Journal of International Information Management

Volume 3
Issue 3 *Special Edition*

Article 11

1994

A clustering algorithm to identify information subsystems

Prafulla Joglekar
LaSalle University

Madjid Tavana
La Salle University

Snehamay Banerjee
Drexel University

A clustering algorithm to identify information subsystems

Prafulla Joglekar
Madjid Tavana
La Salle University

Snehamay Banerjee
Drexel University

ABSTRACT

This paper presents an algorithm to cluster the entities and relationships identified by database designers into a set of internally cohesive subsystems. Our algorithm is based on the calculation of a distance score that is inversely related to the similarity of interactions of a pair of entities with the relationships in a binary entity-relationship matrix. Our algorithm avoids manual manipulation of rows and columns required by some of the available approaches (Feldman et al., 1986; Teorey, et al., 1989). It has been implemented on a PC, and does not require a super computer as the Wei and Gaither (1990) method does. Using a part-machine clustering problem presented by King (1980), we also show that our algorithm is superior to King's "rank order cluster" algorithm which requires manual intervention to suppress exceptional entries before one can arrive at the final solution. Directions for further research are identified.

INTRODUCTION

The design of large and complex information systems relies on the successful development of data models. Many models are available to specify the relationship among information subsystems. While proper implementation of these models can lead to substantial business advantages, such implementation decisions are often seen as the domain of technical experts, such as database designers. Often these technical experts have a very limited understanding of the meaning of the data in a subsystem (Date, 1990), and what specific business implications result from the physical design of a subsystem.

Semantic modeling, such as the entity-relationship (ER) modeling (Chen, 1976), is useful in developing such an understanding. The basic building blocks of ER modeling are the entities and the relationships. An entity is a physical or logical object about which we are interested in storing information. Any entity must be distinguished from all other interactions (Martin, 1982). Relationships create the data about entities which an organization wants to capture in its databases.

In a large database with many entities and corresponding relationships, it is difficult to understand and manage the semantics involved. At the same time, the larger and more complex a database is, the more crucial it is for the database designers and managers to understand it. Clustering of the entities and relationships into meaningful sets is crucial in designing a conceptually sound database schema that can promote this understanding. Properly defined clusters usually represent subsystems of the organization for which the database is being built. Proper clustering allows management to prioritize modular development and implementation of large and complex databases, and it facilitates structured systems development on the part of the technical staff.

Although clustering of entities and relationships is an important step in a sound database design, our literature review shows that its importance has not been adequately recognized by database researchers, and that most of the available approaches to solve this problem are primitive, often requiring manual manipulation of the rows and columns of a matrix (Feldman et al., 1986; Teorey, et al., 1989).

Ironically, clustering problems are encountered in many disciplines including botany, linguistics, and psycho-metrics, among others (Tschudi, 1988). In manufacturing management, machines processing a group of related components need to be clustered in a physical layout, and a number of fairly sophisticated methods have been developed to solve this problem (King, 1980; Wei & Gaither, 1990). In these and related fields, an extensive amount of work is done in the area of statistical clustering (Anderberg, 1973; Jackson, 1983; Kusiak, Vannelli & Kumar, 1986). Unfortunately, some of these models seem to be computationally too complex (at times, requiring the use of super 3 computers for their solutions), to be of practical use in a modern flexible factory which must re-cluster its parts and machines in new work cells every few days as new products are introduced and demands shift. Furthermore, as Crockett et al. (1989) point out, the problem associated with their "canonical analysis" is the interpretability of the subsystems that are grouped together. The procedure maximizes the correlations between sets; however, it does not provide a facility for interpreting the resulting dimensions of sybsystems arranged by this correlation. Furthermore, the method requires that the matrices be defined as completely as possible before using the methodology.

In this paper, we present a logical algorithm which has been implemented on a PC. Thus, our algorithm avoids manual manipulation of rows and columns, but does not require a super computer either, and can be rerun without serious difficulties in case of a change in the original matrix. Given a binary entity relationship (or part-machines) matrix, our algorithm rearranges the rows and columns in such a way as to make a visual identification of the clusters very easy. The ultimate clusters can be finalized using a variety of criteria discussed by King (1980), Martin (1982), Wei and Gaither (1990), and others.

