Available online at www.sciencedirect.com
ScienceDirect
Information Systems 32 (2007) 773–792
www.elsevier.com/locate/infosys

An automated entity–relationship clustering algorithm for conceptual database design

Madjid Tavana[a,*], Prafulla Joglekar[a], Michael A. Redmond[b]

[a]*Management Department, La Salle University, Philadelphia, PA 19141, USA*
[b]*Department of Mathematics and Computer Science, La Salle University, Philadelphia, PA 19141, USA*

Received 31 December 2005; received in revised form 6 July 2006; accepted 11 July 2006
Recommended by F. Carino Jr.

Abstract

Entity–relationship (ER) modeling is a widely accepted technique for conceptual database design. However, the complexities inherent in large ER diagrams have restricted the effectiveness of their use in practice. It is often difficult for end-users, or even for well-trained database engineers and designers, to fully understand and properly manage large ER diagrams. Hence, to improve their understandability and manageability, large ER diagrams need to be decomposed into smaller modules by clustering closely related entities and relationships. Previous researchers have proposed many manual and semi-automatic approaches for such clustering. However, most of them call for intuitive and subjective judgment from "experts" at various stages of their implementation. We present a fully automated algorithm that eliminates the need for subjective human judgment. In addition to improving their understandability and manageability, an automated algorithm facilitates the re-clustering of ER diagrams as they undergo many changes during their design, development, and maintenance phases.

The validation methodology used in this study considers a set of both objective and subjective criteria for comparison. We adopted several concepts and metrics from machine-part clustering in cellular manufacturing (CM) while exploiting some of the characteristics of ER diagrams that are different from typical CM situations. Our algorithm uses well established criteria for good ER clustering solutions. These criteria were also validated by a group of expert database engineers and designers at NASA. An objective assessment of sample problems shows that our algorithm produces solutions with a higher degree of modularity and better goodness of fit compared with solutions produced by two commonly used alternative algorithms. A subjective assessment of sample problems by our expert database engineers and designers also found our solutions preferable to those produced by the two alternative algorithms.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Computers; Databases; Information systems; Analysis and designs; Planning; Decision analysis

1. Introduction

Entity–relationship (ER) modeling [1] is a popular and effective methodology used to construct a *conceptual* data model. An ER diagram (ERD) is a detailed graphical representation of the data

*Corresponding author. Tel.: +1 215 951 1129; fax: +1 267 295 2854.
E-mail addresses: tavana@lasalle.edu (M. Tavana), joglekar@lasalle.edu (P. Joglekar), redmond@lasalle.edu (M.A. Redmond).
URL: http://lasalle.edu/~tavana.

0306-4379/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2006.07.001

requirements in an organization or business unit. ERDs enhance understanding of the system and improve communication among database engineers, designers, and end-users.

Consider the following business scenario documented during systems analysis for the development of an information system for a retailer of custom-made products:

> A **customer** issues a **purchase order** to the **vendor** (retailer) to buy a **product**. The purchase order consists of an **order item**. When the **product** is delivered, the **customer** is given a **receipt** for the **product**. Because the **customer** has a **line of credit**, the actual payment is deferred until later. In the next billing cycle, the **vendor** issues an **invoice** that includes an **invoice item** related to the **order item** on the **purchase order**. Upon receiving the **invoice**, the **customer** makes **sales payment** against the **receipt** of the **product** and the **vendor** receives **vendor payment** for the **invoice**.

This scenario results in the identification of the entities and relationships presented in a clustered ERD presented in Fig. 1a. Entities are represented by rectangles, relationships by diamond-shaped boxes, and connecting lines show which entities participate in which relationship [1]. For example, the fact that a customer (Entity A) buys a product (Entity D) is represented by the "buy" relationship (Relationship 1). Knowledgeable end-users can comprehend and validate the database design implied by an ERD against actual business practices. Once finalized, an ERD serves as the blueprint for database implementation.

The ER model has become so common in database design that today a number of commercial ER diagraming tools are available (e.g. ERWin by Computer Associates, EasyER by Visible Systems, Visio by Microsoft, and ER/1 by Embarcadero). ER tools differ somewhat in their terminology and notation. However, the basic concepts are the same and all ER tools represent data requirements graphically. Several tools generate the code needed for the database schema, including the necessary tables, indexes, triggers, and stored procedures. Most ER tools support systems analysis and database design, implementation, and maintenance.

