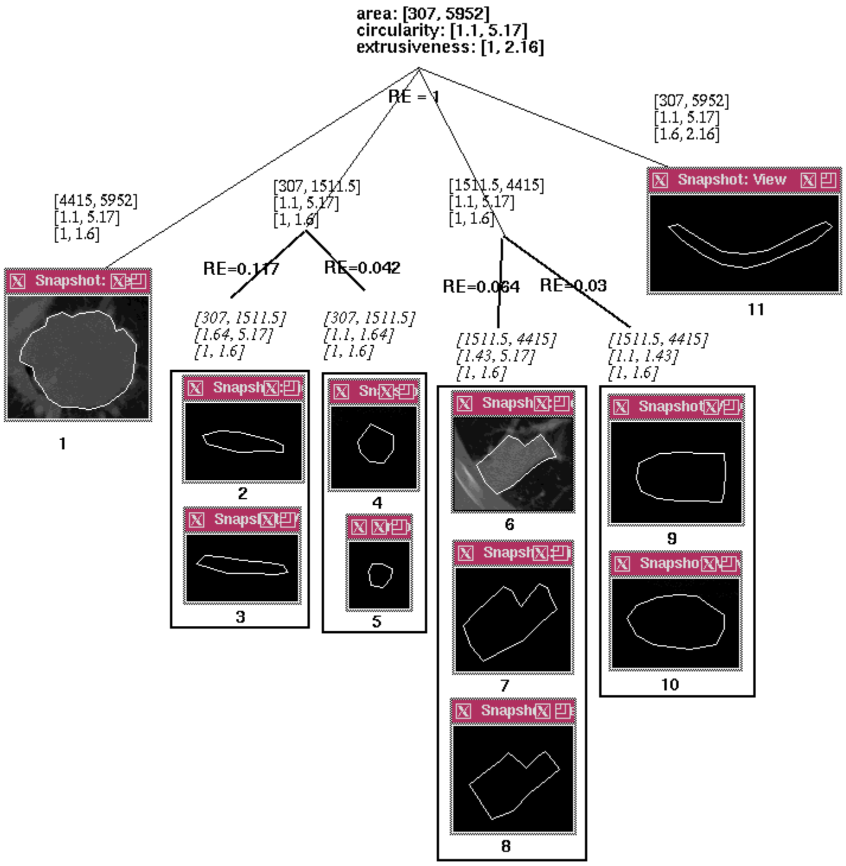

Figure 6: An MTAH for 11 lung tumor contours generated by M-DISC based on *area*, *circularity*, and *extrusiveness*.

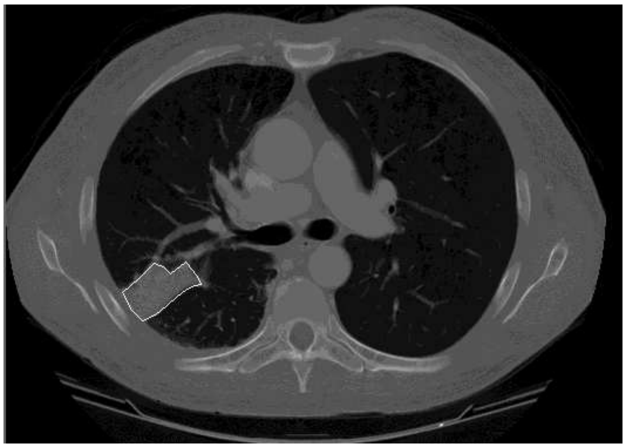

Figure 7: The CT scanned lung image for image 6 (in Figure 3) with the lung tumor contour outlined.

[18] L. Shapiro. A structural model of shape. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, March 1980.

[19] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy: The Principles and Practice of Numerical Classification.* W. H. Freeman and Company, San Francisco, 1973.

[20] M Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision.* Chapman & Hall Computing, 1993.

[21] Andrew K. C. Wong and David K. Y. Chiu. Synthesizing statistical knowledge from incomplete mixed-mode data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):796–805, 1987.

[9] F. Cuppens and R. Demoloube. Cooperative answering: a methodology to provide intelligent access to databases. In *Proceedings of the 2th International Conference on Expert Database Systems*, Virginia, USA, 1988.

[10] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[11] T. Gassterland, P. Godfrey, and Jack Minker. An overview of cooperative answering. *Journal of Intelligent Information Systems*, 1:123–157, 1992.

[12] J. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11–62, 1989.

[13] Stephen Jose Hanson and Malcolm Bauer. Conceptual clustering, categorization, and polymorphy. *Machine Learning*, 3:343–372, 1989.

[14] M. Lebowitz. Experiments with incremental conceptual formation. *Machine Learning*, 2(2):103–138, 1987.

[15] R. S. Michalski and R. E. Stepp. Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning*, volume 1. Margan Kaufmann Publishers, Inc., 1983.

[16] Niblack, Wayne, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The qbic project: Querying images by content using color, texture, and shape. In *Storage and Retrieval for Images and Video Databases, SPIE*, volume 1908, 1993.

[17] C. E. Shannon and W. Weaver. *The Mathematical Theory of Communication.* University of Illinois Press, Urbana, Ill, 1964.

References

[1] J. R. Bach, S. Paul, and R. Jain. A visual information management system for the interactive retrieval of faces. *IEEE Transaction on Knowledge and Data Engineering*, August 1993.

[2] T. Cai, N. Cercone, and J. Han. Attribute-oriented induction in relational databases. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 213–228. AAAI Press/The MIT Press, 1991.

[3] David K. Y. Chiu, Andrew K. C. Wong, and Benny Cheung. Information discovery through hierarchical maximum entropy discretization and synthesis. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*. AAAI Press/The MIT Press, 1991.

[4] Wesley W. Chu, Alfonso F. Cardenas, and Ricky K. Taira. KMeD: A knowledge-based multimedia medical distributed database system *Information System*, Volume 20, No. 2 pages 75-96, 1995.

[5] Wesley W. Chu and Q. Chen. Neighborhood and associative query answering. *Journal of Intelligent Information Systems*, 1(3/4), 1992.

[6] Wesley W. Chu and Kuorong Chiang. A distribution sensitive clustering method for numerical values. Technical Report 93-0006, UCLA Computer Science Department, 1993.

[7] Wesley W. Chu and Kuorong Chiang. Abstraction of high level concepts from numerical values in databases. In *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*, July 1994.

