01 Jul 2006

Attribute-level Neighbor Hierarchy Construction Using Evolved Pattern-based Knowledge Induction

Thanit Puthpongsiriporn

J. David Porter

Bopaya Bidanda

Ming En Wang

et. al. For a complete list of authors, see https://scholarsmine.mst.edu/mec_aereng_facwork/5197

Recommended Citation

T. Puthpongsiriporn et al., "Attribute-level Neighbor Hierarchy Construction Using Evolved Pattern-based Knowledge Induction," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 7, pp. 917 - 929, article no. 1637418, Institute of Electrical and Electronics Engineers, Jul 2006.
The definitive version is available at https://doi.org/10.1109/TKDE.2006.104

Attribute-Level Neighbor Hierarchy Construction Using Evolved Pattern-Based Knowledge Induction

Thanit Puthpongsiriporn, J. David Porter, Bopaya Bidanda, Ming-En Wang, and Richard E. Billo

Abstract—Neighbor knowledge construction is the foundation for the development of cooperative query answering systems capable of searching for close match or approximate answers when exact match answers are not available. This paper presents a technique for developing neighbor hierarchies at the attribute level. The proposed technique is called the evolved Pattern-based Knowledge Induction (ePKI) technique and allows construction of neighbor hierarchies for nonunique attributes based upon confidences, popularities, and clustering correlations of inferential relationships among attribute values. The technique is applicable for both categorical and numerical (discrete and continuous) attribute values. Attribute value neighbor hierarchies generated by the ePKI technique allow a cooperative query answering system to search for approximate answers by relaxing each individual query condition separately. Consequently, users can search for approximate answers even when the exact match answers do not exist in the database (i.e., searching for existing similar parts as part of the implementation of the concepts of rapid prototyping). Several experiments were conducted to assess the performance of the ePKI in constructing attribute-level neighbor hierarchies. Results indicate that the ePKI technique produces accurate neighbor hierarchies when strong inferential relationships appear among data.

Index Terms—Approximate query answering, clustering, knowledge discovery, query-answering systems, similarity measures.

1 Introduction

A conventional database querying system requires an "exact match" between the query conditions and the values of the returned attributes. When a query is overspecified (i.e., none of the existing tuples can satisfy all the query conditions), a null answer is returned. This exact match property of a typical query answering mechanism translates into user frustration, especially when a negative answer is not expected.

A cooperative query answering system, on the other hand, offers approximate or partial answers even when all the conditions of the submitted query cannot be matched exactly. Instead of returning null answers, the cooperative query answering system searches for the neighbors of the unavailable exact-match answers [1], [2] using artificial intelligence mechanisms and rules or facts from the existing and/or supplemental knowledge developed by application domain experts.

The problem of finding approximate answers was first introduced by Cole and Morgan and was motivated by applications in intelligent querying [3]. Since then, many cooperative query answering techniques have been developed to improve information retrieval. These systems have been incorporated (at various degrees of implementation) in many application domains, such as biological, agricultural, and health care [4], [5], [6].

The information retrieval system must possess the ability to loosen overspecified conditions of a query to find approximate or close match answers. Such a process is called *query relaxation*. Query relaxation through attribute value substitutions is carried out by replacing the attribute values of the query selection conditions (i.e., the constants of query conditions) that are evaluated as FALSE with other attribute values until approximate answers are returned. The value used in an attribute substitution can be any value that is included in the domain of the attribute of the selection condition. Obviously, to obtain the best set of approximate answers (i.e., the alternative answers that are closer to the unavailable exact matches), attribute values must be substituted with their close neighbors as opposed to any values in the domain. Distant neighbors of an attribute value should be used in attribute value substitutions only when all of the closer neighbors fail to produce any partial answer.

Before a systematic search for approximate answers by attribute value substitutions can be performed, values in attribute domains must be organized so that different levels of neighboring relationships among the attribute values (e.g., close neighbors, related neighbors, and remote neighbors) can be easily accessed when they are needed. The most accepted way of representing this type of neighbor information of attribute values is through *attribute value neighbor hierarchies*. An attribute value neighbor

- *T. Puthpongsiriporn and J.D. Porter are with the Department of Industrial and Manufacturing Engineering, Oregon State University, 118 Covell Hall, Corvallis, OR 97331-2407. E-mail: {thanit, david.porter}@orst.edu.*
- *B. Bidanda is with the Department of Industrial Engineering, University of Pittsburgh, 1048 Benedum Hall, Pittsburgh, PA 15261. E-mail: bidanda@engr.pitt.edu.*
- *M.-E. Wang is with the Department of Industrial Engineering, University of Pittsburgh, 1034 Benedum Hall, Pittsburgh, PA 15261. E-mail: mewang@engr.pitt.edu.*
- *R.E. Billo is with the College of Engineering, University of Texas at Arlington, Arlington, TX 76019. E-mail: richard.billo@uta.edu.*

Manuscript received 3 Feb. 2004; revised 7 Apr. 2005; accepted 24 Jan. 2006; published online 18 May 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0037-0204.

1041-4347/06/$20.00 © 2006 IEEE Published by the IEEE Computer Society

TABLE 1
Relation ALKUM_PLATE

	STOCK_NO	WIDTH	LENGTH	AREA	THICKNESS
1	s_1	$\boldsymbol{w_1}$	$\boldsymbol{l_1}$	$\boldsymbol{a_1}$	t_1
2	s_2	$\boldsymbol{w_1}$	$\boldsymbol{l_1}$	$\boldsymbol{a_1}$	t_2
3	s_3	w_2	$\boldsymbol{l_1}$	a_2	t_3
4	s_4	w_3	$\boldsymbol{l_1}$	a_3	$\boldsymbol{t_4}$
5	s_5	w_4	$\boldsymbol{l_1}$	a_4	$\boldsymbol{t_4}$
6	s_6	w_5	$\boldsymbol{l_1}$	a_5	t_5
7	s_7	$\boldsymbol{w_1}$	$\boldsymbol{l_1}$	$\boldsymbol{a_1}$	t_6

hierarchy indicates how similar each attribute value is to the other values.

Neighbor knowledge construction is the foundation for the development of cooperative query answering systems capable of searching for close-match or approximate answers. Consider the entities in relation ALUM_PLATE shown in Table 1. Each entity in the relation has five attributes—Stock Number, Width, Length, Area, and Thickness. A stock number is unique and is therefore used as the primary key to identify each record. The values of attributes Width, Length, Area, and Thickness are nonunique because they are shared among multiple tuples (shown in bold in Table 1). The higher the number of unique attribute values associated with an entity, the less similar the entity becomes. Stocks s_1, s_2, and s_7 share three common attribute values (w_1, l_1, and a_1). Thus, one may say that they are relatively closer together than to the other four stocks. Stocks s_4 and s_5 share two common attribute values (l_1 and t_4). Therefore, s_4 and s_5 could also be considered similar. However, the similarity between s_4 and s_5 should be less than the similarity among s_1, s_2, and s_7. All seven tuples share a common attribute value l_1.

Fig. 1 depicts one possible neighbor hierarchy of stocks s_1 to s_7 based on the number of shared attribute values. In this case, the neighbor hierarchy was constructed based on the characteristic "stock dimensions." A node indicates a neighbor relationship at a certain degree of similarity between two tuples, a tuple and a set of tuples, or two sets of tuples.

Many techniques for neighbor hierarchy construction have been proposed in the literature. Yu and Sun [7], Han et al. [8], Han and Fu [9], Puthpongsiriporn [10], and Merzbacher [11] presented a technique called Pattern-based Knowledge Induction (PKI) as a tool to discover data correlations and inferential relationships among data instances in a relation. PKI can be used to generate attribute neighbor hierarchies (which are essential for the query relaxation process used to find approximate answers) when coupled with the Binary Clustering algorithm. However, PKI constructs neighbor hierarchies at the tuple level and takes into consideration only existing combinations of attribute values, thus facilitating approximate answer searching only for those tuples that already exist at the time in which the hierarchy is developed. Therefore, if a query is issued to retrieve a nonexisting tuple such as an aluminum stock with $\text{WIDTH} = w_1$, $\text{LENGTH} = l_1$, and $\text{THICKNESS} = t_5$, the neighbor hierarchy depicted in Fig. 1 will fail to generate approximate answers. Through visual inspection, the closest neighbors of the required aluminum stock are stocks s_6 (with $\text{WIDTH} = w_5$, $\text{LENGTH} = l_1$, and $\text{THICKNESS} = t_5$), s_1, s_2, and s_7 (with $\text{WIDTH} = w_1$, $\text{LENGTH} = l_1$, and $\text{THICKNESS} = t_1$, t_2, and t_6, respectively). Evidently, to determine the approximate answer that is closer to the exact-match answer, the system needs to be able to compare the semantic nearness between $\text{WIDTH} = w_1$ and $\text{WIDTH} = w_5$, and $\text{THICKNESS} = t_5$ and $\text{THICKNESS} = t_1$, t_2, and t_6. Thus, finding approximate answers for a nonexisting tuple requires a neighbor hierarchy construction technique that is applicable at both the tuple and attribute levels.