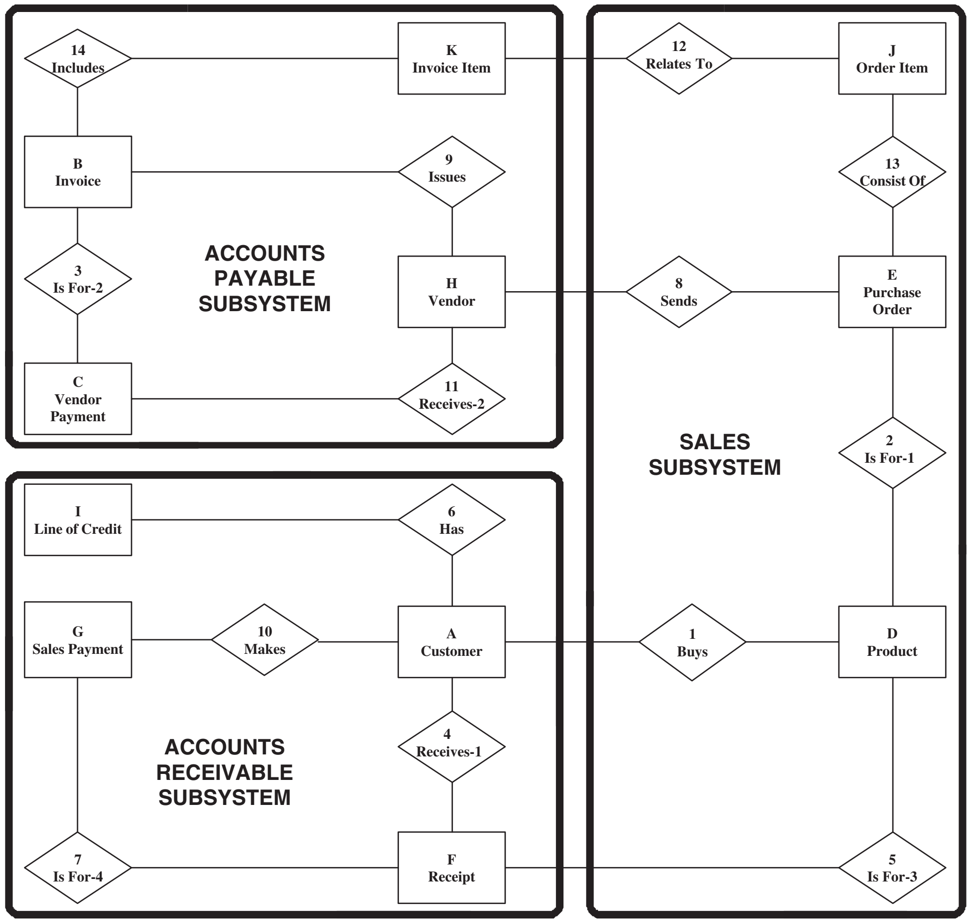
Yet, today ER tools fall short of their true potential. This is because ER diagrams are rarely as small as the one presented in Fig. 1a. A typical application data model consists of 95 entities and an average enterprise model consists of 536 entities [2]. Feldman and Miller [3] suggest that diagrams involving 30 or more entities exceed the limits of easy comprehension, and communications. To improve their understandability and manageability, large ER diagrams need to be decomposed into smaller modules by clustering closely related entities and relationships.

The ERD in Fig. 1a is small enough to comprehend without any decomposition. However, the three clusters (retailer's "sales," "accounts receivable," and "accounts payable" subsystems) identified by our algorithm illustrate some of the advantages of decomposing ERDs. The literature on clustering [3–6] has identified several advantages of ERD decomposition. Clustered ER diagrams:

1. Are easier to develop and maintain, since they are more modular;
2. Are easier to document, since clusters are smaller and easier to understand;
3. Are easier to validate, since they provide better organization;
4. Can assist project management by allowing allocation of modular tasks to individuals or teams; and
5. Can assist identification of reusable subsystems that can be added, removed, or modified relatively independently of one another.

In spite of these advantages, today no ER diagramming tool offers any clustering assistance. This is mainly because the available ER clustering algorithms call for intuitive and subjective judgment from "experts" at various stages of their implementation (see Section 2). ER tools support data model construction, communication, and validation by storing all the entities, relationships, relevant assumptions, and constraints in a repository. Using the repository, multiple conceptual and physical-level ERD "views" can be produced for specific purposes such as end-user communication, database design, and development by presenting only relevant portions of the larger design to specific audiences. Users often prefer such views. As such, we do not want to suggest that clustered diagrams would replace functional views of a database.

However, sometimes even these views are too large for adequate comprehension. More importantly, because specific entities and relationships are often duplicated in several views, the "views" do not offer some of the advantages of a clustered ERD. For example, in allocating responsibility for the



(a)

ENTITY	RELATIONSHIP													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	1	0	0	1	0	1	0	0	0	1	0	0	0	0
B	0	0	1	0	0	0	0	0	1	0	0	0	0	1
C	0	0	1	0	0	0	0	0	0	0	1	0	0	0
D	1	1	0	0	1	0	0	0	0	0	0	0	0	0
E	0	1	0	0	0	0	0	1	0	0	0	0	1	0
F	0	0	0	1	1	0	1	0	0	0	0	0	0	0
G	0	0	0	0	0	0	1	0	0	1	0	0	0	0
H	0	0	0	0	0	0	0	1	1	0	1	0	0	0
I	0	0	0	0	0	1	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	1	1	0
K	0	0	0	0	0	0	0	0	0	0	0	1	0	1

(b)

Weak/Subtype Entity	Strong/Supertype Entity
J	E
K	B

(c)

Fig. 1. Introductory example: (a) clustered ER diagram for introductory example; (b) M_0 for introductory example; (c) T_0 for introductory example.

development and maintenance of specific modules, we must have non-overlapping clusters. Thus, incorporation of a clustering methodology would empower existing ERD tools.