[8] Wesley W. Chu, M. A. Merzbacher, and L. Berkovich. The design and implementation of cobase. In *Proceedings of ACM SIGMOD*, Washington D. C., USA, May 1993.

to the fact that M-DISC considers information from multiple attributes at the same time.

Applying Lemma 1, it is easy to determine that in case (a), DISC will cluster {A,P} together in both dimensions. This yields good relaxation. In cases (b) and (c), DISC will cluster {A,P} together in one dimension and {B,P} together in the other. Using the Minimum Coverage rule[7], for any distribution, relaxation may cover either A or B and the decision is random. As a result, considering all 3 cases, it yields good relaxation in 75% of the time and yields poor relaxation in 25% of the time, while MTAH always yields good relaxation.

Case 2: A and B are in adjacent quadrants

If $d(A,P) \leq d(A,B)$, then relaxation using MTAH yields good results. If $d(A,P) > d(A,B)$, then relaxation using MTAH yields average results.

In case of single-attribute TAHs, there are some data distribution where DISC would cluster {A,P} together in one attribute and {B,P} together in the other. As in Case 1, relaxation guided by single-attribute TAHs returns poor answers. Relaxation using MTAH is superior since MTAH will always include the closest answer while single-attribute TAHs may miss it.

Case 3: A and B are in the same quadrant

As in Case 2, we can show that relaxation using MTAHs would be good or average, while single-attribute TAHs could miss the good answer.

The above analysis shows that using an MTAH offers better relaxation on multiple numerical attributes than using multiple single-attribute TAHs. MTAHs successfully retrieve the best answer in all cases, while single-attribute TAHs fail to retrieve the best answer in some cases. This is due

[7] The rule of Minimum Coverage is used as the relaxation policy for multiple single-attribute TAHs. This rule states that from N different TAHs, select the TAH that yields fewest number of answers so that it results in finer relaxation granularity.

Lemma 2 (Closest Distance for MTAHs) Given three points in a two-dimensional space, the M-DISC cutting algorithm will always cluster the two closest points together.

Proof: For three points in a two-dimensional space, MTAH clusters two of the points together, minimizing the relaxation error of the cluster. From (11), relaxation error of the cluster is:

$$RE(R, S) = d(R, S)/4$$

Since M-DISC minimizes RE(R,S), d(R,S) is minimized.

Consider P as the query point and divide the problem space into 4 quadrants, with P at the center. There are 3 possible topologies, depending on the relative positions of A and B, as shown in figure 5:

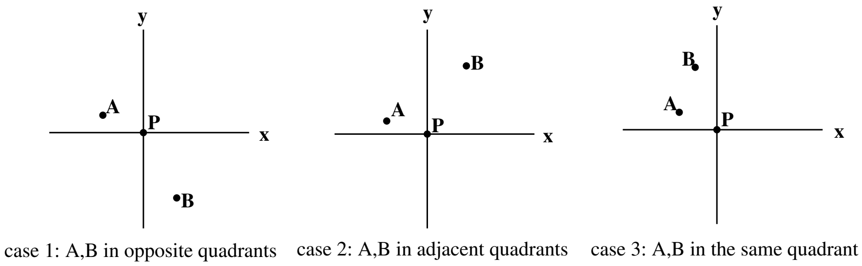

Figure 5: Topologies of 3 points P,A,B

Case 1: A and B are in opposite quadrants

In this case, the shortest Manhattan distance is PA. Applying Lemma 2 , M-DISC will cluster A,P together. Thus, using MTAH to guide relaxation always yields good results.

For single-attribute TAHs, there are three possibilities:

- (a) $|X_P - X_A| < |X_P - X_B|$ and $|Y_P - Y_A| < |Y_P - Y_B|$
- (b) $|X_P - X_A| < |X_P - X_B|$ and $|Y_P - Y_A| > |Y_P - Y_B|$
- (c) $|X_P - X_A| > |X_P - X_B|$ and $|Y_P - Y_A| < |Y_P - Y_B|$

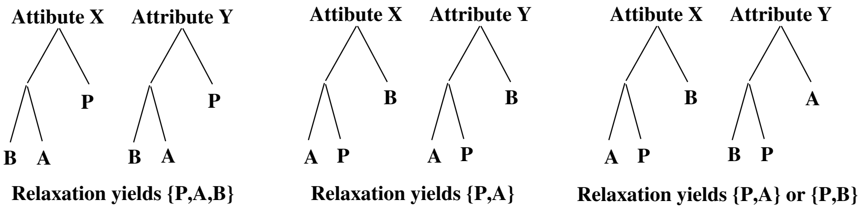

Figure 3: Relaxation using multiple single-attribute TAHs

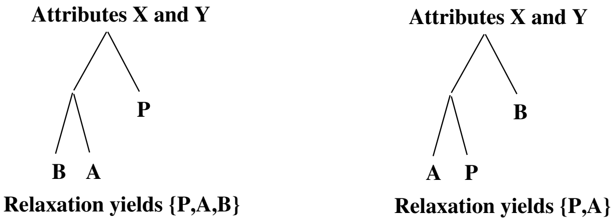

Figure 4: Relaxation using an MTAH

Analysis

Distance Measure

The distance between any two points $R(X_R, Y_R)$ and $S(X_S, Y_S)$ is

$$D(R,S) = (|X_R - X_S|)/RE_x + (|Y_R - Y_S|)/RE_y \qquad (12)$$

Since points are assumed to be randomly distributed, $RE_x = RE_y$. Thus

$$d(R,S) = |X_R - X_S| + |Y_R - Y_S| \qquad (13)$$

which reduces to the Manhattan distance between the points.

Lemma 1 (Closest Distance for Single TAHs) Given three points in a one-dimensional space, the DISC cutting algorithm always clusters the two closest points together.

Proof: In a single-dimensional case, $d(R,S) = |X_R - X_S|$. For three points A,B,C projected on a line, there are two possible cuts (AB, BC). DISC cuts on the longer of AB and BC to minimize relaxation error.

The generated TAHs and MTAHs are used in the *Cooperative Database System* (CoBase) [8] at UCLA for providing approximate answers to structured data and for approximate matching of image features. The approximate query answers derived from MTAHs are empirically evaluated in terms of accuracy, efficiency, and relaxation error. The results reveal that the approximate query answers derived from the MTAH generated by DISC are better than those derived from an index tree generated by the Maximum Entropy method.