The paper is organized as follows: Section 2 provides a background on the existing literature on entity-relationship clustering; Section 3 explains the clustering problem; Section 4 describes the algorithm; Section 5 compares our results with King's (1980) solution. Finally, in Section 6, we conclude the paper with a discussion of directions for further research.

BACKGROUND

Identification of information subsystems or "subject databases" is an important task in data administration. A subject database is a group of logically related entities and relationships in an organization (Martin, 1982). The concept of clustering entities and relationships is mentioned in almost every database design text today. However, most of these tests fail to emphasize the importance of proper clustering and to present any concrete algorithms for such clustering.

The problem of clustering the entities in an ER model has been addressed by Martin (1982), Feldman et al. (1986), and Teorey et al. (1989), among others. These approaches are practical but subjective insofar as user interviews provide the basis for the manual clustering of entities and relationships. While their main advantage is that these approaches can deal with very large databases (e.g., involving 1000 entities and 5000 data elements, according to Teorey et al. [1989]), and provide layers of abstractions (i.e., groupings) supporting a variety of user views, user participation and manual clustering are two major drawbacks of these methods. Martin (1982) suggests a three-step approach, similar to IBM's Business System Planning methodology (IBM, 1981), which relies on manual manipulation of the rows and columns into clusters. Although the idea is useful, the method is not well defined. As a result, the approach is open to multiple answers depending on the interpretation of the individual developing these subsystems (Crockett, Slinkman, & Eakin, 1989). Given today's technology, we believe that any method that requires manual manipulation of the rows and columns (at least in problems involving a 30 x 30 or smaller matrix) must be seen as primitive.

Our review further indicates that database grouping and clustering has been recently incorporated in a few CASE tools. IEF (Information Engineering Facility) from Texas Instruments and IEW (Information Engineering Workbench) from KnowledgeWare Inc. are among them. IEW uses "affinity analysis" for clustering. However, as Crockett et al. (1989) have suggested, "affinity analysis is very rough and does not provide hard guidelines for the delineation of borderline cases into separate groupings." In any case, these methods are proprietary and their logic, effectiveness, and efficiency are not fully shared in the academic literature.

In short, our literature review suggests that in the database design literature the importance of good clustering schemes is down-played and efficient clustering algorithms are lacking. As we have indicated in the introduction section, in other disciplines a number of sophisticated but impractical algorithms are available. We would have liked to present a review of that literature here; however, given limited space, we must forego that. Let us, then, turn to a clearer explanation of the clustering problem and our algorithm.

THE CLUSTERING PROBLEM

We demonstrate the usefulness of clustering with s small example. Consider a business environment described below:

> A customer places an order with the sales department. A copy of the order is sent to the inventory department and another copy is sent to the accounts

receivable department. The accounts receivable department prepares the invoice and sends a copy to the customer and a copy to the sales department. In addition, the accounts receivable department periodically sends billing statements to the customer. Once the order is sent to the inventory department, a copy of the order is sent to the shipping department. If the order is available, the shipping department will ship the order. A copy of the packing list is sent to the inventory department and a second copy is sent to the purchasing department to replenish the inventory. If the order is unavailable, the inventory department will send a backorder to the shipping department and another copy is sent to the purchasing department to fill the order in the future upon inventory replenishment. The purchasing department places an order with a vendor. A copy of the purchase order is sent to accounts payable which is used to prepare checks for the vendor.

This description (along with the appropriate assumptions about business practices) will generate an entity-relationship (ER) diagram which is presented in Figure 1. The conventions used in drawing an entity-relationship diagram are that entities are represented by rectangles and relationships by diamond-shaped boxes. For example, the customer (entity A) places an order with the sales department (entity D). The associaiton between the two participating entities (A and D) creates order entry (relationship 1). For a more complete understanding of the principles of drawing an ER diagram see Chen (1976).