An automatic algorithm would overcome three important shortcomings of the existing algorithms requiring manual intervention. First, the clusters produced by a manual algorithm vary from one implementation to the next, depending on the intervening human expert's assumptions, semantics, intuition, and judgment. Even ERDs clustered by the same individual can lack consistency from one attempt to the next. In repeated trials, an automatic algorithm will produce the same solution for a given problem and consistent solutions to similar problems.

Second, manual algorithms may work fine for small ERDs. However, for typical real-life (and large) ERDs, they are not likely to produce a good solution that seeks several clustering objectives simultaneously. As Moody and Flitman [7] point out, "Because of the enormous number of decompositions that are possible in even small data models, it is clearly beyond human cognitive capabilities to find an optimal solution. According to the principle of bounded rationality, humans will only explore a limited range of alternatives and consider a subset of the decomposition objectives in order to make the task cognitively manageable ..."

Third, manual algorithms take considerable amounts of time to implement. In today's highly dynamic business environment with numerous mergers and acquisitions, existing business models become obsolete at an increasing pace. The ability to rapidly change databases and their underlying data models to support the needs of changing business models is high on the research agenda of information systems researchers [8].

In spite of their limitations, manual methods might have produced satisfactory decompositions of traditional transactional processing systems involving static and localized databases. However, new applications (from E-commerce to web-based decision support systems, and from multimedia to geographic information systems) demand integrated databases with more entities, relationships, attributes, data elements, etc. [9,10]. The added semantics and complexity require that database engineers and designers revisit their conceptual data models periodically in an attempt to expand or modify them [11,12]. Furthermore, in software engineering, emphasis has shifted from a rigid system design, where the software is available at the end of the process, to an incremental and modular design based on iterative clustering refinements [10]. Clearly, an automated clustering algorithm can make iterative refinements considerably simpler and faster. According to Francalanci and Pernici [5], in legacy database re-engineering, automated clustering is particularly useful due to the size of the existing physical schema.

Hence, it is no surprise that recent work in ER clustering has aimed at automated algorithms [4,5,13,14]. Ideally, given the information obtained by database engineers and designers during the construction of an ERD, an automated algorithm should identify suitable entity and relationship clusters without further human intervention. Unfortunately, none of the above referenced works meets that ideal. In this paper, we present a fully automated ERD decomposition algorithm that meets this ideal.

Clustering problems arise in numerous domains ranging from weather forecasting, to epidemiology, cellular manufacturing (CM), linguistics, data mining, and economics. In different contexts, clustering takes different names such as typology, numerical taxonomy, and partitioning. Clustering methodologies originate in equally diverse disciplines including neural networks, genetics, fuzzy sets, matrix manipulation, mathematical programming, and multivariate analysis [15].

Traditionally, ERD clustering algorithms have focused on the hierarchical levels of entities. Entity types are indeed an important consideration in ERD clustering and our algorithm appropriately accommodates different entity types. However, Moody and Flitman's [16] recent experiment shows that "connectivity" (or the number of relationships an entity participates in) is the more appropriate criterion for ERD clustering than the hierarchical levelling criterion. In CM, "connectivity," or the number of machines (similar to relationships) processing a part (similar to entities) and the number of parts processed by a machine, has been the primary criterion for clustering. Hence, we have adopted some of the CM concepts in designing our algorithm.

Furthermore, available ER clustering algorithms (with the exception of [17]) have focused on clustering only the entities. However, as Francalanci and Pernici [5] have noted, relationships play an important role in schema validation, schema reuse, and schema integration. We believe that the inclusion of relationships in ER clustering provides a more precise resource allocation plan for the development and maintenance of specific modules. When a "boundary" relationship involves entities in two different modules, the responsibility for coding and maintenance of that relationship must be assigned to one, and only one, of the two individuals (or teams) in charge of those two modules. Therefore, our algorithm clusters not only the entities but also the relationships. In CM, a manufacturing cell is formed only when one identifies a group of machines as well as a "part family" that "the machine group" processes.

This paper is organized as follows. Section 2 presents a review of prior work on ER clustering. In Section 3, we summarize the criteria for good ER clustering solutions. In Section 4, we outline the relevant concepts and metrics from CM that are used in our algorithm. In Section 5, we explain the logic and details of our algorithm. Section 6 discusses how the complexity of our algorithm is kept manageable and how it compares with a brute force approach to clustering. In Section 7, we assess the effectiveness of our algorithm. In the available literature, we found only two examples of large enough ER diagrams decomposed with alternative algorithms. Hence, we compare our algorithm's solutions to those two problems with the solutions by those algorithms. We show that our algorithm produces better solutions than those obtained by other ERD clustering algorithms based on both, the criteria established in Section 3 and the preferences of a group of expert database engineers and designers at NASA. Our conclusions are summarized and directions for future research are outlined in Section 8.

