining of Association Rules. *Proc. ACM SIGMOD Int'l conf. Management of Data*, ACM Press, 2001, pp. 473-484.
- [17] Wang, G., Chen, H. and Atabakhsh H. Automatically Detecting Deceptive Criminal Identities. *Communications of the ACM*, 47 (3): 71-76, 2004.
- [18] Wu, T. and Pottenger, W. M. A Semi-Supervised Active Learning Algorithm for Information Extraction from Textual Data. *Journal of the American Society for Information Science and Technology*, JASIST. 2004. (In press)
- [19] Wu, T. and Pottenger, W. M. A Software System for Information Extraction in Criminal Justice Information Systems. Technical Report LU-CSE-04-009, Lehigh University, Bethlehem, PA, July 2004. [Online Article]. Retrieved from the World Wide Web: [http://www3.lehigh.edu/images/userImages/jgs2/Page\\_3813/LU-CSE-04-009.pdf](http://www3.lehigh.edu/images/userImages/jgs2/Page_3813/LU-CSE-04-009.pdf)
- [20] Zaki M. J. Parallel and Distributed Association Mining: A Survey. *IEEE Concurrency*, Oct.-Dec. 1999, pp. 14-25.