Figure 1. An Entity-Relationship Diagram

	Purchasing Department (H)	Order Preparation (3)	Vendor (E)
Shipping Department (B)	Replenishment and Backorder (5)	Accounts Payable (C)	Check Preparation (6)
Shipping Preparation (2)	Inventory Department (F)		
Customer (A)	Order Entry (1)	Sales Department (D)	
Billing (7)	Accounts Receivable Department (G)	Invoice Preparation (4)	

Using the ER diagram in Figure 1, we develop a binary entity-relationship matrix for this problem which is shown in Figure 2. Note that this is an 8 x 7 matrix in which a "1" in the cell (i,j) signifies that entity i is interacting with relationship j. For example, a "1" at entry (A,1) indicates that a customer (entity A) participates in order entry (relationship 1), whereas, a "0" at entry (A,3) indicates that a customer (entity A) does not participate in order preparation (relationship 3).

Figure 2. Initial Binary Entity-Relationship Matrix Relationship

	1	2	3	4	5	6	7
Entity							
A	1	1	0	1	0	0	1
B	0	1	0	0	1	0	0
C	0	0	1	0	0	1	0
D	1	0	0	1	0	0	0
E	0	0	1	0	0	1	0
F	1	1	0	0	1	0	0
G	1	0	0	1	0	0	1
H	0	1	1	0	1	0	0

As can be seen, Figure 2 is not much help in identifying subsystems or subject databases as there are no apparent cohesive groups or clusters. By rearranging the rows and columns of Figure 2 one can arrive at Figure 3, where the entity-relationship matrix is clustered.

Figure 3. Clustered Binary Entity-Relationship Matrix Relationship

	6	3	5	2	1	4	7
Entity							
E	1	1	0	0	0	0	0
Cluster-X							
C	1	1	0	0	0	0	0
H	0	1	1	1	0	0	0
Cluster-Y							
B	0	0	1	1	0	0	0
F	0	0	1	1	1	0	0
A	0	0	0	1	1	1	1
G	0	0	0	0	1	1	1
Cluster-Z							
D	0	0	0	0	1	1	0

As can be seen, there are three clusters X, Y, and Z. A closer look at Figure 3 illustrates that the cluster X is the accounts payable subsystem, cluster Y is the inventory subsystem, and cluster Z is the accounts receivable subsystem. The three 1's at A2, F1, and H3 are the linkage points. In other words, A2 and F1 are the linkages between clusters Y and Z, or the inventory and accounts receivable subsystems. The linkage points show where two or more subsystems interact with one another. Furthermore, H3 is the linkage point between clusters X and Y, or the accounts payable subsystem and the inventory subsystem. Once these subsystems are identified, management can use this identification for the modular development and implementation of each subsystem and for project management. Subsystem development could also be prioritized according to the immediate needs of the organization. Subsystems critical to the organization could be implemented first and less critical subsystems could be delayed. Finally, subsystem identificaiton facilitates the organization and control of resources during various phases of systems development.

This example shows how clustering could be beneficial in identifying subject database grouping for development and implementation purposes. The role of a clustering algorithm is precisely to take a matrix such as the one presented in Figure 2 and by rearranging its rows and columns, come up with a matrix such as the one presented in Figure 3. Of course, in very large databases, involving thousands of entities and relationships, there is a role for the kinds of higher level manual groupings generated by approaches such as those due to Feldman et al. (1986) and Teorey, et al. (1989). However, at a given level of abstraction with a reasonably sized (e.g., a 30 x 30) matrix, we believe one needs a well defined computerized algorithm to identify the type of pattern presented in Figure 3. As we shall show, this is precisely what our algorithm can do.

THE CLUSTERING ALGORITHM

Let m be the number of entities (rows) that could have potential interactions with n relationships (columns). First, we construct an mxn entity-relationship interaction matrix where

xij, the element in the i th row and j th column is such that

xij = 1 if the i th entity has an interaction with the j th relationship, and

xij = 0 if the i th entity has no interaction with the j th relationship. (1)