2. Prior work on ER clustering

In his earliest work on ER clustering, Martin [18] used "hierarchical leveling" to organize an ERD into a set of hierarchies, each defined by a chain of one-to-many (parent–child) relationships. A "root" entity with no many-to-one relationships was placed at the top of each hierarchy. Clusters were formed with root entities as centers and their descendents as other elements in the cluster. The requirement that main entities must have no many-to-one relationships was too inflexible. When hierarchical chains overlapped, human judgment was used to resolve boundaries. Two-level clustering was considered an advantage of Martin's [18] approach. At the second level, identified entity clusters were clustered into *entity supergroups* based on the strength of association (frequency of use) between clusters. Feldman and Miller [3] changed Martin's [18] second level clustering by focusing on "subject areas" to group the entity clusters. Teorey et al.'s [6] method identified the subject (or functional) areas first and then clustered entities within a functional area by sequentially grouping weak entities with their dominant entities, subtype entities with their supertypes, and binary entities together, followed by tertiary entities together. All of these methods depended heavily on human judgment to resolve boundaries, to define strength of association, and/or to identify suitable subject areas.

Huffman and Zoeller [13] were the first to attempt to automate ER clustering by developing an expert system to find the dominant entities based on a set of rules. However, Campbell et al. [19] suggested that the heuristic was too simplistic for complex cases and sometimes resulted in overlooked dominant entities. Francalanci and Pernici [5] presented a semi-automatic method in which each entity started as its own cluster. Then the most closely linked pair of clusters was combined into one cluster. This continued until a predetermined number of clusters were found. Unfortunately, determining the closeness of a pair of clusters required a weighted average of human judgments on several dimensions of similarities and dissimilarities among the entities and relationships in the two clusters. According to Moody and Flitman [7], this was an enormous task and defeated the purpose of automation.

Akoka and Comyn-Wattiau's [17] algorithm comes closest to being fully automated. They define closeness of entities by considering types of entities and relationships. The distance between a weak entity and its dominant entity is 1, between a supertype entity and its subtype entity is 10, between two entities in an exclusive relationship is 100, between two entities involved in a binary relationship is 1000, and so on. This approach removes the subjectivity of measurement. Once the distances are established, Akoka and Comyn-Wattiau's [17] algorithm can automatically find the "best" solution for a prescribed number of clusters. A distinguishing feature of Akoka and Comyn-Wattiau's [17] algorithm is that it clusters relationships as well as entities while other methods focus on clustering entities only. One shortcoming of this algorithm is that it provides no guidance for the number of clusters to be found. Furthermore, as Moody and Flitman [14] point out, it is difficult to justify the weights assigned to each type of semantic distance between entities.

While all of the foregoing algorithms can be seen as extensions of Martin's [18] "hierarchical leveling" approach, Moody [20] argued that hierarchical leveling was inappropriate for ER clustering. He found no theoretical justification for it in the literature. He argued that while there was strong evidence that grouping attributes based on hierarchical leveling reduces redundancies, there was no empirical evidence that it improves understanding. Moody's [20] empirical evidence showed that "connectivity" (i.e. the number of relationships an entity participates in) was a better basis for ER clustering.

We believe hierarchical leveling based clustering might have worked for hierarchical database designs of the past. However, given the overlapping nature of hierarchical chains, and the redundant duplication of entities, relational databases have come to be the dominant approach for today's large-scale systems. While separable hierarchical chains can be useful in designing semantically meaningful clusters in relational databases, when numerous hierarchical chains overlap, and boundaries between overlapping chains must be determined, connectivity provides an objective (and hence, automable) criterion for setting those boundaries.

Moody and Flitman [14] presented both a manual method and an automatic genetic algorithm method for ER clustering based on connectivity. Genetic algorithms are patterned after biological genetics. The parameters to be specified include the "population size"—the number of individuals who compete for survival, the "crossover rate"—the likelihood of genetic combination via crossover of "chromosomes" from different individuals in the population while producing offspring, and the "mutation rate"—the likelihood of random mutations of chromosomes in the offspring. As Moody and Flitman [14] noted, "Choosing optimal parameters for the genetic algorithm optimization is not straightforward." Moody and Flitman [14] carried out repeated experiments with the program until good results were obtained. Users of a genetic algorithm approach would need to carry out this type of "tuning" of the parameters for each problem. Such tuning requires some knowledge of genetic algorithms as well as the knowledge of ER diagrams.

In short, the parameter tuning required by many of the available clustering algorithms really amounts to a trial and error method to find a solution that is judged to be the best by some expert. In contrast, the clustering algorithm presented in this paper is fully automated. Our algorithm uses only the information contained in an ER diagram and, through an automated procedure, identifies suitable entity and relationship clusters without any further human intervention and subjective judgment.

3. Criteria for evaluation of clustering solutions

Several authors have discussed the criteria for good ER clustering [3–5,13]. Moody and Flitman [7] proposed a set of nine principles and associated measurable criteria for ER clustering. We also conducted a survey of 16 expert database engineers and designers who are members of the Mission Control Center Systems Architecture Team at the Johnson Space Center. We provided a list of Moody and Flitman's [7] nine criteria (along with their brief definitions) to our expert database engineers and designers and asked them to rate the importance of each one in assessing the effectiveness of a clustering arrangement. The scale was 1 (= least important) to 5 (= most important). The results are presented in Table 1.