Appendix: Comparison of Using MTAH to Multiple Single-Attribute TAHs for Query Relaxation On Multiple Numerical Attributes

We shall use a simplified model to illustrate how the dependence relationships which are captured by MTAHs but not captured by single-attribute TAHs is useful in guiding the relaxation process.

Considering data with two attributes, we can represent it as a 2-dimensional problem space with tuples represented as points distributed randomly on it. A single point P is chosen randomly as the query condition. The area of consideration is restricted to P and the two closest (the notion for distance will be defined below) points to P, denoted as A and B. The problem is to consider the relative effects of using an MTAH versus using 2 single-attribute TAHs to relax the query conditions on the 2 dimensions in certain relaxation order. The goal is to obtain the closest approximate answers.

Without loss of generality, consider A as the closest point to P. There are 3 possible results of relaxation. The answers resulting from traversing the MTAH and single-attribute TAHs may cover {A,P}, {B,P}, or {A, B, P} (see Figures 3 and 4). Since A is the closest point to P, {A,P} is considered to be the *good* answer, {B,P} is considered to be the *poor* answer, and {A,B,P} is considered to be the *average* answer.

quality (0.151) of the node shows the difference is small, and therefore, the quality of the TAHs is still good and need not be updated.

When the derived approximate answers significantly deviate from the expected quality, then the quality of the TAH is deemed to be inadequate and a regeneration is necessary. The following incremental TAH regeneration procedure can be used. First, identify the node within the TAH that has the worst query relaxations. Apply partial TAH regeneration for all the database instances covered by the node. After several such partial regenerations, we then initiate a complete TAH regeneration.

8 Conclusion

In this paper, we present a Multi-attribute DIstribution Sensitive Clustering method (M-DISC) for numerical attribute values. For m attribute of n distinct values, M-DISC generates a Multi-attribute Type Abstraction Hierarchy (MTAH) in $O(m^2n^2(\log n)^2)$ time. gM-DISC, a greedy version of M-DISC, yields comparable result with that of M-DISC but only takes $O(mn\log n)$ to generate an MTAH. Thus, it is suitable for tables with a large number of tuples.

For clustering m dependent attributes, we have shown that the relaxation error from using MTAH is smaller than from m single-attribute TAHs because single-attribute TAHs cannot take the attribute dependency relationship into consideration. Further, using MTAH is more efficient than using multiple single-attribute TAHs in processing queries because MTAH requires fewer relaxation steps than multiple single-attribute TAHs, thus reducing the number of database access.

We have used M-DISC to generate MTAHs for a large transportation database which consists of 94 relations, the largest one of which has 12 attributes and 195,598 tuples. M-DISC generates approximately 400 numerical TAHs and MTAHs in less than a few hour of processing time on a Sun Sparc 10 Workstation.

	MTAH	gMTAH	ME-Tree	E-S
accuracy	0.85	0.84	0.68	1.0
efficiency	0.54	0.53	0.64	0.011
relaxation error	1.14	1.17	1.57	1.0

Table 7: Performance comparison of the MTAH, gMTAH, ME tree, and the exhaustive search

efficient than the exhaustive search, yet provide answers close to those generated by the exhaustive search with errors less than 17%. We also note that MTAH and gMTAH yield comparable performances. This confirms that the heuristics used in the greedy algorithm gM-DISC provides good approximation.

7 Maintenance of TAHs

Since the quality of TAH affects the quality of derived approximate answers, TAHs should be kept up to date. One simple way for maintaining TAHs is to regenerate them whenever an update occurs. This approach is not desirable, however, because it causes a lot of overhead for the database system. Although each update changes the distribution of data (thus changing the quality of the corresponding TAHs), it by its self may not be significant enough to warrant a TAH regeneration. TAH regeneration is only necessary when the cumulative effect of updates has greatly degraded the TAHs.

The quality of a TAH can be monitored by comparing the derived approximate answers to the expected relaxation error which is computed at TAH generation time and recorded at each node of the TAH. For example, consider the query in section 5.1. The relaxation error of the first approximate answer is 0.168. From the TAHs in Figure 2, the expected quality of this approximate answer is 0.131 for SQUARE-FEET and 0.170 for WEIGHT, with an average quality of $(0.131 + 0.170)/2 = 0.151$. Comparison between the actual quality (0.168) and the expected

6.2 Multiple Attributes

In addition to the relaxation error, we shall introduce two additional performance measures, *accuracy* of the answer and *efficiency* of the retrieval[5], to compare the performance of DISC with ME:

$$\text{accuracy of the answer} = \frac{\text{retrieved relevant answers}}{\text{all relevant answers}}$$
$$\text{efficiency of the retrieval} = \frac{\text{retrieved relevant answers}}{\text{all retrieved answers}}$$

where "all relevant answers" are the best k answers determined by exhaustive search. In general, there is a trade-off between the two measures: the higher the accuracy of the answer, the lower the efficiency of the retrieval.

We generate two MTAHs (one from the algorithm M-DISC and another from the greedy gM-DISC) and an ME tree[6] based on attributes Longitudes and Latitudes of 972 geographical locations from a transportation database. We generate 500 queries with the form: "find the k locations nearest to $(long, lat)$" where k is randomly selected from 1 to 20, and $long$ and lat are randomly generated based on the distributions from the location relation in the database. To answer the query using the MTAHs or the ME tree, we first locate the most specific node in the tree that "covers" k' locations where $k' \geq k$. Then we compute the k' distances to $(long, lat)$ and use them to select the nearest k locations. The accuracy, efficiency, and the relaxation error from the best k answers to the target location $(long, lat)$, averaged over the 500 queries, are shown in Table 7. For easy comparison, performance is given relative to that of the exhaustive search (E-S). We label the MTAH generated by the greedy algorithm gM-DISC as gMTAH.

Notice that the answers provided by the MTAHs (i.e., MTAH and gMTAH) are more accurate (i.e., closer to the target location) than those by the ME tree. Further, the MTAHs are more

[5]These measures are known as *recall* and *precision* in Information Retrieval.

[6]Since the classification tree generated by the ME method is balanced, it can be viewed as an efficient index.

as the similar images of contour 6.

For contours with more complex shapes, additional features may be needed to capture the shape characteristics. One approach is based on a prior shape knowledge to decompose the complex shape into several simple ones [18]. We can represent these decomposed simple shapes by an object-oriented shape model which allows us to selectively combine these simpler shapes for retrieving a specific image.