Let M0 represent the original matrix with the rows and columns in any arbitrary order. In clustering these rows and columns, our objective is to rearrange the rows and columns such that columns which interact with the same rows are close to one another and rows which interact with the same columns are also close to one another. Then, one can visually identify clusters of rows and columns with greatest interactions with one another and very few, if any, interactions outside a given cluster. In creating this rearrangement, we shall first construct a matrix M1 where we leave the rows in the same order as in the matrix M0, but rearrange the columns. Next we shall construct a matrix Mz where the columns are left in the same order as in M1 but the rows are rearranged. This final matrix Mz is then used to visually identify the desired number of clusters and the contents of each. The procedure for creating the matrix M1 is as below:

Let djk (= dkj), the "distance" between the two relationships (columns) j and k, be given by the formula:

$$djk = 4m - * (xij \text{ I } xik) + 5 * (xij - xik) \quad (2)$$

Since in M0, xij and xik each can be only 0 or 1, this formula leads to the smallest distance (djk = 0), when all m entries in each of the columns j and k are 1. When all m entries in each of the columns j and k are 0, the distance is 4m, and when the m entries in column j are never identical with the m entries in column k, i.e., a 1 in one column is accompanied by a 0 in the other column, the distance is the largest (djk = 8m). Note that any linear transform of formula (2) will be fine since the algorithm depends on the relative distances rather than their absolute values. In any case, with distance defined this way, we are now ready to juxtapose the various columns so that a pair with the least distance from each other is next to each other in the matrix M1.

To begin with our algorithm, we first calculate the distance between each one of the possible pairs of columns in the Matrix M0. We compare these distances with one another, and choose the pair with the least distance. If there is a tie for the least distance aM0ng several pairs, we choose one pair arbitrarily. Suppose columns g and h represent our chosen pair.

Then in matrix M1 we shall juxtapose these two columns next to each other, and temporarily designate g as the "leftM0st of the used columns" (L, for short), and h as the "rightM0st of the used columns" (R, for short). We also flag columns g and h in matrix M0 as the "used" columns, all other columns being considered as "unused."

Now, the following procedure (LOOP) will be implemented until there are no unused columns left.

Let u represent an unused column, duL and duR being its distances from the current leftM0st and the current rightM0st of the used columns. Assume that column a has the smallest of all duL values (as always, in case of a tie for the smallest duL, we choose one of the tied columns arbitrarily) and column b has the smallest of all duR values. Note that column a and column b need not be two distinct columns; in fact, when only one unused column is remaining, a and b will indeed be identical.

If a and b are identical, and daL = dbR, put down this column to the right of the current R in M1, designate it as the new R in Ml, mark it as a used column in M0, and go back to the LOOP.

Otherwise (i.e,, when a and b are not identical, or daL and dbR are not equal), implement the following procedure.

If daL < dbR, then put down column a to the left of the current L in M1, designate a as the new L in M1, flag a as a used column in M0, and go back to the LOOP.

If daL > dbR, then put down column b to the right of the current R in M1, designate b as the new R in M1, flag b as a used column in M0, and go back to the LOOP.

This is the end of the LOOP.

By now we have constructed the matrix M1. As one final step, check dLR, the final distance between the leftmost and the rightmost of the matrix M1. If this distance is too small (e.g., 0 to B), we may have to consider the "wrap-around" problem in visually identifying the clusters. In constructing M2, the columns in M1 are left in the same order but the rows are rearranged following an exactly similar procedure. In fact, instead of rewriting the entire algorithm, we implemented it simply by first transposing all rows and columns in M1, implementing our algorithm for the rearrangement of columns, and retransposing the columns and rows. As indicated before, once the M2 matrix is created, the rest of the clustering is to be done visually. Of course, one can use any one of the several criteria proposed by Martin (1982), King (1980), Wei and Gaither (1990), or others in finalizing these clusters.

A Comparison of the Results of our algorithm with King's (1980) "Rank Order Clustering" Method

As we have suggested before, the state of the art in clustering databases seems primitive. As such, there are not many test cases in this body of the literature. On the other hand, in manufacturing management, there are many sophisticated techniques proposed for clustering parts-machines groups. Therefore, we try to establish the credibility of our algorithm by using a problem in machine-part clustering, first described by King (1980). This happens to also be one of the problems Wei and Gaither (1990) used in establishing the effectiveness of their algorithm. The problem tested by King (1980) is a 24 parts (rows), 14 machines (columns) problem. The initial matrix is given in Figure IV.