The expert database engineers and designers rated the "non-redundancy" criterion the highest (with an average score of 4.938) and the "centered" criterion the lowest (with an average score of 3.375). We used a two-population t-test ($\alpha = 0.05$ level of significance) to verify if there is significant difference in the mean scores of each pair of criteria. The results in Table 1 show that the mean score of the "centered" criterion is statistically different from (smaller than) the mean scores of all the other criteria. Thus, for expert database engineers and designers, "centered" is the least important criterion. Table 1 also shows that the mean score of the "non-redundancy" criterion is statistically different from (larger than) the mean scores of all the other criteria except for the "modularity" criterion. Hence, we believe practicing database engineers and designers want to see non-redundant clusters.

Of course, in an informal conversation after the survey, some of our expert database engineers and designers pointed out that had we surveyed database users, we might not have found such a high rating for non-redundancy. Users who must deal with multiple functional areas of a system often prefer to see overlapping database views. If some entities and relationships are relevant to the sales function and the accounting function, users prefer those entities to be shown in both, the sales view as well as the accounting view. However, there is no reason to assume that a "cluster" in a database is a substitute for a functional "view" of the database. We agree that a user's understanding may be better served by overlapping views, and clustered ERDs cannot be substitutes for functional views. On the other hand, non-overlapping clusters better serve a

Table 1
The expert database engineers' mean importance ratings of various criteria for good clustering: A test of significance

Criteria	Average	Non-redundancy	Completeness	Modularity	Fully connected	Semantically meaningful	Manageable size	Minimal coupling	Maximal cohesion
Non-redundancy	4.938	—	—	—	—	—	—	—	—
Completeness	4.625	Y	—	—	—	—	—	—	—
Modularity	4.625	N	N	—	—	—	—	—	—
Fully connected	4.563	Y	N	N	—	—	—	—	—
Semantically meaningful	4.375	Y	N	N	N	—	—	—	—
Manageable size	4.313	Y	Y	N	N	N	—	—	—
Minimal coupling	4.188	Y	Y	Y	Y	N	N	—	—
Maximal cohesion	4.125	Y	Y	Y	Y	N	N	N	—
Centered	3.375	Y	Y	Y	Y	Y	Y	Y	Y

$\alpha = 0.05$ level of significance.

database designer's understanding and work allocation.

We also asked our expert database engineers and designers to identify any other criteria that they considered relevant to ER clustering. No new criterion was identified. Hence, we believe that Moody and Flitman's [7] criteria set is comprehensive.

However, we find the definitions of some of the nine criteria questionable. One criterion proposed by Moody and Flitman [7] suggests that clusters should be kept to a cognitively manageable size of 5–9 entities. While, in principle, it is desirable to have a manageable size for each cluster, we see some flaws with the cognitive argument that most people can only keep up to 7 ± 2 items in their short-term memory. First, this limit is too restrictive in practice since, in an ER diagram, one has to be concerned about entities as well as relationships. Hence, a typical cluster would have to involve 3 or 4 entities with their associated relationships. Second, keeping something in one's short-term memory does not equate to comprehension. Hence, cluster size need not be dictated by short-term memory limitations. Third, Francalanci and Pernici [5] have noted that for purposes of reuse and schema validation, clusters should fit their task domains regardless of their size. Thus, we have not set an upper limit on the cluster size in our algorithm. On the other hand, because the word cluster implies that we are grouping more than one item, we believe that no cluster should be so small as to include only a single entity and a single relationship. We recognize that "reasonable" size clusters are preferred. However, instead of cluster size, we focus on the degree of modularity in a clustering arrangement. When all other criteria are satisfied, a solution with a greater number of clusters (which also results in relatively small sized clusters) is preferable to one with a smaller number of clusters. After all, one purpose of ERD decomposition is to assist in planning and allocation of work to manageable modules.

Another criterion proposed by Moody and Flitman [7], namely "centered," suggests that all the entities in a cluster should relate to a single central concept; and two entities, each involved in a large number of relationships, should be in different clusters. We recognize that this criterion is somewhat related to the "semantically meaningful" criterion. However, we believe that in some situations, a logical cluster could involve two highly connected entities, each one of which may be connected to a common group of large number of entities. For instance, in a video store rental system, MEMBER and RENTAL TAPE are both important entities, but are likely to be closely related to the same group of other entities. "Centered" was also the least important criterion in our survey of the expert database engineers and designers. Hence, our algorithm is not designed to specifically accomplish the "centered' criterion.

Thus, our algorithm is designed to meet the following criteria:

1. *Semantically meaningful*: People familiar with the task domain should find the clusters logical and coherent.
2. *Completeness*: Decomposition should cover all of the entities and relationships in the complete model and no entities or relationships should be left out.
3. *Non-redundancy*: Each entity and relationship should be in one, and only one, cluster.
4. *Fully connected*: All the entities in a cluster should be connected to each other, via relationship paths that are *within* the cluster.
5. *Maximal cohesion within clusters*: To the extent possible, all entities within a cluster should be closely related to each other.
6. *Minimal coupling between clusters*: To the extent possible, entities in different clusters should not be closely related to each other.
7. *High degree of modularity*: Provided all of the other criteria are satisfied, a solution with a greater number of clusters is preferred to a solution with smaller number of clusters.