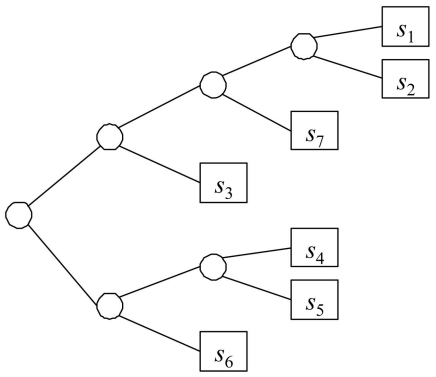

Fig. 1. Neighbor hierarchy of members of attribute STOCK_NO.

In this paper, a novel technique for neighbor knowledge construction based on the original PKI is proposed. This technique, referred to as *Evolved Pattern-based Knowledge Induction* (ePKI), has several advantages over the original PKI:

- It can be used to construct attribute value neighbor hierarchies at both the tuple and attribute levels utilizing inferential relationships among attributes in the relation.
- The resulting attribute-level neighbor hierarchies allow for a search of approximate answers by relaxing each individual query condition separately. Therefore, query relaxations are not limited to a certain set of standard or predefined queries.
- Finally, the proposed neighbor hierarchy construction permits approximate answer searching for tuples that did not exist in the relation when the neighbor hierarchy was first created.

2 Related Work

Relaxing queries through attribute value substitutions requires that the database management system possesses attribute value neighbor information. These sets of neighbor knowledge are generally stored in the database in the form of abstraction hierarchies. Neighbors of an attribute value could vary from one application to another as opposed to the domain of an attribute value that stays the same within the database. This is because different applications may have different abstract meanings for the same attribute value. In addition, different applications may require different resolutions or precisions for their attribute value neighbor knowledge. For example, the

neighbors of "Thursday" could be {"Wednesday," "Friday"} based on the day order for a particular application, or they could be {"Sunday," "Tuesday"} based on the alphabetical order in another application.

In the early years of cooperative query answering research, the creation and updating of abstraction hierarchies was mostly performed by application experts. However, as the databases quickly grew larger, it was impossible to keep up with the changes in their structure and stored data. Many techniques have been proposed in the literature that could be used to automate the two activities considered as the main challenges of using abstraction hierarchies: the development and the subsequent maintenance of the hierarchies [7], [8], [9].

Huang [12] used various knowledge discovery tools to develop the knowledge base for an intelligent query answering system including generalization, data summarization, concept clustering, rule discovery, query rewriting, lazy evaluation, and semantic query optimization. Ozawa and Yamada [13] used fuzzy logic techniques in conjunction with a category utility to construct concept hierarchies of data tuples to generate cooperative answers. Their methods require that the predefined linguistic labels of the background knowledge for the data attributes, which were used in classification of the data, be provided.

A more popular method used to construct neighbor abstraction hierarchies for numeric values is *information entropy* [14], which involves the evaluation of clustering results using *entropy measure* and its main objective is to maximize the entropy of data partitions [15], [16]. Since the entropy is maximized when the data are partitioned evenly, this approach considers only the frequency of each attribute value. The value distribution of the data is excluded from this method. Another method for developing neighbor hierarchies of numerical values was proposed by Genari et al. [17]. The proposed method, called CLASSIT, uses the mean and standard deviation to classify numerical values into classes. The original goodness measure used in their work was $1/\sigma$, but this was further revised to σ to resolve the goodness value problem for single member clusters. These two numerical value clustering methods consider only the frequencies of values. Chu et al. [18], [19] presented a method to automatically generate abstraction hierarchies for numerical attribute values by considering both frequency and value distribution of the data. They developed a clustering measure called *relaxation error* that allowed them to optimize their clustering results.

Finally, Merzbacher and Chu [20] developed a method to generate abstraction hierarchies for both numerical and categorical attribute values using PKI and the Binary Clustering techniques. PKI is a tool to discover the data correlations and inferential relationships among data instances in a relation and can be used to perform binary clustering of both discrete and continuous attribute values. The algorithm not only generates hierarchies of clusters, but also calculates nearness values between neighbors, which are essential for the query relaxation process used to find approximate answers. However, PKI constructs neighbor hierarchies at the tuple level and takes into consideration only the existing values of the attributes to do so. Thus, the resulting neighbor hierarchies cannot be used for nearness measuring or transformation of a selection condition that contains a value that does not exist at the time in which the hierarchy is developed.

3 Preliminaries

Similar to association rule development, the ePKI technique derives inferential relationships between two values of two different attributes based on *confidence* and *popularity* of occurrences in which the two attribute values coexist in the same tuple. In order to formalize the concept of inferential relationship first, a *pattern* is defined as the abstract representation for a group of database instances with a specified property. A pattern is expressed by an atomic query condition. The formal representation of a pattern P is $P_C = P : D \rightarrow T$, where C is an atomic query condition that expresses P, D is the domain of a relation, and T is a set of tuples that satisfies the condition C. For example, if D defines all tuples in Table 1 and pattern P is defined with condition "WIDTH $= w_1$," then $P_{WIDTH=w_1}$ refers to tuples 1, 2, and 7. In other words, $P_{WIDTH=w_1} : D \rightarrow \{s_1, s_2, s_7\}$. By the definition of a pattern, a relation instance can be a member of more than one abstract pattern class. Tuple 1 is a member of patterns $P_{STOCK\_NO=s_1}$, $P_{WIDTH=w_1}$, $P_{LENGTH=l_1}$, $P_{AREA=a_1}$, and $P_{THICKNESS=t_1}$.

Second, the inferential relationship between attribute values v_1 and v_2 is defined as the likelihood that if v_1 is the value of attribute A of a tuple, v_2 is also the value of attribute B of the same tuple. The inferential relationship between attribute values can be defined based on the subsumption property of one pattern to another. In Table 1, every time WIDTH $= w_1$ is observed, one may expect the value of attribute LENGTH to be l_1. However, the value l_1 of attribute LENGTH does not strongly imply the value w_1 of attribute WIDTH. The usefulness of an inferential relationship is measured using the confidence and the popularity of the relationship. Inferential confidence is used for determining the accuracy of the relationship and is defined based on the cardinalities of the patterns. The cardinalities of patterns A and B are denoted by $|P_A|$ and $|P_B|$, and are defined as the number of distinct data instances (i.e., tuples) matching the conditions of patterns A and B, respectively. The inferential confidence of the inferential relationship $A \rightarrow B$ is defined as:

$$\xi(A \rightarrow B) = \frac{|P_A \cap P_B|}{|P_A|}. \quad (1)$$

$\xi(A \rightarrow B)$ can assume any value between 0 and 1, inclusive. A confidence value of 0 indicates that there is no inferential relationship between patterns A and B, whereas a confidence value of 1 means that the relationship between patterns A and B is deterministic. Equation (1) assumes that set T always has one or more members. In other words, there is at least one tuple in the domain D that satisfies the condition C; therefore, $P_A > 0$. One can view inferential confidence of the relationship $A \rightarrow B$ as the conditional probability that pattern B is true (i.e., occurs) given that pattern A is true. For the Table 1 example, the confidence of the inferential relationship from w_1 to l_1 is

$$\xi(P_{WIDTH=w_1} \rightarrow P_{LENGTH=l_1}) = \frac{|P_{WIDTH=w_1} \cap P_{LENGTH=l_1}|}{|P_{WIDTH=w_1}|} = \frac{3}{3} = 1,$$

while the confidences of the inferential relationships from l_1 to w_1, l_1 to w_2, and l_1 to w_3 are .429, .143, and .143, respectively.

If the premise and/or the consequence are sets of values, the confidence of the inferential relationship could be computed as follows:

$$\xi(\{A_1, A_2\} \rightarrow B) = \frac{|(P_{A_1} \cup P_{A_2}) \cap P_B|}{|(P_{A_1} \cup P_{A_2})|} = \frac{|(P_{A_1} \cap P_B) \cup (P_{A_2} \cap P_B)|}{|(P_{A_1} \cup P_{A_2})|}$$

$$\xi(\{A_1, A_2\} \rightarrow \{B_1, B_2\}) = \frac{|(P_{A_1} \cup P_{A_2}) \cap (P_{B_1} \cup P_{B_2})|}{|(P_{A_1} \cup P_{A_2})|}.$$

Popularity, as the second measure of an inferential relationship, indicates how common a relationship is compared to others and is defined as:

$$\eta(A \rightarrow B) = \frac{|P_A \cap P_B|}{|C|}, \quad (2)$$

where $|C|$ is the total cardinality of class C.

By requiring that the popularity of an inferential relationship exceeds some threshold, the system can discard any inferential relationships that have low occurrence frequencies, i.e., $\xi(l_1 \rightarrow w_2)$, which do not provide strong evidence of true relationships. Only inferential relationships having confidence and popularity values exceeding some predefined thresholds are accepted and stored in the knowledge base. Those relationships that do not meet the threshold are discarded. Using a high threshold value results in fewer number of quantified inferential relationships and translates into less computation time to calculate neighbor hierarchies. Choosing a low threshold value, on the other hand, may result in passing on some important inferential relationships.

Once a set of inferential relationships is acquired, PKI can be used to create a clustering hierarchy for attribute values based upon the following rule of shared consequence:

If two inferential relationships share a consequence and have the same attribute as a premise (but different values), then those values are candidates for clustering [11].