Figure 4. King's Problem - The Initial Binary Matrix

	Machines													
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
Parts														
1	0	0	0	1	1	0	1	0	0	0	0	0	0	0
2	0	0	0	1	1	0	1	0	0	0	0	0	0	0
3	0	1	1	0	0	0	0	0	0	1	1	0	0	0
4	0	1	1	0	0	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	0	0	1	1	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0	0	1	0
7	1	0	0	0	0	0	1	0	0	0	0	1	1	0
8	0	0	0	0	0	0	0	0	0	0	0	1	1	0
9	0	0	0	0	0	1	0	1	1	0	0	0	0	1
10	0	0	0	0	0	1	0	1	0	0	0	0	0	0
11	0	0	0	0	0	1	0	0	0	0	0	0	0	1
12	0	0	0	0	0	1	0	1	1	0	0	0	0	0
13	0	0	0	0	0	0	0	0	1	0	0	0	0	1
14	0	0	0	0	0	1	0	1	0	0	0	0	0	0
15	0	0	0	0	0	1	0	1	1	0	0	0	0	1
16	0	0	0	0	0	1	0	1	0	0	0	0	0	0
17	0	0	0	1	1	0	1	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	1	0
19	0	0	0	1	0	0	0	0	0	0	0	0	0	0
20	0	0	0	1	1	0	1	0	0	0	0	0	0	0
21	0	0	1	0	0	0	0	0	0	0	1	0	0	0
22	0	0	0	0	0	1	0	1	0	0	0	0	0	0
23	0	0	0	1	1	0	0	0	0	0	0	0	1	0
24	0	0	0	0	0	0	0	0	0	1	1	0	0	0

In identifying the clusters in this matrix, King's algorithm is to read each row as a binary number and rearrange the rows in their decreasing numerical order. Next, King's algorithm reads each column as a binary number and rearranges all columns in their decreasing numerical order. However, this rearrangement of the columns may change the binary words in the rows and hence the rows must be rearranged again, which in turn may change the binary words in the columns. King's algorithm requires that such a rearrangement of columns and rows be repeated until there are no changes from one iteration to the next. Using his algorithm, King (1980) arrives at his initial solution as the pattern in Figure V which shows four clusters where clusters I and II are not mutually independent.

Figure 5. King's Initial Solution

	Machines													
	G	D	E	M	A	L	K	C	B	J	H	I	F	N
Parts														
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
CLUSTER-1														
17	1	1	1	0	0	0	0	0	0	0	0	0	0	0
20	1	1	1	0	0	0	0	0	0	0	0	0	0	0
7	1	0	0	1	1	1	0	0	0	0	0	0	0	0
23	0	1	1	1	0	0	0	0	0	0	0	0	0	0
19	0	1	0	0	0	0	0	0	0	0	0	0	0	0
CLUSTER-2														
6	0	0	0	1	1	0	0	0	0	0	0	0	0	0
8	0	0	0	1	0	1	0	0	0	0	0	0	0	0
18	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	1	1	1	0	0	0	0
CLUSTER-3														
21	0	0	0	0	0	0	1	1	0	0	0	0	0	0
24	0	0	0	0	0	0	1	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1														
15	0	0	0	0	0	0	0	0	0	0	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	1	1	1	0
5	0	0	0	0	0	0	0	0	0	0	1	1	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	1	0
CLUSTER-4														
16	0	0	0	0	0	0	0	0	0	0	1	0	1	0
22	0	0	0	0	0	0	0	0	0	0	1	0	1	0
13	0	0	0	0	0	0	0	0	0	0	0	1	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Using his initial solution, King (1980) notes that the reason clusters I and II are not independent is because components 7 and 23 appear in both. Therefore, King (1980) manually "suppresses" the exceptional elements (7,G) and (23,M), i.e., removes the "1" from these entries, and reruns his algorithm. Then he puts back the suppressed elements and presents his final solution as in Figure 6. As can be noted, this solution comprises of four mutually independent clusters with the exceptional elements as encircled entries.