Attaining any one of our criteria 1, 2, 3, 4, and 7 does not detract from attaining other criteria. However, simultaneous attainment of maximal cohesion (criterion 5) and minimal coupling (criterion 6) is impossible. Since every entity is directly or indirectly connected to one or more entities, invariably, when decomposition increases within-cluster cohesion, it also increases inter-cluster coupling. This is because, when each cluster contains a large number of entities and relationships, many entities are only indirectly connected to one another. Consequently, each cluster is not very cohesive. At the same time, since the total number of entities and relationships outside a cluster is relatively small, very few entities are connected directly to entities outside their clusters. Thus, inter-cluster coupling is also small. In a properly clustered ERD, as the average cluster size decreases, since there are not as many indirect connections between entity pairs within each cluster, within-cluster cohesion increases. At the same time, inter-cluster coupling increases since now there are many more entities outside each cluster that may be directly related to the entities in that cluster.

Thus, we need a metric that properly trades off the attainment of maximal cohesion against the attainment of minimal coupling. CM authors have studied this problem of trade-off between cohesion and coupling for quite sometime. Hence, we use one of the metrics from the CM literature to determine the goodness of fit of our algorithm. In Section 4, we present this metric and its properties along with other relevant concepts from CM.

4. Relevant concepts and metrics from CM

In CM, a firm's manufacturing system is organized into temporary work-cells (or simply, cells) to exploit the advantages of a mass production system while maintaining the flexibility required by rapid changes in product mix and demand patterns. Each work-cell consists of a number of dissimilar machines clustered together to produce a set of parts (called a "part family") with similar processing requirements [21–23]. Miltenburg and Zhang [24] explain that a good cell formation (CF) solution is such that: *within a cell, each machine processes many parts, and few parts require processing on machines outside the cell.* If we recognize that entities are similar to parts and relationships are similar to machines, the two properties of a good CF solution are precisely the objectives of maximizing cohesion and minimizing coupling in ER clustering.

Furthermore, in Section 1, we made a case for an automatic ER clustering algorithm. Because work-cells are rearranged every few months in response to the changing demand patterns and product mix, CM literature has focused on the development of efficient and automatic algorithms for solving CF problems. Therefore, we have borrowed and used several concepts from CM in our algorithm.

In the earliest CF research, Burbidge [21] develops a binary machine-part incidence matrix $[a_{ij}]$, where an entry of "1" indicates that machine i is used to process part j, while "0" indicates that machine i is not used to process part j. When an initial machine-part incidence matrix is constructed, using the similarity in the machines required for processing various parts and the similarity in the parts processed on the same machines, Burbidge's [21] method rearranges the rows and columns of the initial incidence matrix to identify clusters of highly compatible parts and machines. The machine–part incidence matrix has become a standard method of representing a CF problem. In our algorithm, we also represent an ERD by a binary matrix where entities are represented by rows and relationships by columns. A "1" in an ER matrix shows that an entity participates in the corresponding relationship, and a "0" shows that an entity does not participate in the corresponding relationship.

In CM, a "distance" metric is often defined to rearrange the rows of the matrix to bring together parts that are processed by similar machines. The distance metric measures the lack of similarity between a pair of rows. Our algorithm uses the distance metric in Eq. (1) to rearrange the entities of an ER matrix. The distance between two entities, j and k, is given by

$$d_{jk} = 4n - \sum_{i=1}^{n}(a_{ij} + a_{ik})^2 + 5\sum_{i=1}^{n}(a_{ij} - a_{ik})^2, \qquad (1)$$

where n is the number of columns (relationships) in the ER matrix and a_{ij} is the 1 (or 0) value of the entry representing the participation (or the lack of participation) of the ith entity in the jth relationship.

Initially used by Joglekar et al. [25], this distance metric has several desirable properties for assessing the relative "closeness" of entities. First, since in a binary matrix, a_{ij} and a_{ik} each can be either 1 or 0, all the distances calculated by this formula are in [0, $8n$]. If each one of a pair of entities participated in each relationship in the matrix (i.e. when all $a_{ij} = 1$) the calculated distance would be smallest ($d_{jk} = 0$). When a pair of entities is such that whenever one entity participates in a relationship, the other does not, and vice versa (i.e. when the n entries in row j are never identical with the n entries in row k), the distance is the largest ($d_{jk} = 8n$). If a pair of entities participates in many matching relationships, the calculated distance is smaller than the distance for a pair of entities that participates in only a few matching relationships. Thus, the smaller the calculated distance between two entities, the closer they are to one another in terms of their participation and non-participation with the relationships in an ERD.

In short, our distance metric is designed to indicate that entities that are involved with the largest number of same relationships are the closest. While the distance metric helps us in the rest of the clustering process, it sometimes fails to keep the *weak*, *subtype*, and *singular* (an entity involved in a relationship with only one other entity) entities in their appropriate clusters. Yet, to comply with the criterion of semantic meaningfulness, weak and subtype entities must be clustered with their respective strong and supertype entities. Similarly, singular entities must be clustered with their respective connected entities. Hence, our algorithm first removes those three types of entities (and associated relationships) before applying the above distance metric to juxtapose closely related entities and cluster them together. As explained in Section 5, the *weak*, *subtype*, and *singular* entities (and associated relationships) are then added back to the matrix in their appropriate entity clusters.