The rule of shared consequence views the values in the other attributes as "characteristics" of the two values. In essence, two values from the same attribute (premise) should be grouped together if both have the same corresponding attribute value (consequence) in other attributes. Stocks s_1 and s_2 could potentially be grouped together since they both share the same consequence when only STOCK_NO and WIDTH are considered.

3.1 Clustering Correlation Equation

A considerable amount of research has concentrated on the development of techniques aimed at quantifying the dissimilarity (or similarity) between two continuous or discrete probability distributions defined for a common range of values [21], [22], [23], [24], [25]. The family of dissimilarity coefficients is defined as the expected value of a continuous convex function ϕ of the likelihood ratio $r = p(x)/q(y)$, $E_P[\phi(r)]$, where $\phi(1) = 0$. It obeys the discriminating property, according to which the measure of divergence between the two discrete events should not decrease for any refinement of both the discrete events and their two distributions [26].

For the purpose of this research and by using the rule of shared consequence, the clustering correlation equation is defined as the product of the confidence of the two inferential relationships and it is expressed as:

$$\gamma(a_1, a_2) = \xi(A = a_1 \rightarrow B = b) \times \xi(A = a_2 \rightarrow B = b). \quad (3)$$

Based on (3), the correlation of clustering s_1 and s_2 based solely on the WIDTH attribute is

$$\gamma(s_1, s_2) = \xi(s_1 \rightarrow w_1) \times \xi(s_2 \rightarrow w_1) = 1.$$

In an m-attribute relation, the clustering correlation between any two values of an attribute based upon the shared consequences from the rest of the attributes in the relation is then defined as:

$$\gamma(a_1, a_2) = \sum_{i=1}^{m} \xi(A = a_1 \rightarrow B_i = b_i) \times \xi(A = a_2 \rightarrow B_i = b_i). \quad (4)$$

From (4), the clustering correlation between attribute values a_1 and a_2 of attribute A is equal to 1 if these have the same corresponding attribute value (b_i) for each other attribute in the table (B_i where $i = 1$ to m). In that case, attribute values a_1 and a_2 of attribute A should be clustered together. Equation (4) possesses some interesting properties:

1. It can be determined for all pairs of attribute values.
2. It is symmetric, thus $\gamma(a_1, a_2) = \gamma(a_2, a_1)$.
3. Neighbor hierarchies generated using ePKI and the binary clustering algorithm obey the discriminating property as long as the threshold (i.e., the minimum normalized clustering correlation value) is chosen appropriately. If the selected threshold is too low, it is possible that this property will not hold. One way to avoid such incidence is to allow only the pair(s) of values (or sets of values) with the highest clustering correlation in each iteration to be grouped. Although it takes longer time to generate a neighbor hierarchy, this approach guarantees that the discriminating property is always achieved. In such a case, the use of a threshold is irrelevant.
4. The proposed clustering correlation yields its maximum value when $|P_{a_1}| = |P_{a_1} \cap P_{B_i=b_j}|$ and $|P_{a_2}| = |P_{a_2} \cap P_{B_i=b_j}|$ for all consequence attributes (i) and consequence attribute values (j). On the other hand, the correlation takes its minimum value when $P_{(a_1 \cap B_i=b_j) \cap (a_2 \cap B_i=b_j)} : D \rightarrow \phi$ for all i and j.

Based on (4), the clustering correlation of (s_1, s_2) is

$$\gamma(s_1, s_2) = \begin{bmatrix} \xi(s_1 \rightarrow w_1) \times \xi(s_2 \rightarrow w_1) \\ +\xi(s_1 \rightarrow l_1) \times \xi(s_2 \rightarrow l_1) \\ +\xi(s_1 \rightarrow a_1) \times \xi(s_2 \rightarrow a_1) \\ +\xi(s_1 \rightarrow t_1) \times \xi(s_2 \rightarrow t_1) + \xi(s_1 \rightarrow t_2) \times \xi(s_2 \rightarrow t_2) \end{bmatrix} = (1 + 1 + 1 + 0 + 0) = 3.$$

Notice that introducing factors $\xi(s_1 \rightarrow w_2) \times \xi(s_2 \rightarrow w_2), \ldots, \xi(s_1 \rightarrow w_5) \times \xi(s_2 \rightarrow w_5)$ in the equation to calculate the clustering correlation of (s_1, s_2) does not alter the final value of $\gamma(s_1, s_2)$ since the premises come from a unique attribute (i.e., an attribute with nonduplicate values). From the fact that the maximum value of the clustering correlation formula is $m-1$, the normalized clustering correlation is:

$$\bar{\gamma}(a_1, a_2) = \frac{1}{m-1}\gamma(a_1, a_2), \tag{5}$$

where m represents the number of attributes in the relation.

The normalized clustering correlation of (s_1, s_2) is $\bar{\gamma}(s_1, s_2) = 1 \div (5-1) \times 3 = 0.75$. Such correlation can be calculated for each possible pair of values in attribute STOCK_NO, i.e., $(s_1, s_3), \ldots, (s_6, s_7)$. Notice that there is no need to calculate (s_2, s_1) once (s_1, s_2) is obtained since $\bar{\gamma}(a_1, a_2) = \bar{\gamma}(a_2, a_1)$. One could choose to form groups of pairs of values if their clustering correlations are relatively high compared to those of other possible groupings. Once a cluster is formed, the same process can be repeated again to form more and/or larger groups. In the second or later iteration, clustering correlation calculations are performed for: 1) two individual values that have not been clustered with other values, 2) a nonclustered value and a set of two grouped values, and 3) two sets of clustered values. If a set and a value are considered for grouping, their clustering correlation is calculated as follows:

$$\gamma(S_1, a_3) = \gamma(\{a_1, a_2\}, a_3) = \sum_{i=1}^{m} \xi(A = \{a_1, a_2\} \rightarrow B_i = b_i) \times \xi(A = a_3 \rightarrow B_i = b_i). \tag{6}$$

The correlation value of clustering two sets of values can be calculated using the same approach. Group formation could continue until all values are contained in one large group or none of the available cluster possibilities has a high enough value for clustering correlation.

Merzbacher [11] proposed the use of the Binary Clustering algorithm in conjunction with the normalized clustering correlation function for neighbor hierarchy constructions. The binary clustering algorithm earned its name from the fact that each individual value (or set of values) is allowed to form a new (larger) cluster by combining with one and only one other value (or set of values) once at any iteration. The algorithm simply starts with determining and ranking $\bar{\gamma}$ for all possible clustering pairs, i.e., a value with another value, a value with a set of values, and two sets of values. Then, for each pair with $\bar{\gamma}$ greater than a threshold, form a new cluster, expand an existing cluster, or combine two clusters together. In the next iteration, the new clusters then become new clustering candidates with another value or set of values. For example, s_7 may be grouped with s_3 in the first iteration, despite the fact that it is more similar to s_1 and s_2.

3.2 Illustrative Example

An example of a neighbor hierarchy generated using PKI and the Binary Clustering algorithm is illustrated in Fig. 2. In this example, a Binary Clustering algorithm is performed on the attribute STOCK_NO to construct a neighbor hierarchy for the set of aluminum plates shown in Table 2.

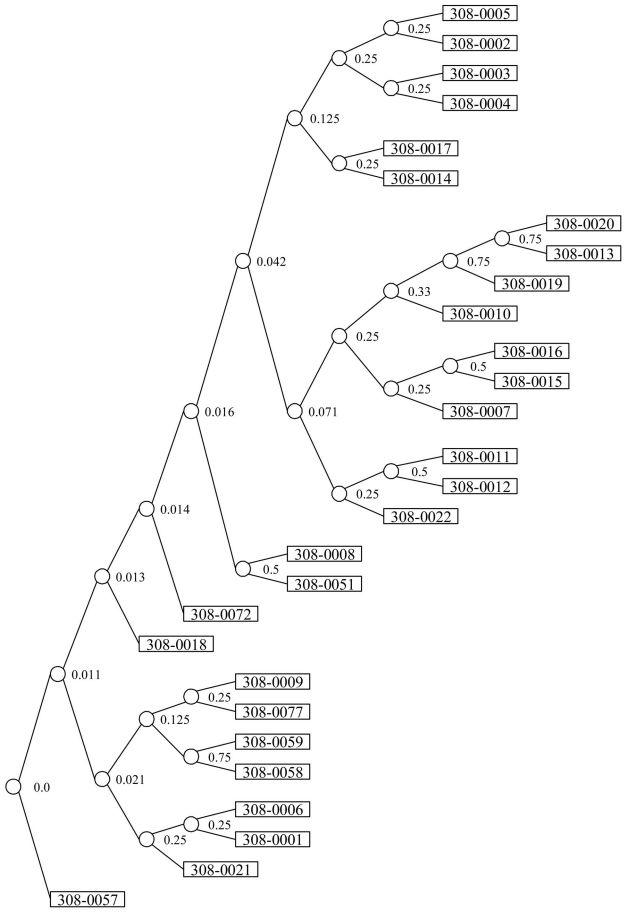

Fig. 2. Neighbor hierarchy of members of attribute STOCK_NO in relation ALUM_PLATE.