Figure 6. King's Final Solution

Machines

Parts	D	E	G	M	A	L	K	C	B	J	H	I	F	N
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
17	1	1	1	0	0	0	0	0	0	0	0	0	0	0
CLUSTER-1														
20	1	1	1	0	0	0	0	0	0	0	0	0	0	0
23	1	1	0	1	0	0	0	0	0	0	0	0	0	0
19	1	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	1	1	0	0	0	0	0	0	0	0	0
CLUSTER-2														
8	0	0	0	1	0	1	0	0	0	0	0	0	0	0
18	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	1	1	1	0	0	0	0
4	0	0	0	0	0	0	1	1	1	0	0	0	0	0
CLUSTER-3														
21	0	0	0	0	0	0	1	1	0	0	0	0	0	0
24	0	0	0	0	0	0	1	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	1	1	1	1
12	0	0	0	0	0	0	0	0	0	0	1	1	1	0
5	0	0	0	0	0	0	0	0	0	0	1	1	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	1	0
CLUSTER-4														
14	0	0	0	0	0	0	0	0	0	0	1	0	1	0
16	0	0	0	0	0	0	0	0	0	0	1	0	1	0
22	0	0	0	0	0	0	0	0	0	0	1	0	1	0
13	0	0	0	0	0	0	0	0	0	0	0	1	0	1
11	0	0	0	0	0	0	0	0	0	0	0	0	1	1

When we applied our algorithm to the same problem, we obtained the pattern in Figure 7.

Figure 7. The Solution from Our Algorithm

	Machines													
	F	H	I	N	A	L	M	B	C	K	J	G	E	D
Parts														
24	0	0	0	0	0	0	0	0	0	1	1	0	0	0
3	0	0	0	0	0	0	0	1	1	1	1	0	0	0
CLUSTER-3														
4	0	0	0	0	0	0	0	1	1	1	0	0	0	0
21	0	0	0	0	0	0	0	0	1	1	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	1
20	0	0	0	0	0	0	0	0	0	0	0	1	1	1
17	0	0	0	0	0	0	0	0	0	0	0	1	1	1
CLUSTER-1														
2	0	0	0	0	0	0	0	0	0	1	1	1		
10	0	0	0	0	0	0	0	0	0	0	0	1	1	1
23	0	0	0	0	0	0	1	0	0	0	0	0	1	1
8	0	0	0	0	0	1	1	0	0	0	0	0	0	0
7	0	0	0	0	1	1	1	0	0	0	0	0	0	0
CLUSTER-2														
6	0	0	0	0	0	1	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	1	0	0	0	0	0	0	0
13	0	0	1	1	0	0	0	0	0	0	0	0	0	0
5	0	1	1	0	0	0	0	0	0	0	0	0	0	0
12	1	1	1	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	0	0	0	0	0	0	0	0	0	0
15	1	1	1	1	0	0	0	0	0	0	0	0	0	0
CLUSTER-4														
10		1	1	0	0	0	0	0	0	0	0	0	0	0
14		1	1	0	0	0	0	0	0	0	0	0	0	0
16		1	1	0	0	0	0	0	0	0	0	0	0	0
22		1	1	0	0	0	0	0	0	0	0	0	0	0
11		1	0	0	1	0	0	0	0	0	0	0	0	0

A visual inspection of our pattern shows that the solution of our algorithm is identical to the final solution derived by King (1980). Note that unlike King's method our solution does not require manual intervention to first identify the exceptional elements, suprress them, and rerun the algorithm. In short, our algorithm is clearly superior to King's algorithm.

Of course, Wei and Gaither (1980) have presented an optimal 0-l integer programming model that is also capable of obtaining King's (1980) final solution, provided that one can define in advance a number of factors including:

i. an appropriate optimizing function (perhaps in terms of opportunity cost of producing an exceptional part outside the work-cells)

ii. machine capacity constraints

iii. number of clusters desired

iv. maximum number of machines allowed per cluster, etc.