In many current image database systems, such as VIMS [1], similar images are retrieved based on mean and standard deviations of each extracted feature. As a result, similarity is analyzed only based on each attribute separately. Since these features may be dependent, such methods represent a shortcoming. In our approach, however, all the related features can be jointly considered, and thus, it yields more accurate feature clustering.

6 Performance Comparison of DISC with ME

6.1 Single Attribute

Empirical results based on a large transportation database show that clusters discovered by DISC have less relaxation error than those by the Maximum Entropy method (ME). It can be shown [6] that only when the data distribution is symmetrical at the median will the ME method and DISC perform equally well. For skewed distributions, which hold for most data in the transportation database, DISC performs better than ME. Empirical results show that the performance improvement of DISC over ME increases as the skewness increases.

TUMOR ID	AREA ($pixel^2$)	CIRCULARITY	EXTRUSIVENESS
1	5952.5	1.15	1.03
2	819.5	2.17	1.02
3	743.5	2.8	1.04
4	699	1.1	1
5	307	1.12	1
6	1753.5	1.52	1.07
7	2203.5	1.59	1.04
8	2748.5	1.59	1.05
9	2754	1.27	1
10	2877.5	1.18	1
11	1478.5	5.17	2.16

Table 6: Measurement of features for eleven lung tumors

For the above answers, the relaxation error is 0.334, which is greater than the previous one because the values deviate more from the specified ones in this case.

Thus, from the above example, we see that the TAHs generated by DISC can be used for relaxing query conditions to obtain "neighborhood answers" as well as their quality measures.

5.2 Application to Feature-based Image Retrieval

Images can be retrieved based on features and contents [4, 16]. Extracting the essential features that can capture the specific aspects of the image object depends on context and image domains. Consider lung tumors as an example, three features can be used to classify the shape of lung tumors : *area*, *circularity*, and *extrusiveness*, where *circularity* is defined as $\frac{perimeter^2}{4 \times \pi \times area}$, and *extrusiveness* is defined as $\frac{area\_of\_convex-hull\_of\_contour}{area}$ [20] (see Figure 6. Based on this set of features, we can use M-DISC to classify the images and generate an MTAH, as shown in Figure 6.

We can retrieve images with similar feature characteristics from the MTAH. The MTAH leaf nodes represent tumor contours obtained from CT scanned chest images. For example, contour 6 (in Figure 6) is taken from the image in Figure 7. Using this MTAH, contours 7 and 8 are returned

The corresponding SQL query is

select CARGO-ID

from CARGOS

where SQUARE-FEET = 300 and WEIGHT = 740.

The query conditions are too specific; no answers are found for the query. To obtain approximate answers, the query is modified by relaxing cargo size and weight according to the TAHs for SQUARE-FEET and WEIGHT (see Figure 2). (These TAHs are generated by DISC from a transportation database for the CARGOS relation.) As a result, the query is modified to

select CARGO-ID

from CARGOS

where 294 $\leq$ SQUARE-FEET $\leq$ 300 and 737 $\leq$ WEIGHT $\leq$ 741.

The modified query is submitted to the database and the following cargo is returned:

CARGO-ID	SQUARE-FEET	WEIGHT
10	296	740

The quality of the answer can be measured by its relaxation error from equation (11). From the CARGOS relation, we obtain Δ(SQUARE-FEET) = 11.95 and Δ(WEIGHT) = 9.88. Thus, assuming the two attributes SQUARE-FEET and WEIGHT are equally important, the relaxation error is 0.168. This quality measure corresponds to a 16.8% relaxation when a 100% relaxation is "total relaxation" and the condition is simply dropped.

If more answers are needed, the conditions can be further relaxed by moving one more step up the TAHs:

select CARGO-ID
from CARGOS
where 294 $\leq$ SQUARE-FEET $\leq$ 306 and 737 $\leq$ WEIGHT $\leq$ 749.

The following four cargos are returned for this query:

CARGO-ID	SQUARE-FEET	WEIGHT
10	296	740
21	301	737
30	304	746
44	306	745

level clusters. The aircrafts C-23,...,L-100-30 have *narrow* runway (Runway-Width ≤ 75), while the other aircrafts have *wide* runway (Runway-Width > 75). The *wide* aircrafts are further classified by Max-Weight: the aircrafts L-727-100,...,L-1011-500 are *light* (Max-Weight $\leq 496,000$) while the aircrafts KC-10,...,B-747-200F are *heavy*(Max-Weight $> 496,000$). The rest of the MTAH can be understood in the same manner.

Note that the relaxation errors are computed (from Eq 11) and shown at each node. Note that the relaxation errors of the MTAH are monotonically decreasing from the root to the leaves, indicating the quality of nodes increases towards the leaves.

5 Applications

5.1 Application to Approximate Query Answering

We shall now use an example to show how the TAHs generated by DISC can be used for providing approximate answers.

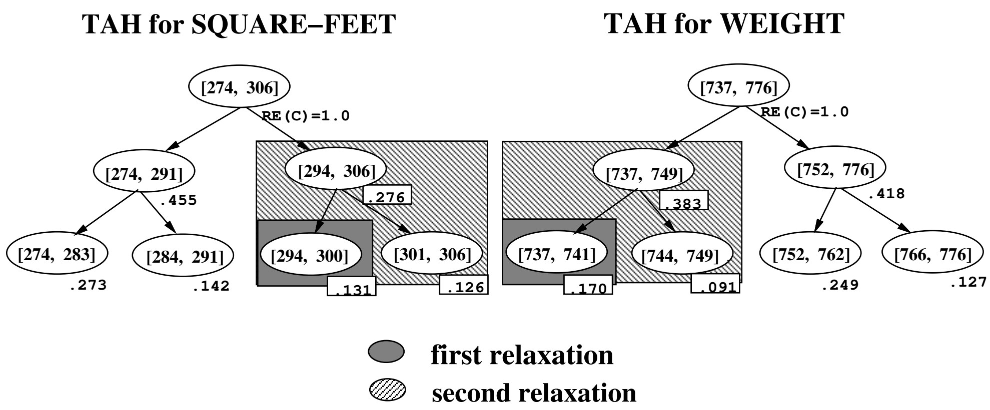

Figure 2: TAHs for SQUARE-FEET and WEIGHT generated by DISC from a transportation database. Higher nodes have higher relaxation errors.