In this example, due to the small number of samples (tuples) involved in the construction of the neighbor hierarchy, a popularity threshold of .125 or 1/28 is utilized. This means that at least one occurrence of any pair of attribute values must exist for an inferential relationship to be valid. To determine the normalized clustering correlation value between stock 308-0013 and 308-0020, the confidences and popularities of the inferential relationships

$$(''308-0013'' \rightarrow \text{WIDTH} = 3),$$
$$(''308-0013'' \rightarrow \text{LENGTH} = 144),$$
$$(''308-0013'' \rightarrow \text{AREA} = 432),$$
$$(''308-0013'' \rightarrow \text{THICKNESS} = 0.5),$$
$$(''308-0013'' \rightarrow \text{THICKNESS} = 0.375),$$
$$(''308-0020'' \rightarrow \text{WIDTH} = 3),$$
$$(''308-0020'' \rightarrow \text{LENGTH} = 144),$$
$$(''308-0020'' \rightarrow \text{AREA} = 432),$$
$$(''308-0020'' \rightarrow \text{THICKNESS} = 0.5),$$

and $(''308-0020'' \rightarrow \text{THICKNESS} = 0.375)$ are first calculated.

Then, the normalized clustering correlation is calculated using (4) and (5).

TABLE 2
Relation ALUM_PLATE

STOCK_NO	WIDTH	LENGTH	AREA	THICK-NESS
308-0001	3.5	5.25	18.38	0.375
308-0002	21.24	22.25	472.59	0.25
308-0003	3.75	6.75	25.31	0.25
308-0004	6.75	7	47.25	0.25
308-0005	6.375	11	70.13	0.25
308-0006	2.65	21	55.65	0.375
308-0007	1.25	144	180.00	1
308-0008	0.875	72	63.00	1
308-0009	2	54	108.00	1.25
308-0010	0.3125	144	45.00	0.5
308-0011	12.25	12.25	150.06	0.5
308-0012	8	12.25	98.00	0.5
308-0013	**3**	**144**	**432.00**	**0.5**
308-0014	3.25	5.75	18.69	0.25
308-0015	3.5	144	504.00	0.25
308-0016	0.75	144	108.00	0.25
308-0017	3.25	5.25	17.06	0.125
308-0018	2.24	7.75	17.36	0.75
308-0019	3	144	432.00	0.75
308-0020	**3**	**144**	**432.00**	**0.375**
308-0021	5	6.25	31.25	0.375
308-0022	12.75	36	459.00	0.5
308-0051	4.25	9	38.25	1
308-0057	19	22	418.00	2.25
308-0058	17	20	340.00	8
308-0059	17	20	340.00	1.25
308-0072	8.375	18	150.75	0.125
308-0077	15	21	315.00	1.25

$$\bar{\gamma}(''308-0013'',''308-0020'')$$
$$= \frac{\left[\begin{array}{l}\xi(''308-0013''\rightarrow \text{WIDTH}=3)\times\xi(''308-0020''\rightarrow \text{WIDTH}=3)\\ +\xi(''308-0013''\rightarrow \text{LENGTH}=144)\times\xi(''308-0020''\rightarrow \text{LENGTH}=144)\\ +\xi(''308-0013''\rightarrow \text{AREA}=432)\times\xi(''308-0020''\rightarrow \text{AREA}=432)\\ +\xi(''308-0013''\rightarrow \text{THICKNESS}=.5)\times\xi(''308-0020''\rightarrow \text{THICKNESS}=.5)\\ +\xi(''308-0013''\rightarrow \text{THICKNESS}=.375)\times\xi(''308-0020''\rightarrow \text{THICKNESS}=.375)\end{array}\right]}{(m-1)}$$
$$= \frac{(1+1+1+0+0)}{(5-1)} = \frac{3}{4} = .75.$$

Using the process just described, the normalized clustering correlation could be determined for every possible pair of premises. Since the clustering correlation between stock 308-0013 and stock 308-0020 is relatively high (when compared to the other pairs of premises), both stocks are clustered together in one of the earlier iterations. The neighbor hierarchy of aluminum stocks shown in Fig. 2 is obtained after several iterations of clustering attempts.

If a search is conducted based on the neighbor hierarchy for a substitute for an aluminum stock with WIDTH = 3, LENGTH = 144, and THICKNESS = 0.375 (or stock 308-0020), the database management system may respond to the request with the list shown in Table 3.

4 Evolved Pattern-Based Knowledge Induction (ePKI)

As mentioned in Section 2, the original PKI technique constructs neighbor hierarchies only at the tuple level. Also, it generates neighbor hierarchies using only attribute values currently stored in a database management system. The resulting neighbor hierarchies cannot be used for nearness measurement or transformation of a selection condition that contains an attribute value that does not exist at the time in which the hierarchy is developed, as illustrated in the simple ALUM_PLATE relation example presented in Section 1.

TABLE 3
Approximate Answers of Q: WIDTH = 3, LENGTH = 144, and THICKNESS = 0.375

STOCK_NO	WIDTH	LENGTH	AREA	THICKNESS	NEARNESS
308-0019	3	144	432	0.75	0.75
308-0013	3	144	432	0.50	0.75
308-0010	0.3125	144	45	0.50	0.33
308-0015	3.5	144	504	0.25	0.25
308-0016	0.75	144	108	0.25	0.25
308-0007	1.25	144	180	1.00	0.25

PKI was developed for clustering relation instances at the tuple level using the induced inferential relationships between the values of the key attribute and the values of the other attributes in the relation. Because of the uniqueness property of the key attribute, each clustered value matches up with one and only one value in another attribute in the relation. However, mapping between values from a nonunique attribute (i.e., an attribute with duplicate values) to another attribute could result in one-to-many relationships. For example, the value of 144 for attribute LENGTH can be mapped to five values of attribute THICKNESS (i.e., 0.25, 0.375, 0.5, 0.75, and 1). Therefore, the clustering correlation function presented in (4) works only if attribute A, as the premise of the inferential relationship, is a unique attribute. To develop a neighbor hierarchy of a nonunique attribute, the rule of shared consequence proposed by Merzbacher and Chu is generalized so that it can be applied to any attribute in the relation. The Evolved Shared Consequence Rule is as follows:

If two inferential relationships share the same "set of consequences" (same set of values from an attribute) and have the same attribute as a premise (but different values), then those values are candidates for clustering.

The clustering correlation between a_1 and a_2 of attribute A based upon their relationships with values in attribute B according to the evolved shared consequence rule is:

$$\gamma'(a_1,a_2) = \sum_{j=1}^{n} \xi(A=a_1 \rightarrow B=b_j) \times \xi(A=a_2 \rightarrow B=b_j), \tag{7}$$

where b_j is a unique value observed for attribute B. For example, the unique values observed for attributes

THICKNESS and LENGTH in the ALUM_PLATE relation example are $\{t_1, t_2, t_3, t_4, t_5\}$ and $\{l_1\}$, respectively.

The clustering correlation function for any two values based on the rule of shared consequence (i.e., multiple consequent attributes), shown in (4), is also modified to:

$$\gamma'(a_1, a_2) = \sum_{i=1}^{m}\sum_{j=1}^{n} \xi(A = a_1 \rightarrow B_i = b_j) \times \xi(A = a_2 \rightarrow B_i = b_j). \tag{8}$$

Based on the information in Table 1, seven unique values are identified in attribute STOCK_NO, one in LENGTH, five in AREA, and six in THICKNESS. The clustering correlation between w_3 and w_4 is determined as:

$$\gamma'(w_3, w_4) = \begin{bmatrix} \sum_{j=1}^{7} \xi(w_3 \rightarrow STOCK\_NO = s_j) \times \xi(w_4 \rightarrow STOCK\_NO = s_j) \\ + \sum_{j=1}^{1} \xi(w_3 \rightarrow LENGTH = l_j) \times \xi(w_4 \rightarrow LENGTH = l_j) \\ + \sum_{j=1}^{5} \xi(w_3 \rightarrow AREA = a_j) \times \xi(w_4 \rightarrow AREA = a_j) \\ + \sum_{j=1}^{6} \xi(w_3 \rightarrow THICKNESS = t_j) \times \xi(w_4 \rightarrow THICKNESS = t_j) \end{bmatrix}.$$

Since STOCK_NO is a unique attribute, the first sum factor always yields 0. Therefore, the equation above is reduced to:

$$\gamma'(w_3, w_4) = \begin{bmatrix} 0 + \xi(w_3 \rightarrow l_1) \times \xi(w_4 \rightarrow l_1) + \xi(w_3 \rightarrow a_3) \\ \times \xi(w_4 \rightarrow a_3) + \xi(w_3 \rightarrow a_4) \times \xi(w_4 \rightarrow a_4) \\ + \xi(w_3 \rightarrow t_4) \times \xi(w_4 \rightarrow t_4) \end{bmatrix} = [0 + (1 \times 1) + (1 \times 0) + (0 \times 1) + (1 \times 1)] = 2.$$

Notice that when B_i is a unique attribute, then:

$$\xi(A = a_1 \rightarrow B_i = b_j) \times \xi(A = a_2 \rightarrow B_i = b_j) = 0$$

because no two tuples ($A = a_1$ and $A = a_2$) can have the same consequence $B_i = b_j$. Such a characteristic prevents the normalized clustering correlation between two values of a nonunique attribute from reaching 1. The property creates a bias in favor of substituting those unique attribute values before those of nonunique attribute values. Therefore, the theoretical maximum value of $\gamma(a_1, a_2)$ is $m - 2$ (instead of $m - 1$) when A is not the key attribute. The normalized clustering correlation presented in (5) is modified to:

$$\bar{\gamma}'(a_1, a_2) = \frac{1}{m - n - p}\gamma'(a_1, a_2), \tag{9}$$

where m is the total number of attributes in the relation, n is the number of unique attributes in the relation, and p is 0 if A is also a unique attribute; otherwise, p equals 1 and $m - n - p \neq 0$. The Appendix includes an example of the calculation of the normalized clustering correlation.