Insofar as these specificaitons may be difficult, and their "optimality" impossible to prove, Wei and Gaither's "optimal" model really requires trial and error in the specifications stage. Furthermore, Wei and Gaither's (1990) model requires the use of a super computer. This is precisely why their model may be best used in conjunction with a heuristic model. We believe that we have presented a heuristic algorithm that is better than those available in the literature. Once the most clustered arrangement is produced by our algorithm, we can use considerations such as desired number of clusters, or maximum number of machines per cluster to arrive at the final designations of the clusters and the exceptions. Thus, we believe that our approach is more practical than the Wei and Gaither (1990) approach.

Conclusion and Directions for Further Work

In this paper we presented a logical but simple algorithm for clustering entities and relationships into meaningful information subsystems. Our algorithm avoids manual manipulation of rows and columns, but does not require a super computer either, and can be rerun without serious difficulties in case of a change in the original matrix. As we have argued, it is a very practical approach.

Given a binary entity-relationship (or part-machines) matrix of a reasonable size, our algorithm rearranges the rows and columns in such a way as to make a visual identification of the clusters very easy. The ultimate clusters can be finalized using a variety of criteria discussed by King (1980), Martin (1982), Wei and Gaither (1990), and others. Clearly, we need to do further work on how to systematically integrate these criteria once our algorithm has produced the most clustered pattern.

In this paper, we tested our algorithm using only one of the problems in the available literature. In a follow-up work, we intend to test our algorithm by comparing it against a variety of other clustering methods using the problems presented by the proponents of those methods. Our algorithm can be useful in many disciplines that encounter similar clustering problems. One direction for further work is to seek such applications in other disciplines. As we have noted, our algorithm may be particularly useful in quickly redesigning manufacturing work-cells as new products are introduced and the demands for older products change.

Finally, in database design, clustering based on the binary entity-relationship matrices has one major limitation. The data about an entity are either created, retrieved, updated, or deleted (CRUD) as a result of participation in one or more relationships. Furthermore, as Teorey et al. (1989) indicate, in some cases, the original ER model needs to be extended to accommodate

conditional and unconditional membership in the relationships. A binary matrix cannot represent these variety of the types of interactions. We are currently exploring methods to generate clustered CRUD entity-relationship matrices.

NOTES

1. In part-machine clustering problems, these linkages are called "exception entries" since they identify parts and processes that may have to be manufactured outside the clustered work-cells.
2. A careful examination reveals that the matrix reported by Wei and Gaither (1990) is not identical to the matrix in King (1980). We do not know whether this discrepancy is the result of a typographical error or whether Wei and Gaither (1990) actually solved a different problem.

REFERENCES

Anderberg, M. R. (1973). *Cluster analysis for applications*. Academic Press.

Chen, P. P. (1976). The entity-relationship model - towards a unified view of data. *ACM Transactions on Database Systems, 1*(1), 9-36.

Crockett, H. D., Slinkman, C. W. & Eakin, M. F. A strategy for database planning. *Proceedings, 1989 Annual Meeting of Decision Sciences Institute*, 656-658.

Date, C. J. (1990). *An introductin to database systems*. Addison Wesley Publishing.

Feldman, P. & Miller, D. (1986). Entity model clustering: Structuring a data model by abstraction. *Computer Journal, 29*(4), 348-360.

IBM Corp. (1981). *Business systems planning: Information systems planning guide*, 3rd Ed., Publication GE20-0527-3, IBM Corporation, Technical Publications.

Jackson, B. B. (1983). *Cluster analysis, multivariate data analysis* (chapter 8), Irwin Publishing.

Kink, J. (1980). Machine-component group formation in group technology. *OMEGA, 8*(2), 193-199.

Kusiak, A., Vannelli, A. & Kumar, K. R. (1986). Clustering analysis: Models and algorithms. *Control and Cybernetics, 15*(2), 139-154.

Martin, J. (1982). *Strategies for data-planning methodologies*. Prentice Hall Publishing.

Teorey, T. J., Wei, G., Bolton, D. L. & Koenig, J. A. (1989). ER model clustering as an aid for user communication and documentation in database design. *Communicaitons of the ACM, 32*(8), 975-987.

Tschudi, F. (1988). *Matrix representation of expert systems, AI expert*, 975-987.

Wei, J. C. & Gaither, N. (1990). An optimal model for cell formulation decisions. *Decision Sciences, 21*(2), 416-433.