In CF, often machines are also rearranged using a similar distance metric. Once the rows and columns are rearranged, machine–part cells are identified with the dual objectives of maximizing within-cell cohesion and minimizing inter-cell coupling. As in ER clustering, in CM, typically, when a clustering arrangement is modified to increase within-clusters cohesion, the modification also results in an increase in inter-cluster coupling. Hence, it is advantageous to seek a goodness of fit metric that provides a balanced trade-off between these two objectives.

CM researchers [22,24,26,27] have proposed several different metrics to judge the goodness of fit of a CF solution. Among those metrics, Joglekar et al. [28] found that Chandrasekharan and Rajagopalan's [22] "grouping efficiency measure" and Miltenburg and Zhang's [24] "primary performance measure" balanced the trade-off between the two objectives of CF, cohesion and coupling, reasonably well. However, Chandrasekharan and Rajagopalan's [22] metric involved a weighting factor between [0, 1]. Although the recommended default value for the weighting factor is 0.5, Chandrasekharan and Rajagopalan [22] suggested that an analyst might want to change that value depending on the problem addressed. In contrast, Miltenburg and Zhang's [24] metric does not change from problem to problem. Furthermore, using some simple examples, Seifoddini [27] and Ng [29] have pointed out that, with the default value of 0.5 for the weighting factor, Chandrasekharan and Rajagopalan's [22] metric under-emphasized inter-cluster coupling. In their evaluation of the two metrics, Joglekar et al. [28] found that for some problems, Miltenburg and Zhang's [24] metric also under-emphasized inter-cluster coupling. However, in some other problems solved by maximizing one of these metrics at a time, while the solutions were equally cohesive, the solutions maximizing the Miltenburg and Zhang [24] metric displayed a smaller number of inter-cluster couplings than the solutions yielded by maximizing the Chandrasekharan and Rajagopalan [22] metric. Hence, we believe that Miltenburg and Zhang's [24] "primary performance measure" represents the best available metric for the trade-off between the objectives of maximal cohesion and minimal coupling. As such, our algorithm uses this metric to assess the goodness of fit of an ER clustering solution.

Adapted for the purposes of ER clustering, this metric is shown in Eq. (2) and will be referred to as "the goodness of fit" metric, G

$$G = G_1 - G_2, \quad -1 \leqslant G \leqslant 1, \tag{2}$$

where a binary ER matrix, A, is partitioned into k clusters $\{D_r | r = 1, 2, \ldots, k\}$; Cluster D_r, consisting of entities E_r and relationships R_r, is the rth cluster; and G_1 and G_2 are:

$$G_1 = \left(\sum_r \sum_{\substack{i \in E_r \\ j \in R_r}} a_{ij} \right) \bigg/ \left(\sum_r |E_r||R_r| \right), \quad 0 \leqslant G_1 \leqslant 1, \tag{3}$$

$$G_2 = 1 - \left(\sum_r \sum_{\substack{i \in E_r \\ j \in R_r}} a_{ij} \right) \bigg/ \left(\sum_{i,j} a_{ij} \right), \quad 0 \leqslant G_2 \leqslant 1, \tag{4}$$

where $a_{ij} = 1$, if Entity i participates in Relationship j, and 0 otherwise; $|E_r|$ denotes the number of entities, and $|R_r|$ denotes the number of relationships in D_r.

G's cohesion clause, G_1, is the sum of a_{ij} in each cluster divided by the sum of the number of elements in each cluster. It measures a cluster's density, i.e. how many of the entities participate in each of the relationships within their clusters. G_1's value is normalized by its denominator—the sum of the number of elements in each cluster ($0 \leqslant G_1 \leqslant 1$). A high value of G_1 indicates high cohesion, which is desirable.

G's coupling clause, G_2, is 1 minus the sum of a_{ij} within each cluster divided by the total number of 1's in the full matrix. G_2 measures inter-cluster relationships, normalized in terms of the total number of 1's in the full matrix ($0 \leqslant G_2 \leqslant 1$). A high value of G_2 indicates high coupling, which is undesirable. To seek a balance between the objectives of maximizing cohesion and minimizing coupling, in the algorithm presented in this paper, we seek to maximize G. In short, we have borrowed the following concepts from CM:

1. A binary matrix representation of the entities and relationships;
2. A distance metric to assess the similarities and dissimilarities between pairs of entities;
3. Rearrangement of entity rows to juxtapose the most similar entity pairs; and
4. The goodness of fit metric to assess alternative clustering arrangements.

In order to decompose a particular ERD, we first create a binary matrix, M_0, with entities as rows and relationships as columns. Thus, if p is the number of entities in an ERD, and q is the number of relationships, the dimension of M_0 is $p \times q$. In M_0, a '1' in a cell indicates that the corresponding entity participates in the corresponding relationship. When there is no participation, the cell entry is 0. Fig. 1b presents such a binary matrix representation of the ERD in Fig. 1a. The information in M_0 will allow our computerized algorithm to automatically identify several types of entities and relationships. For example, if an entity row has a '1' in only one column, it is a *singular* entity; if a relationship column has '1's in only two rows, it is binary relationship; and so on. Thus, the connectivity dimension of the entities and relationships is fully captured by the binary matrix representation of the ERD.

In addition, our algorithm requires a table, T_0, which lists all the weak and subtype entities in the ERD along with their corresponding strong or supertype entities and respective relationships. Fig. 1c shows such a table for the introductory example.