Neighbor hierarchies can be created for any attributes in the relation by combining the modified clustering correlation functions with the Binary Clustering algorithm. A neighbor hierarchy of attribute WIDTH in the relation ALUM_PLATE developed with the ePKI is shown in Fig. 3. For example, the nearness value between WIDTH = 3 and WIDTH = 1.25 is 0.095 based on the neighbor hierarchy in Fig. 3.

Using the same approach, an additional neighbor hierarchy for attribute THICKNESS can be constructed (not shown here). The nearness value between THICKNESS = 1 and THICKNESS = 0.375 is 0.062. Comparing the two nearness values indicates that relaxing the WIDTH condition is better than relaxing the THICKNESS condition (since 0.095 is greater than 0.062). Therefore, when the user searches for a part with WIDTH = 3, LENGTH = 144, and THICKNESS = 1, the cooperative system would suggest the aluminum plate with stock number "308-0019."

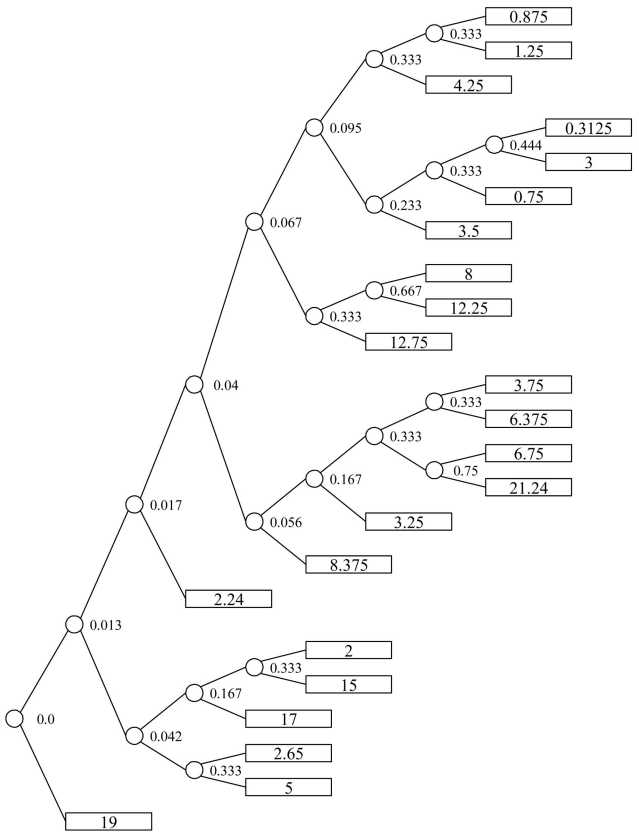

Fig. 3. A neighbor hierarchy of WIDTH in ALUM_PLATE through the ePKI technique.

5 Experimental Results

Evaluating the soundness of a neighbor hierarchy could be done using domain experts' opinions. However, users may prefer different neighbor hierarchies for different applications even though the information is derived from the same data source. The authors' experience has been that expert decisions can be quite subjective. An excellent neighbor hierarchy for one expert could receive a poor ruling from another. Thus, in order to fairly judge the performance of the proposed attribute-level neighbor hierarchy construction, data sets with explicit inferential relationships were utilized.

An experiment was conducted to test whether the ePKI was capable of correctly capturing inferential relationships among attribute values using component-machine incidence matrices. Component-machine incidence matrices are used for grouping parts into families and/or machines into manufacturing centers as part of group technology [27]. In addition, an algorithm for transforming a component-machine incidence matrix into its block diagonal form using ePKI's clustering correlations was developed.

TABLE 4
Component-Machine Incidence Matrix

component \ Machine	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1									1							1	1		1														1							
2										1			1	1								1											1			1				
3		1									1	1			1								1	1										1							
4								1											1		1							1										1	1	
5					1					1				1								1													1	1			
6					1													1							1	1			1							1		
7																									1						1								1
8				1	1																				1	1			1									
9						1	1																						1				1						1
10						1	1													1					1														
11										1			1									1														1	1			
12				1	1													1		1				1			1			1									
13	1								1							1	1																1							
14			1											1		1									1															
15	1			1	1													1								1	1									1			
16								1											1		1											1						1	1	1	
17							1													1							1		1										1
18				1	1													1								1	1			1									
19									1				1	1								1														1	1			1
20		1									1	1			1				1				1	1						1				1					
21	1								1	1						1																	1							
22	1															1	1														1		1							
23			1																						1							1							
24		1	1																													1							

TABLE 5
A Block Diagonal Form of a Component-Machine Incidence Matrix

component \ Machine	18	26	27	5	30	4	20	29	6	7	40	9	16	17	33	1	3	25	32	28	8	21	37	38	39	19	11	12	15	2	23	24	31	34	13	14	10	22	35	36
15	1	1	1	1		1										1							1																	
6	1	1	1	1	1																			1																
8		1	1	1	1	1																																		
12	1		1	1	1	1	1																									1								
18	1	1	1	1	1	1																																		
17							1	1		1	1																						1							
9								1	1	1	1				1																									
10		1					1		1	1																														
21												1	1		1	1																					1			
1												1	1	1	1											1														
13												1	1	1	1	1																								
22					1								1	1	1	1																								
24																	1		1										1											
14													1				1	1																		1				
23																	1	1	1																					
7											1							1	1																					
4																				1	1	1		1	1	1														
16																			1		1	1	1	1	1	1														
3																											1	1	1	1	1	1		1						
20																										1	1	1	1	1	1	1	1	1						
2															1																				1	1	1	1		1
5				1																																1	1	1	1	1
11																																			1		1	1	1	1
19											1																								1	1	1	1	1	1

An example of a component-machine incidence matrix is shown in Table 4. Rows and columns in the table represent components (i.e., parts) and machines, respectively. The (i, j)th element of the matrix (the value of attribute jth of tuple ith) has a value of 1 if part ith visits machine jth; otherwise, a value of 0 is assigned (shown as a blank cell in Table 4). For example, component 7 stops at machine 25, 32, and 40. It is important to note that the component-machine incidence matrix does not specify the visiting order. The matrix in Table 4 is transformed into a block diagonal form to form part families and manufacturing centers by rearranging components (tuples) and machines (attributes), as shown in Table 5. In this example, parts 7, 14, 23, and 24 are processed by a common set of machines (i.e., 3, 25, and 32). Each submatrix in the block diagonal form suggests a possible part family and a potential manufacturing center.

For a given set of data, a perfect block diagonal form in which all 1s are in the diagonal submatrices and all 0s occupy the off-submatrices may not exist. Chandrasekharan and Rajagopalan [28] developed a methodology called ZODIAC for transforming component-machine incidence matrices into their block diagram forms. They also suggested using *Grouping Efficiency* [29] as the measure of the goodness of a block diagonal matrix—how close a matrix is to its perfect block diagonal form. The reader is encouraged to review [22] for a detailed explanation of the grouping efficiency calculation. A perfect block diagonal form has a 100 percent grouping efficiency. The matrix in Table 5 has a grouping efficiency of 85.40 percent.

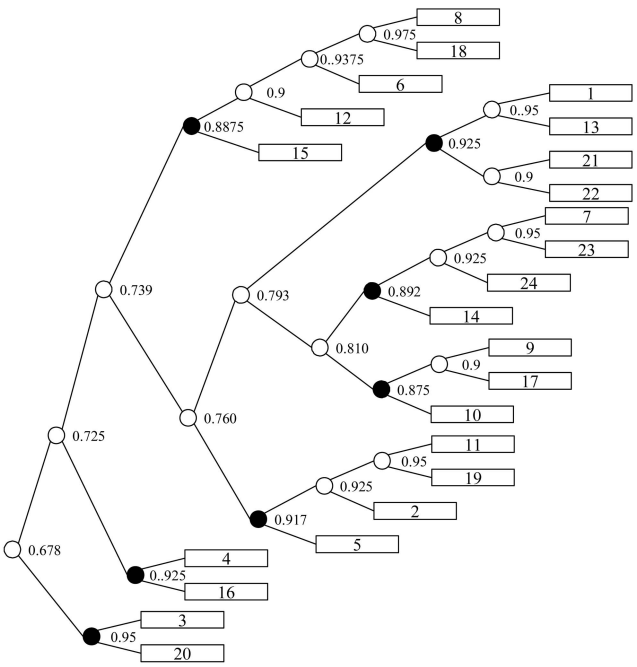

Fig. 4. Part families constructed by the ePKI-based block diagonal transformation algorithm.