Example. Consider the query "find a cargo with size 300 square feet and weight 740 kg."

Aircraft	Length		Runway	Weight		Storage
Type	Takeoff	Landing	Width	Max.	Empty	Space
C–5	13600	5000	90	769000	665000	21508
C–141B	9000	5000	90	343000	239000	10454
KC–10	11800	5400	90	590000	414000	11600
C–130E	6250	3000	60	173700	NULL	5173
C–130H	6250	3000	60	173700	NULL	5173
C–9A	8100	3200	10	108000	90000	4325
C–12	4500	4500	75	12500	11000	915
C–21	6000	3400	75	18300	13500	747
C–20	5000	5000	75	69700	44000	2518
C–23	4000	3000	60	22900	16200	1690
B–707–300	10800	7000	90	336600	230000	5600
DC–8–33	9400	5500	90	315000	224000	10000
DC–8–50	9400	5500	90	315000	224000	10000
DC–8–61	10400	6100	90	325000	261000	8100
DC–8–62	11500	6100	90	350000	230000	8825
DC–8–63	10400	6100	90	355000	261000	10800
DC–8–71	8850	6100	90	355000	261000	10800
DC–8–73	9800	6100	90	355000	261000	10800
DC–10–10	9500	5800	90	440000	335000	11000
DC–10–30	11850	6100	90	572000	391000	11600
DC–10–40	10600	5800	90	572000	391000	11600
B–747SP	7500	5600	90	696000	410000	14100
B–747–100	9500	6600	90	750000	526400	17500
B–747–200C	11800	6600	90	775000	590000	17500
B–747–200F	10500	6600	90	775000	590000	17500
L–1011–100	10640	7300	90	466000	320000	10800
L–1011–500	9760	7300	90	496000	338000	10800
B–727–100	9200	4800	90	207500	160000	5600
L–100–30	6000	4900	75	155000	122000	5800

Table 5: The database table AIRCRAFT

using gM-DISC is $O(mn^2)$ where m is the number of attributes and n is the number of distinct values. The average case time complexity is $O(mn \log n)$.

4.4 An Example

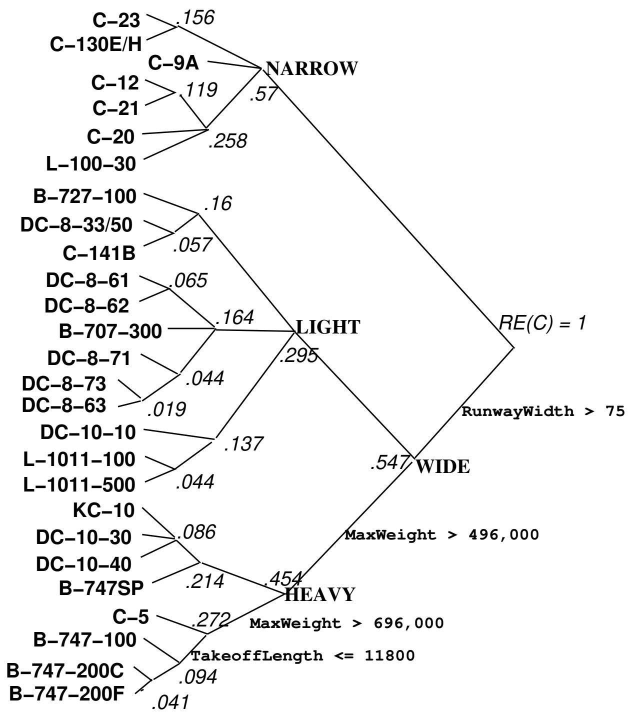

Figure 1: The MTAH for *AIRCRAFT*

We shall present an example here using the *AIRCRAFT* relation shown in Table 5. We use M-DISC to generate an MTAH based on the six numerical attributes. Each attribute is assumed to be equally important in characterizing an aircraft. In certain cases, however, this may not be true and different *weights* can be assigned to attributes for a better classification of objects. Using equal weights, the resulting MTAH is shown in Figure 1.

One advantage of MTAH is that it is easy to understand. Each node of the MTAH stores conditions that all instances covered by the node must satisfy. In addition, each node stores the cuts information which shows *how* the objects are partitioned. For example, consider the three top

Algorithm gM-DISC(C)

```
if the number of objects in C < T, return /* T is a threshold */
for each attribute a = 1 to m
    for each possible binary cut h
        compute ΔRE_a for h
        if ΔRE_a > MaxΔRE then /* remember the best cut */
            MaxΔRE = ΔRE_a, BestAttribute = a, cut = h
partition C based on cut of the attribute BestAttribute
let the resultant sub-clusters be C_1 and C_2
    call gM-DISC(C_1) and gM-DISC(C_2)
```

Table 4: Greedy M-DISC algorithm: gM-DISC

4.3 gM-DISC: a Greedy M-DISC Algorithm

The time complexity of M-DISC can be greatly reduced if we are willing to accept an approximation. We shall present the heuristic for obtaining the sub-optimal cuts and the resulted *greedy* M-DISC algorithm here. Later, we shall show that the performance of the greedy M-DISC is comparable with that of the non-greedy M-DISC.

For ease of discussion, we shall say a partition is *disjointed* on an attribute a if the values of a are grouped into non-overlapping ranges. In general, a partition will have only one disjointed attribute which among all attributes will have the largest reduction of relaxation error. Thus, the magnitude of CU is generally dominated by the reduction of relaxation error on the disjointed attribute. (Recall that the meaning of CU is the relaxation error reduction due to the partition.) Therefore, we may replace CU with the relaxation error reduction on the disjointed attribute without losing too much accuracy.

Let ΔRE_a be the reduction of relaxation error for attribute a. Replacing CU with ΔRE_a in M-DISC, we have the greedy M-DISC (shown in Table 4).