Component-machine incidence matrices were used in this research for assessing the performance of the proposed attribute-level neighbor hierarchy construction algorithms for the following reasons:

1. Since data in a component-machine incidence matrix exhibit true inferential relationships that are visible after a block diagonal transformation, there is no need to involve experts. The algorithms can be evaluated to determine whether Type I or Type II error exists in the resulting neighbor hierarchies.
2. By using the Grouping Efficiency measure, the strength of the inferential relationships among attributes and attribute values can be quantified.

5.1 ePKI-Based Block Diagonal Transformation Algorithm

As demonstrated earlier, ePKI can be used with the Binary Algorithm to construct neighbor hierarchies of attribute values based on their relationship represented by other attributes. Based on the fact that the resulting neighbor hierarchies can also be viewed as a suggestion on how to order attribute values (based on their similarities), the following algorithm to transform a part-machine incidence matrix into its block diagonal form was developed:

1. Select a column by which the rows of the part-machine incidence matrix will be ordered.
2. Choose a target number of part families. The default number of families is 1.
3. Construct a neighbor hierarchy based on the values in the selected column using ePKI's clustering correlations and the binary clustering algorithm until the desired number of part families is achieved. If the default number of families is selected, continue parts clustering until all parts are grouped. The later case results in one large part family.
4. Rearrange the parts (rows) based on their order shown in the resulting neighbor hierarchy.
5. If necessary, a part family can be broken into two smaller families by removing the node with the smallest clustering correlation value of the part family.

A neighbor hierarchy of parts resulting from applying the ePKI-based block diagonal transformation algorithm on the part-machine incidence matrix shown in Table 4 is depicted in Fig. 4. The part number column was chosen for

TABLE 6
Resulting Table from the Binary Data Modifications

	Machine																																							
		18	26	27	5	30	4	20	29	6	7	40	9	16	17	33	1	3	25	32	28	8	21	37	38	39	19	11	12	15	2	23	24	31	34	13	14	10	22	35	36
Component	15	1	1	1	1		1									10	1							1																		
	6	1	1	1	1	1										2									1																
	8		1	1	1	1	1									4																									
	12	1		1	1	1	1	1								10																1									
	18	1	1	1	1	1	1									8																									
	17							1	1		1	1				16																		1								
	9								1	1	1	1				13																									
	10		1					1	1	1						14																									
	21												1	1		21	1																						1			
	1												1	1	1	19										1															
	13												1	1	1	23	1																									
	22					1								1	1	19	1																									
	24															28		1		1											1										
	14													1		32		1	1																			1			
	23															32		1	1	1																					
	7											1				30			1	1																					
	4															18					1	1	1		1	1	1														
	16															34				1		1	1	1	1	1	1														
	3															36												1	1	1	1	1	1	1		1					
	20															38											1	1	1	1	1	1	1	1	1						
	2															37																					1	1	1	1		1
	5					1										42																						1	1	1	1	1
	11															44																					1		1	1	1	1
	19											1				46																					1	1	1	1	1	1

TABLE 7
Component-Machine Incidence Matrix with a Group Efficiency of 100 Percent

component \ Machine	18	26	27	5	30	4	20	29	6	7	40	9	16	17	33	1	3	25	32	28	8	21	37	38	39	19	11	12	15	2	23	24	31	34	13	14	10	22	35	36
15	1	1	1	1		1										1							1																	
6	1	1	1	1	1																			1																
8		1	1	1	1	1																																		
12	1		1	1	1	1	1																									1								
18	1	1	1	1	1	1																																		
17							1	1		1	1																						1								
9								1	1	1	1				1																										
10		1					1		1	1																														
21												1	1		1	1																						1			
1												1	1	1	1											1														
13												1	1	1	1	1																								
22					1								1	1	1	1																								
24																	1		1												1									
14													1				1	1																			1			
23																	1	1	1																					
7											1							1	1																					
4																				1	1	1		1	1	1														
16																			1		1	1	1	1	1	1														
3																											1	1	1	1	1	1		1						
20																										1	1	1	1	1	1	1	1	1						
2															1																				1	1	1	1		1
5				1																															1	1	1	1	1
11																																			1		1	1	1	1
19											1																								1	1	1	1	1	1

the clustering. The default number of part families of 1 was used in this case. Seven part families (represented by the solid black dots in Fig. 4) were obtained after removing six nodes with the least clustering correlation values.

5.2 Synthetic Data Sets

Traditional component-machine incidence matrices only contain 0s and 1s. If part (or machine) clustering is performed on a machine column, the result will always be a hierarchy of two attribute values (0 and 1). Some modifications must be performed on the binary data so that it represents the data found in typical relations. For our performance evaluation purposes, an attribute is randomly selected from the relation. Then, all 0s and 1s of the attribute of interest (whose values are to be clustered) are replaced with integers ranging from 1 to $2n$, where n is the number of tuples.

First, the entire range from 1 to $2n$ is divided into unequal subsets based on the number of families and the number of components in each family. The division of the range into subsets ensures that no two interfamily instances will be forced to combine together. Such a conglomerate is acceptable only if it occurs within the same family. Then, these subsets of integers are used to replace 0s and 1s in each component family: 0s are substituted with randomly selected even numbers, whereas 1s are substituted with randomly selected odd numbers. For example, Family 1 consists of five components (i.e., 15, 6, 8, 12, and 18); thus, integers 1 to 10 are allocated to the family. For a given family, an integer may be used more than once. Table 6 shows an example of a component-machine incidence matrix after the binary data modifications are performed for column "33."

5.3 Validation of the Resulting Neighbor Hierarchies

The ePKI with the Binary Clustering algorithm was utilized to construct neighbor hierarchies for component-machine incidence matrices. The results were compared with the best matrices' block diagonal forms produced by Chandrasekharan and Rajagopalan's approach. It was ruled that the proposed neighbor hierarchy construction algorithm committed a mistake for each incident of Type I error. Type I error is defined here as grouping two values together when they were not supposed to be grouped. In our case, there was no Type II error (i.e., two values were not grouped together when they were supposed to be grouped) because all values were eventually clustered with another value or group of values by the ePKI technique. Also, considering Type II error would result in a double penalty for each fault.

All seven matrices mentioned in [29] were utilized in the validation process. Based on Chandrasekharan and Rajagopalan's work, the highest achievable grouping efficiencies of their matrices are 100 percent, 95.20 percent, 91.14 percent, 85.40 percent, 77.31 percent, 72.43 percent, and 69.33 percent. Table 7 represents a block diagonal matrix with a grouping efficiency of 91.14 percent.

5.4 Effect of Inferential Relationship Strength Analysis

The effect of the strength of inferential relationships on the resulting neighbor hierarchy was studied using the seven component-machine incidence matrices. Ten attributes (machines) were randomly selected and neighbor hierarchies were constructed using the proposed algorithm described in Section 5.1. The number of occurrences of Type I error was determined for each trial. The ratios of the average number of fault groupings to times of clustering (number of nodes) for all seven data sets are shown in Fig. 5.

When group efficiencies are at least 85.41 percent, results show that the neighbor hierarchies constructed via the ePKI agree with those suggested by the matrices' block diagonal forms generated using the ZODIAC technique [29]. The percentage of Type I error is approximately at 30 percent when the group efficiency is 69.33 percent. It is important to note that despite the fact that Type I errors were observed for the last three matrices, the recalculated Grouping

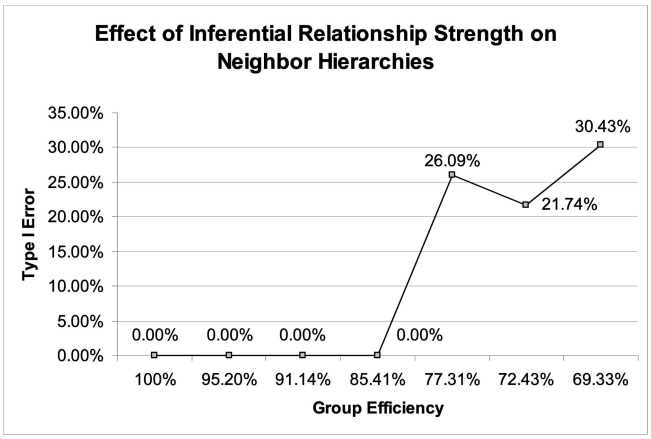

Fig. 5. Effect of inferential relationship strength on neighbor hierarchies.

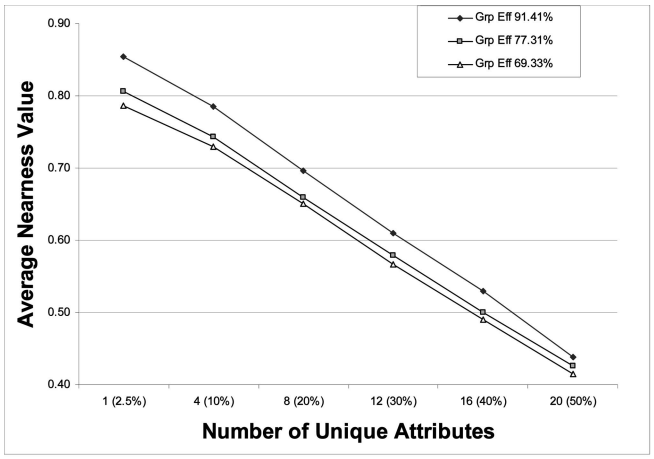

Fig. 6. Effect of the number of unique attributes on nearness values.

Efficiencies of those matrices' block diagonal forms produced by the ePKI method were better than those suggested by Chandrasekharan and Rajagopalan. For example, Chandrasekharan and Rajagopalan's best block diagonal form of the last component-machine incidence matrix has a Grouping Efficiency value of .6933, while the block diagonal form for the same matrix that resulted from the ePKI-based block diagonal transformation algorithm receives a score of .732.

When a component-machine incidence matrix is used to develop families of components, application domain experts may choose to accept a suboptimum solution for some practical constraints. Therefore, any deviation from Chandrasekharan and Rajagopalan's results is still considered as an error for the consistency of the analysis. Unlike those results of typical block diagonal form transformation techniques, neighbor hierarchies generated by the ePKI-based block diagonal transformation algorithm not only group components into families, they also provide additional information in terms of which families are closer to which.

In summary, the ePKI technique is capable of generating reliable attribute-level neighbor hierarchies if there are strong inferential relationships among data in the table.

5.5 Effect of Unique Attribute Analysis

Both the original and the ePKI technique construct neighbor hierarchies based on the rule of shared consequence. This means these techniques can interpolate any inferential relationships and group two attribute values together only if they share at least one same consequence attribute value. In the case that all attributes are unique, all nearness values will be 0. Attributes with little duplication of attribute values, which are more difficult to identify than a unique attribute, will cause the same problem with lesser degrees. In such cases, the nearness values of the resulting neighbor hierarchy will obviously be biased, which could cause an inaccurate approximate answer. Therefore, an experiment was conducted to study the effect of unique attributes on the final nearness values.

Additional unique attributes were introduced into the calculation by replacing values (0s and 1s) in certain attributes with sets of unique values. Then, the number of unique attributes in the table was gradually increased and the average nearness values in the neighbor hierarchies were collected, which are shown in Fig. 6. Based on the experimental results, average nearness values drop when the number of unrecognized unique attributes increases. The problem is more severe for data sets with strong inferential relationships. Also, average nearness values almost converge as the number of unique attributes grows.

Through the modified normalized clustering correlation (see (9)), the average nearness values stay almost the same for all numbers of unique attributes introduced.

6 Concluding Remarks

In this paper, an effective technique for developing neighbor hierarchies at the attribute level has been introduced. The ePKI technique allows for the construction of neighbor hierarchies for nonunique attributes based upon confidences, popularities, and correlations of relationships among attribute values. In addition, the ePKI technique is capable of clustering both numerical and categorical attribute values.

Attribute value neighbor hierarchies generated by the ePKI technique allow cooperative query answering systems to search for approximate answers by relaxing each individual query condition separately. As a result, users can search for approximate answers even when the exact match answers do not exist in the database. This capability is essential for developing cooperative query answering systems in many domains such as rapid prototyping applications in the manufacturing information domain.

Several experiments were conducted to assess the performance of the ePKI in constructing attribute-level neighbor hierarchies. The results indicate the following:

- The ePKI technique produces accurate neighbor hierarchies when strong inferential relationships

appear among data. Degradations in the result qualities become evident when the technique is used on less related sets of data.

- The higher the number of irrelevant attributes (such as unique attributes or alternate key attributes) utilized in the calculation, the lower the resulting nearness values. The effect of unique attribute on the overall nearness values is relatively stronger when the technique is used with tightly related data. In other words, the domains of the consequence attributes can affect nearness value suggested by the algorithms.

APPENDIX

NORMALIZED CLUSTERING CORRELATION CALCULATION EXAMPLE

From Table 2, 28 unique values are identified in attribute STOCK_NO, 18 in LENGTH, 24 in AREA, and eight in THICKNESS. The normalized clustering correlation between WIDTH=3 and WIDTH=0.3125 (represented as w_1 and w_2, respectively, in the formulas below) is calculated as follows:

$$\gamma'(w_1,w_2)=\begin{bmatrix}\sum_{j=1}^{28}\xi(w_1\rightarrow STOCK\_NO=s_j)\times\xi(w_2\rightarrow STOCK\_NO=s_j)\\+\sum_{j=1}^{18}\xi(w_1\rightarrow LENGTH=l_j)\times\xi(w_2\rightarrow LENGTH=l_j)\\+\sum_{j=1}^{24}\xi(w_1\rightarrow AREA=a_j)\times\xi(w_2\rightarrow AREA=a_j)\\+\sum_{j=1}^{8}\xi(w_1\rightarrow THICKNESS=t_j)\\\times\xi(w_2\rightarrow THICKNESS=t_j)\end{bmatrix}.$$

Since STOCK_NO is a unique attribute, its contribution to the equation is zero. Due to the same reason, only attribute values found in LENGTH, AREA, and THICKNESS that associate with either or both w_1 and w_2 are shown in the calculation below:

$$\gamma'(w_1,w_2)=\begin{bmatrix}0+\xi(w_1\rightarrow LENGTH=144)\times\xi(w_2\rightarrow LENGTH=144)\\+\xi(w_1\rightarrow AREA=432)\times\xi(w_2\rightarrow AREA=432)+\xi(w_1\rightarrow AREA=45)\\\quad\times\xi(w_2\rightarrow AREA=45)\\+\xi(w_1\rightarrow THICKNESS=.375)\times\xi(w_2\rightarrow THICKNESS=.375)\\+\xi(w_1\rightarrow THICKNESS=.5)\times\xi(w_2\rightarrow THICKNESS=.5)\\+\xi(w_1\rightarrow THICKNESS=.75)\times\xi(w_2\rightarrow THICKNESS=.75)\end{bmatrix}$$

$$=\begin{bmatrix}0+(1\times1)+(1\times0)+(0\times1)+(1/3\times0)\\+(1/3\times1)+(1/3\times0)\end{bmatrix}=1.333$$

$$\bar{\gamma}'(w_1,w_2)=1/(5-1-1)\times1.333=.444.$$

Obviously, a clustering of WIDTH=3 and WIDTH=0.3125 will not happen in the early iterations because of their intermediate clustering correlation values. In this example, (WIDTH=3, WIDTH=0.3125) was chosen to form a new cluster in the third iteration after (WIDTH=6.75, WIDTH=21.24) and (WIDTH=8, WIDTH=12.25) were grouped together in the first and second iterations with clustering correlations of .75 and .6667, respectively. In the fourth iteration, one of the clustering candidates is combining {(WIDTH=3, WIDTH=0.3125) with WIDTH=.75}. WIDTH=.75 is represented as w_3. Their clustering correlation computation is shown below:

$$\gamma'(\{w_1,w_2\},w_3)$$

$$=\begin{bmatrix}\sum_{j=1}^{28}\xi((w_1\vee w_2)\rightarrow STOCK\_NO=s_j)\times\xi(w_3\rightarrow STOCK\_NO=s_j)\\+\sum_{j=1}^{18}\xi((w_1\vee w_2)\rightarrow LENGTH=l_j)\times\xi(w_3\rightarrow LENGTH=l_j)\\+\sum_{j=1}^{24}\xi((w_1\vee w_2)\rightarrow AREA=a_j)\times\xi(w_2\rightarrow AREA=a_j)\\+\sum_{j=1}^{9}\xi((w_1\vee w_2)\rightarrow THICKNESS=t_j)\times\xi(w_2\rightarrow THICKNESS=t_j)\end{bmatrix}$$

$$=\begin{bmatrix}0\\+\xi((w_1\vee w_2)\rightarrow LENGTH=144)\times\xi(w_3\rightarrow LENGTH=144)\\+\xi((w_1\vee w_2)\rightarrow AREA=45)\times\xi(w_2\rightarrow AREA=45)\\+\xi((w_1\vee w_2)\rightarrow AREA=108)\times\xi(w_2\rightarrow AREA=108)\\+\xi((w_1\vee w_2)\rightarrow AREA=432)\times\xi(w_2\rightarrow AREA=432)\\+\xi((w_1\vee w_2)\rightarrow THICKNESS=.25)\times\xi(w_2\rightarrow THICKNESS=.25)\\+\xi((w_1\vee w_2)\rightarrow THICKNESS=.375)\times\xi(w_2\rightarrow THICKNESS=.375)\\+\xi((w_1\vee w_2)\rightarrow THICKNESS=.5)\times\xi(w_2\rightarrow THICKNESS=.5)\\+\xi((w_1\vee w_2)\rightarrow THICKNESS=.75)\times\xi(w_2\rightarrow THICKNESS=.75)\end{bmatrix}$$

$$=\begin{bmatrix}0+((3+1)/4\times1)+((0+0)/4\times1)+((3+1)/4\times0)r+((0+0)/4\times1)\\+((1+0)/4\times0)+((1+1)/4\times0)+((1+0)/4\times0)\end{bmatrix}=1$$

$$\bar{\gamma}'(\{w_1,w_2\},w_3)=1/3=.333.$$

REFERENCES

[1] J. Minker, G.A. Wilson, and B.H. Zimmerman, "Query Expansion by the Addition of Clustered Terms for a Document Retrieval System," *Information Storage and Retrieval*, vol. 8, pp. 329-348, 1972.