The relaxation error reduction ΔRE_a used in gM-DISC is normalized such that ΔRE_a from different attributes can be compared. The worst case time complexity for generating an MTAH

Algorithm M-DISC(C)

if the number of objects in $C < T$, return /* T is a threshold */
for each attribute $a = 1$ to m
  for each possible binary cut h
    compute CU for h
    if $CU > MaxCU$ then /* remember the best cut */
      $MaxCU = CU$, $BestAttribute = a$, $cut = h$
partition C based on *cut* of the attribute *BestAttribute*
let the resultant sub-clusters be C_1 and C_2
  call M-DISC(C_1) and M-DISC(C_2)

Table 3: Multi-attribute DISC (M-DISC) algorithm

MTAHs for query relaxation, interrelated attributes can always be relaxed together. Approximate answers can be derived efficiently using MTAH because only a single query modification and database access is necessary. The approximate answers derived by using MTAH are better than those derived by using multiple TAHs.[4]

To cluster objects of multiple attributes, DISC can be extended to M-DISC (shown in Table 3). The generated multi-dimensional TAHs are called MTAHs. The algorithm DISC is a special case of M-DISC, and TAH is a special case of MTAH.

Let us now consider the time complexity of M-DISC. Let m be the number of attributes and n be the number of distinct attribute values. The computation of relaxation error for a single attribute takes $O(n \log n)$ to complete [7]. Since the computation of CU involves computation of relaxation error for m attributes, its complexity is $O(mn \log n)$. The nested loop in M-DISC is executed mn times, so the time complexity of M-DISC is $O(m^2 n^2 \log n)$. To generate an MTAH, it takes no more than n calls of M-DISC, therefore, the worst case time complexity of generating an MTAH is $O(m^2 n^3 \log n)$. The average case time complexity is $O(m^2 n^2 (\log n)^2)$ since M-DISC needs only to be called $\log n$ times on the average.

[4] A more detailed comparison between MTAH and multiple TAHs is given in the Appendix.

Algorithm N-ary Partition(C)
let C_1 and C_2 be the two sub-clusters of C
compute CU for the partition C_1, C_2
for $N = 2$ to $n - 1$
    let C_i be the sub-cluster of C with maximum relaxation error
    call BinaryCut to find the best sub-clusters C_{i1} and C_{i2} of C_i
    compute and store CU for the partition $C_1, ..., C_{i-1}, C_{i1}, C_{i2}, C_{i+1}, ..., C_N$
    if current CU is less than the previous CU
        stop
    else
        replace C_i by C_{i1} and C_{i2}
/* the result is an N-ary partition of C */

Table 2: The N-ary partition algorithm

4.2 The Clustering Algorithm for Multiple Attributes

Query relaxation for multiple attributes using multiple single-attribute TAHs relaxes each attribute independently disregarding the relationships that might exist among attributes. This may not be adequate for the applications where attributes are dependent.[3] In addition, using multiple TAHs is inefficient since it may need many iterations of query modification and database access before approximate answers are found. Furthermore, relaxation control for multiple TAHs is more complex since there are a large number of possible orders for relaxing attributes. In general, we can only rely on simple heuristics such as *best first* or *minimal coverage first* to guide the relaxation. These heuristics cannot guarantee best approximate answers since they are rules of thumb and are not necessarily accurate.

Most of the above mentioned difficulties can be overcome by using Multi-attribute TAH (MTAH) for the relaxation of multiple attributes. Although MTAHs can be generated from any set of attributes, MTAHs should be generated only from *semantically dependent* attributes. Using

[3]Dependency here means that all the attributes as a whole define a coherent concept. For example, the length and width of a rectangle are said to be "semantically" dependent. This kind of dependency should be distinguished from the *functional dependency* in database theory.

Algorithm DISC(C)

if the number of distinct values $\in C < T$, return /* T is a threshold */
let *cut* = the best cut returned by **BinaryCut**(C)
partition values in C based on *cut*
let the resultant sub-clusters be C_1 and C_2
call DISC(C_1) and DISC(C_2)

Algorithm BinaryCut(C)

/* input cluster $C = \{x_1, ..., x_n\}$ */
for $h = 1$ to $n - 1$ /* evaluate each cut */
Let P be the partition with clusters $C_1 = \{x_1, ..., x_h\}$ and $C_2 = \{x_{h+1}, ..., x_n\}$
compute category utility CU for P
if $CU < MinCU$ then
$MinCU = CU$, $cut = h$ /* the best cut */
Return *cut* as the best cut

Table 1: The algorithms DISC and BinaryCut

In [7], an implementation of the algorithm BinaryCut is presented whose time complexity is $O(n)$. Since DISC needs to execute BinaryCut $n - 1$ times at most to generate a TAH, the worst case time complexity of DISC is $O(n^2)$. (The average case time complexity of DISC is $O(n \log n)$.)

4.1.1 N-ary Partitioning

N-ary partitions can be obtained from binary partitions by a hill climbing method. Starting from a binary partition, the sub-cluster with greater relaxation error is selected for further cutting. We shall use CU as a measure to determine if the newly formed partition is better than the previous one. If the CU of the binary partition is greater than that of the tri-nary partition, then the tri-nary partition is dropped and the cutting is terminated. Otherwise, the tri-nary partition is selected and the cutting process continues until it reaches the point where a cut decreases CU. The procedure is outlined in Table 2. Note that N-ary TAH can be generated by replacing BinaryCut with N-ary Partition in DISC algorithm.

as significant as a difference of 265,602.38 in Max-Weight.

Given a cluster of n m-attribute tuples $C = \{t_1, ..., t_n\}$, the relaxation error for C is defined as the average pair-wise distance among tuples in C, that is,

$$RE(C) = \frac{1}{m} \sum_{i=1}^{n} \sum_{j=1}^{n} P(t_i)P(t_j)D(t_i, t_j) \tag{11}$$

where $P(t_i)$ and $P(t_j)$ are the probabilities of tuples t_i and t_j, respectively. The division by m in (11) normalizes RE(C) per attribute and allows us to compare relaxation errors computed from different numbers of attributes. The category utility (CU) for multiple attributes can be obtained by simply substituting $RE_1(C)$ in Eq (9) by its multi-attribute counterpart $RE(C)$ in Eq (11).

4 The Clustering Algorithm

We shall now present a class of DISC algorithms for clustering numerical values. We shall present the algorithm for a single attribute, and then extend it for multiple attributes.

4.1 The Clustering Algorithm for a Single Attribute

Given a cluster with n distinct values, the number of partitions is exponential with respect to n, so the best partition according to (9) takes exponential time to find. To reduce computation complexity, we shall only consider binary partitions (i.e., $N = 2$ in (9)). Later we shall show a simple hill climbing strategy can be used for obtaining N-ary partitions from binary partitions.

Our method is top down: we start from one cluster consisting of all the values of an attribute, and then we find "cuts"[2] to recursively partition the cluster into smaller clusters. The partition result is a concept hierarchy called Type Abstraction Hierarchy (TAH). The clustering algorithm is called the DISC (DIstribution Sensitive Clustering) Method and is given in Table 1.