[2] F. Cuppens and R. Demoloube, "Cooperative Answering: A Methodology to Provide Intelligent Access to Database," *Proc. Second Int'l Conf. Expert Database Systems*, 1988.

[3] H. Grice, "Logic and Conversation," *Syntax and Semantics*, P. Cole and J. Morgan, eds., Academic Press, 1975.

[4] E. Petrakis, G.M. Euripides, and C. Faloutsos, "Similarity Searching in Medical Image Databases," *IEEE Trans. Knowledge and Data Eng.*, vol. 9, no. 3, pp. 435-447, May/June 1997.

[5] D. Che, K. Aberer, and Y. Chen, "The Design of Query Interfaces to the GPCRDB Biological Database," *Proc. User Interfaces to Data Intensive Systems*, pp. 22-31, 1999.

[6] D. Che, C. Yangjun, and A. Karl, "Query System in a Biological Database," *Proc. 11th Int'l Conf. Scientific and Statistical Database Management*, pp. 158-167, 1999.

[7] C.T. Yu and W. Sun, "Automatic Knowledge Acquisition and Maintenance for Semantic Query Optimization," *IEEE Trans. Knowledge and Data Eng.*, vol. 1, pp. 362-375, 1989.

[8] J. Han, Y. Cai, and N. Cercone, "Data-Driven Discovery of Quantitative Rules in Relational Databases," *IEEE Trans. Knowledge and Data Eng.*, vol. 5, pp. 29-40, 1993.

[9] J. Han and Y. Fu, "Dynamic Generation and Refinement of Concept Hierarchies for Knowledge Discovery in Database," *Proc. AAAI Workshop Knowledge Discovery in Databases (KDD 94)*, pp. 157-168, July 1994.

[10] T. Puthpongsiriporn, "Neighbor Hierarchy Construction at the Attribute Level for Cooperative Query Answering," *Proc. 2003 Industrial Eng. Research Conf.*, May 2003.

[11] M. Merzbacher, "Nearness and Cooperative Query Answering," unpublished PhD dissertation, Computer Science Dept., Univ. of California, Los Angeles, 1993.

[12] Y. Huang, "Intelligent Query Answering by Knowledge Discovery Techniques," thesis, Simon Fraser Univ., Canada, 1993.

[13] J. Ozawa and K. Yamada, "Discovery of Global Knowledge in a Database for Cooperative Answering," *Proc. Joint Conf. Fourth IEEE Int'l Conf. Fuzzy Systems and Second Int'l Fuzzy Eng. Symp.*, 1995.

[14] C.E. Shannon and W. Weave, *The Mathematical Theory of Communication.* Urbana, Ill.: Univ. of Illinois Press, 1964.

[15] D.K.Y. Chiu, A.K.C. Wong, and B. Cheung, "Information Discovery through Hierarchical Maximum Entropy Discretization and Synthesis," *Knowledge Discovery in Databases*, AAAI Press/The MIT Press, 1991.

[16] A.K.C. Wong and D.K.Y. Chiu, "Synthesizing Statistical Knowledge from Incomplete Mixed-Mode Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 9, no. 6, pp. 796-803, 1987.
[17] J. Genari, P. Langley, and D. Fisher, "Models of Incremental Concept Formation," *Artificial Intelligence,* vol. 40, pp. 11-62, 1987.
[18] W. Chu and K. Chiang, "Abstraction of High Level Concepts from Numerical Values in Databases," *Proc. AAAI Workshop Knowledge Discovery in Database,* pp. 133-144, 1994.
[19] W.W. Chu, K. Chiang, C.C. Hsu, and H. Yau, "An Error-Based Conceptual Clustering Method for Providing Approximate Query Answers," *Comm. ACM,* vol. 39, no. 12, pp. 216-230, Dec. 1996.
[20] M. Merzbacher and W. Chu, "Pattern-Based Clustering for Database Attribute Values," *Proc. AAAI Workshop Knowledge Discovery in Database,* 1993.
[21] S.M. Ali and S.D. Silvey, "A General Class of Coefficient of Divergence of One Distribution from Another," *J. Royal Statistical Soc., Series B,* vol. 2, pp. 131-142, 1966.
[22] I. Csiszár, "Information-Type Measures of Difference of Probability Distributions and Indirect Observatons," *Studia Scient. Math. Hung.,* vol. 2, pp. 299-318, 1967.
[23] J. Lin, "Divergence Measures Based on the Shannon Entropy," *IEEE Trans. Information Theory,* vol. 37, no. 1, pp. 145-151, 1991.
[24] S. Kullback and R.A. Leibler, "On Information and Sufficiency," *Annals of Math. Statistics,* vol. 22, pp. 76-86, 1951.
[25] L. Withers, "Some Inequalities Relating Different Measures of Divergence between Two Probability Distributions," *IEEE Trans. Information Theory,* vol. 45, no. 5, pp. 1728-1735, 1999.
[26] D. Malerba, F. Esposito, and M. Monopoli, "Comparing Dissimilarity Measures for Probabilistic Symbolic Objects," *Data Mining III, Series Management Information Systems,* vol 6, pp. 31-40, 2002.
[27] N. Singh, *Systems Approach to Computer-Integrated Design and Manufacturing.* John Wiley and Sons, 1996.
[28] M.P. Chandrasekharan and R. Rajagopalan, "ZODIAC—An Algorithm for Concurrent Formation of Part-Families and Machine-Cells," *Int'l J. Production Research,* vol. 25, no. 6, pp. 835-850, 1987.
[29] M.P. Chandrasekharan and R. Rajagopalan, "Groupability: An Analysis of the Properties of Binary Data Matrices for Group Technology," *Int'l J. Production Research,* vol. 27, no. 6, pp. 1035-1052, 1989.

Thanit Puthpongsiriporn received the doctoral degree from the University of Pittsburgh in 2002. He is an assistant professor at Oregon State University, where he teaches courses in the Information Systems Engineering program. He has performed research in the areas of wireless computing, distributed information systems, Web services and applications, automatic identification, and data capture. He has led several information system designs and development projects in the past 10 years. Selected projects include the development of Disease Management and Early and Periodic Screening, Diagnostic, and Treatment (EPSDT) systems for The University of Pittsburgh Medical Center Health Plan and the evaluation of Radio Frequency technologies for the Defense Research Projects Agency (DARPA), US Department of Defense.

J. David Porter received the MS and PhD degrees in industrial engineering from the University of Pittsburgh in 1999 and 2000, respectively. He is an assistant professor in the Department of Industrial and Manufacturing Engineering at Oregon State University. His research interests include automatic identification and data capture, intelligent transportation systems, supply chain engineering, wireless communications, and manufacturing systems engineering. He is a member of SME and ASEE and is the current Director-Elect of the Computers and Information Systems Division of the Institute of Industrial Engineers (IIE).

Bopaya Bidanda is the Ernest Roth Professor and Chairman of the Department of Industrial Engineering at the University of Pittsburgh. His research interests include manufacturing systems, reverse engineering, rapid prototyping, and product development. He has copublished two books with McGraw Hill (*Automated Factory Handbook* and *Shared Manufacturing*), in addition to more than 100 papers in international journals and conference proceedings. He is also the president-elect of the Council of Industrial Engineering Academic Department Heads (CIEADH). He has had industrial experience in aerospace manufacturing, precision manufacturing, and tooling accrued before graduate study. He regularly consults with corporate organizations and government agencies.

Ming-En Wang received the BS degree from the National Chiao Tung University in Taiwan, the MS degree from Iowa State University, and the PhD degree in industrial and operations engineering from the University of Michigan, Ann Arbor. He is currently an assistant professor in the Department of Industrial Engineering at the University of Pittsburgh. While pursuing his PhD, he worked at Ford Motor Company building an integrated information and engineering system for its car-body stamping division. His research interests are in computational problems that arise in design, manufacturing, and information technology. Some of his current projects include ad hoc localization of sensor networks, supply chain diagnosis in an information rich environment, preemption traffic network design, and hybrid design representations.

Richard E. Billo is the associate dean of engineering research at the University of Texas at Arlington. He conducts research in database systems, wireless networks, and bioinformatics. He is currently the founding director of the Texas Radio Frequency Identification Systems Research Center. He has published more than 60 refereed papers, and been awarded more than $20 million in funded research grants and gifts. He has conducted research for the US Navy, the US Defense Logistics Agency, the US National Science Foundation, and such companies as Intel, Symbol Technologies, and Hewlett Packard. He currently serves on the editorial board for the *International Journal of Manufacturing Systems*. Previously, Dr. Billo served as chair of the Industrial and Manufacturing Engineering Department at Oregon State University, held joint appointments as an associate professor in the Industrial Engineering Department and the Information Science Department at the University of Pittsburgh, and worked as a technical group leader for Pacific Northwest National Laboratories. He has won awards as an Intel Faculty Fellow, Outstanding Engineering Faculty, and the Whiteford Professorship.