[2]A cut c is a value that separates a cluster of numbers $\{x|a \leq x \leq b\}$ into two sub-clusters $\{x|a \leq x \leq c\}$ and $\{x|c < x \leq b\}$.

3.1 Relaxation Error for Multiple Attributes

As mentioned above, relaxation error is the expected pair-wise difference between values in a cluster. To extend the notion of relaxation error from a single attribute to multiple attributes, we shall consider distance between tuples instead of difference between values. Given two m-attribute tuples $t_i = \{x_1, ..., x_m\}$ and $t_j = \{y_1, ..., y_m\}$, their distance is defined as

$$D(t_i, t_j) = \sum_{k=1}^{m} W_k \frac{|x_k - y_k|}{\Delta_k} \tag{10}$$

where W_k and Δ_k are the weight and the normalization constant for the k-th attribute, respectively. W_k can be assigned to reflect relative importance among attributes. Δ_k is necessary for summing up differences from different attributes because different attributes have different distributions. For example, consider two attributes from the *AIRCRAFT* relation (Table 5): Runway-Width and Max-Weight. Max-Weight has a much greater value range, so its value differences tend to be much greater than that of Runway-Width and will dominate $D(t_i, t_j)$. Thus, normalization is necessary for summing up pair-wise differences from different attributes.

There are two possible normalization factors. One is $x_{max} - x_{min}$, and another is $RE_1(X)$, the relaxation error of the attribute X. Both factors are based on pair-wise difference: $x_{max} - x_{min}$ is the *maximum* pair-wise difference and $RE_1(X)$ is the *average* pair-wise difference. $x_{max} - x_{min}$ is much easier to compute but is easily subjected to influence by out-liers of values. $RE_1(X)$ is more costly to compute but is not easily subjected to influence from out-liers. Since the cost of computing $RE_1(X)$ is small (linear with respect to the number of distinct values in X), we select $RE_1(X)$ as the normalization factor

To see the effect of the normalization factor, consider the attributes Runway-Width and Max-Weight again. The relaxation error for Runway-Width and Max-Weight are 12.89 and 265,602.38, respectively. Using $RE_1(X)$ as the normalization factor, a difference of 12.89 in Runway-Width is

Equation (5) cannot serve as a quality measure for numerical values as illustrated in the following: consider the two clusters $C_1 = \{0, 100\}$ and $C_2 = \{0, 1\}$ as mentioned in section 1. According to (5), C_1 is as good as C_2: $G(C_1) = G(C_2) = 0.5$. Intuitively, this is unsatisfying because the values in C_2 are much closer to each other than those in C_1. As a result, a label such as *small* can be attached to C_2 but not to C_1.

Based on the above observation, a good quality measure for numerical values can be derived by substituting $s(x_i, x_j)$ in (5) with the absolute difference between x_i and x_j. The result is called the *relaxation error* of C, $RE_1(C)$. Formally,

$$RE_1(C) = \sum_{i=1}^{n}\sum_{j=1}^{n} P(x_i)P(x_j)|x_i - x_j| \tag{6}$$

That is, $RE_1(C)$ is the average pair-wise difference among values in C. Using $RE_1(C)$ for the above example, we have $RE_1(C_1) = 50$ and $RE_1(C_2) = 0.5$. Clearly, C_2 is much better than C_1.

$RE_1(C)$ can also be interpreted from the standpoint of query relaxation. Let us define the *relaxation error of* x_i, $RE_1(x_i)$, as the average difference from x_i to x_j, j=1,...,n. Formally,

$$RE_1(x_i) = \sum_{j=1}^{n} P(x_j)|x_i - x_j| \tag{7}$$

where $P(x_j)$ is the occurring probability of x_j in C. $RE_1(x_i)$ can be used to measure the quality of an approximate answer where x_i in a query is relaxed to x_j, j=1,...,n. Summing $RE_1(x_i)$ over all values x_i in C, we have

$$RE_1(C) = \sum_{i=1}^{n} P(x_i)RE_1(x_i). \tag{8}$$

Thus, $RE_1(C)$ is the expected error of relaxing any value in C.

Using relaxation error, the category utility can be defined as the relaxation error reduction per sub-cluster, that is,

$$\begin{aligned} CU &= \frac{\sum_{k=1}^{N} P(C_k)[1 - RE_1(C_k)] - [1 - RE_1(C)]}{N} \\ &= \frac{RE_1(C) - \sum_{k=1}^{N} P(C_k)RE_1(C_k)}{N}. \end{aligned} \tag{9}$$

a system parameter called *acuity* which is the minimum value allowed for σ. *Acuity* avoids the infinite quality problem, but it may cause undesirable data classification for approximate query answering. For example, an impulse value, say v_i, cannot be the only value in a concept; to satisfy the *acuity* constraint, the concept is forced to include other values, say v_j. Thus, whenever v_j is relaxed, it is relaxed to $\{v_i, v_j\}$ which may be a very poor approximation for v_j.

To remedy these shortcomings, we develop a new goodness measure for the classification of numerical data in view of approximate query answering.

3 Relaxation Error - A Goodness Measure for Clustering Numerical Values

To deal with numerical values, we need to generalize category utility. To simplify our presentation, let us consider the single-attribute case. For a class $C = \{x_1, ..., x_n\}$, (3) reduces to

$$G(C) = \sum_{i=1}^{n} P(x_i)^2 \tag{4}$$

where x_i is the i-th distinct attribute value and $P(x_i)$ is the occurring probability of x_i in C.

The goodness measure can be interpreted as follows. The class C in (4) is represented by an urn of balls, each of them representing an attribute value. The number of balls of a particular color represents the frequency of occurrence of the corresponding attribute value. We randomly draw two balls from the urn, one at a time with replacement. If the two balls have the same value, we score 1. Otherwise, we score 0. If we do this experiment a large number of times, the expected score we have will be the goodness of the cluster $G(C)$. Let $s(x_i, x_j)$ be the score for the drawn values x_i and x_j, then $s(x_i, x_j) = 1$ if $x_i = x_j$ and 0 otherwise. Thus, (4) becomes

$$G(C) = \sum_{i=1}^{n} \sum_{j=1}^{n} P(x_i) P(x_j) s(x_i, x_j). \tag{5}$$

defined as the increase in the average class *goodness*[1]. That is,

$$CU(C_1, ..., C_N) = \frac{\sum_{k=1}^{N} P(C_k)G(C_k) - G(C)}{N} \tag{1}$$

where $P(C_k)$ is the occurrence probability of C_k in C, and $G(C_k)$ and $G(C)$ are the goodness functions for C_k and C, respectively. That is,

$$G(C_k) = \sum_{a \in A} \sum_{x_i^a \in X_k^a} P(x_i^a)^2 \tag{2}$$

$$G(C) = \sum_{a \in A} \sum_{x_i^a \in X^a} P(x_i^a)^2 \tag{3}$$

where A is the set of all the attributes, and X_k^a and X^a are the distinct values of attribute a in C_k and C respectively.

COBWEB cannot be used for abstracting numerical data; it only deals with categorical data. Moreover, its classification tree serves as *the database* for the instances and requires a large storage space. Furthermore, matching of objects with existing classes is time consuming. For providing approximate answers, we want to build a classification tree *in addition to* the original database. The tree can be viewed as an index so that its storage and traversal should be very efficient.

CLASSIT [12], an extension of the COBWEB system, classifies numerical attributes where each attribute is normally distributed. Concepts are represented in terms of mean and standard deviation (σ). $1/\sigma$ is used as the quality measure for a concept and the data is classified to maximize the weighted sum of $1/\sigma$. The problem of using CLASSIT for approximate query answering is that data are assumed to be normally distributed. This is not true in many cases; our experience with a transportation database shows many numerical attributes are skewed and multi-modal. Further, we also observed multiple "impulse values" which represent occurrences of certain attribute values with a very high frequency. In addition, CLASSIT cannot represent concepts with a single value because $\sigma = 0$, making $1/\sigma = \infty$. To overcome this problem, CLASSIT introduces

[1]We introduce the term *goodness* for ease of presenting the quality of approximate query answers.

and the *value* distributions of data. In this paper, we propose a DIstribution Sensitive Clustering (DISC) method that considers both the frequency and the value distributions in discovering high level concepts from data.

The rest of the paper is organized as follows. After a brief discussion of prior related work, we develop the notion of *relaxation error* as a quality measure for clusters. Then we present the DISC algorithm for generating concept hierarchies, Type Abstraction Hierarchies (TAH), that minimize the relaxation error for a single attribute as well as multiple attributes. Next, we present applications of TAH for providing approximate query answers and for feature-based image retrieval. To show the effectiveness of our discovered knowledge for approximate query answering, we compare the results derived from TAH generated by DISC with those of the traditional index tree. Finally, we address the issue of maintaining TAHs and offer a feasible solution of keeping TAHs up to date.

2 Related Work

Prior work in discretization aims at decreasing cardinality of data by maximizing/minimizing certain heuristic measures. A commonly used measure is the information entropy [17]. It can be shown that the entropy is maximum when the data is partitioned most evenly. (We call this the ME method [3, 21].) However, no semantic meaning is attached to the resultant clusters because the discretization is concerned with the frequency distribution rather than the value distribution in the cluster. Therefore, the ME method is not suitable for abstracting numerical data.

COBWEB [10], a conceptual clustering system, uses *category utility* (CU) as a quality measure to classify the objects described by a set of attributes into a classification tree. Formally, for a partition from a class C to N mutually exclusive classes $C_1, ..., C_N$, the category utility (CU) is

be used to process queries intelligently [2, 11, 9, 5]. In particular, abstraction can be used to derive approximate answers – when the objects requested by a query are not available, the query conditions can be relaxed to their corresponding abstraction where "neighborhood objects" can be found and returned as the approximate answers.

For clustering objects into "neighborhoods," two methods can be used: statistical clustering and numerical taxonomy [19], or conceptual clustering [15, 13, 14]. In statistical clustering and numerical taxonomy, most similarity metrics are defined between pairs of objects. Objects are pair-wise clustered in a bottom up manner until all objects are in a single cluster. In conceptual clustering, the "goodness measures" are usually defined for the overall *partitioning* of objects instead of for pairs of objects. Clustering methods are designed to maximize the goodness measure. We find the conceptual clustering technique more suitable for approximate query answering [5, 8] since the goodness measure can be directly related to the expected quality of approximate answers. By minimizing the expected error associated with the query relaxation, we are able to derive the *best* clustering of objects.

Most current *conceptual clustering* systems only deal with non-numerical values. That is, the values are categorical and can only be equal or not equal. These systems only consider *frequency distribution* of data because they are concerned with the *grouping* of values rather than the *values* themselves. Such clustering, however, is inadequate for providing approximate answers as illustrated in the following example. Consider two clusters $C_1 = \{0, 100\}$ and $C_2 = \{0, 1\}$. C_1 and C_2 are equivalent based on frequency distribution alone because they both have two distinct values. As approximate answers, however, they are very different because the values in C_2 are much closer to each other than those in C_1. Thus, for defining a neighborhood of objects, C_2 is much better than C_1.

For discovering abstraction, therefore, a clustering method must consider both the frequency

An Error-based Conceptual Clustering Method for Providing Approximate Query Answers *

Wesley W. Chu, Kuorong Chiang, Chih-Cheng Hsu, Henrick Yau †
Computer Science Department
University of California, Los Angeles

Abstract

A conceptual clustering method is proposed for discovering high level concepts of numerical attribute values from databases. The method considers both *frequency* and *value* distributions of data. Thus it is able to discover relevant concepts from numerical attributes. The discovered knowledge can be used for representing data semantically and for providing approximate answers when exact ones are not available.

Our knowledge discovery approach is to partition the data set of one or more attributes into clusters that minimize the *relaxation error*. Efficient clustering algorithms are developed which can be recursively called to generate a concept hierarchy. Applications of such clustering method to structured data and feature-based image are given. The effectiveness of our clustering method is demonstrated by applying it to a large transportation database for approximate query answering.

Key words: approximate query answering, type abstraction hierarchy, conceptual clustering, discretization, knowledge discovery in databases, feature-based image retrieval.

1 Introduction

Knowledge discovered from databases can be used to abstract the data into high level concepts. The discovered concept, or abstraction, makes it easier for users to understand the data and can

*This work is supported in part by ARPA contract F30602-94-C-020

†The authors may be reached at {wwc,kuorong,ccheng,henrick}@cs.ucla.edu