5. The clustering algorithm

Starting with the M_0 and the T_0 associated with an ERD, our algorithm consists of eight major computerized steps presented in Fig. 2 and described in detail below.

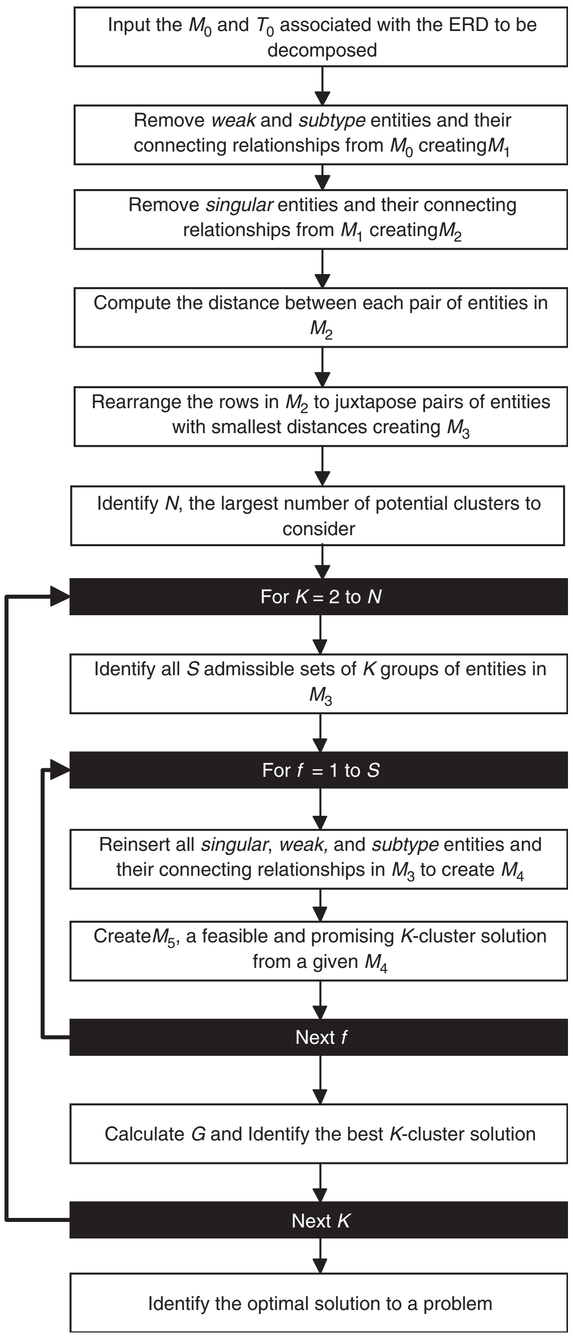

Input the M_0 and T_0 associated with the ERD to be decomposed
Remove *weak* and *subtype* entities and their connecting relationships from M_0 creating M_1
Remove *singular* entities and their connecting relationships from M_1 creating M_2
Compute the distance between each pair of entities in M_2
Rearrange the rows in M_2 to juxtapose pairs of entities with smallest distances creating M_3
Identify N, the largest number of potential clusters to consider
For $K = 2$ to N
Identify all S admissible sets of K groups of entities in M_3
For $f = 1$ to S
Reinsert all *singular*, *weak,* and *subtype* entities and their connecting relationships in M_3 to create M_4
Create M_5, a feasible and promising K-cluster solution from a given M_4
Next f
Calculate G and Identify the best K-cluster solution
Next K
Identify the optimal solution to a problem

Fig. 2. Overview of major steps in our automated algorithm.

5.1. *Input the M_0 and T_0 associated with the ERD to be decomposed*

The algorithm described here is implemented as a Java application. In our Java implementation, we identify each cell by its row and column names and cells containing '1's are shaded while cells containing '0's are not. A "problem," an ER matrix to be clustered, can be created by choosing the number of rows and columns in the matrix, then clicking on cells which should contain '1's. Converting the ERD into its corresponding M_0 and producing the corresponding T_0 are easy, mechanical, and automable tasks, since they require no information that is not already contained in the ERD. Alternatively, M_0 and T_0 could be constructed through appropriate interface with existing ERD tools. All the steps following this input of M_0 and T_0 are fully automated. That is, from this point on, no human intervention is needed before the computer produces the optimal solution.

5.2. *Remove weak and subtype entities and their connecting relationships from M_0 creating M_1*

For the semantic meaningfulness of an ER decomposition, it is essential that a weak entity is always grouped with its strong entity and a subtype entity is grouped with its supertype. Unfortunately, our distance metric would not necessarily find weak entities to be closest to their strong entities or subtype entities to be closest to their supertypes. Hence, using the information in T_0, the computerized algorithm's first step is to remove from M_0 all weak and subtype entities as well as the relationships connecting the weak to the strong and the subtype to the supertype entities. In this step, the algorithm ensures that the respective strong/supertype entity is now connected to all of the other relationships that were originally connected to its weak/subtype entity. This procedure produces a new matrix, M_1.

5.3. *Remove singular entities and their connecting relationships from M_1 creating M_2*

Our distance metric also cannot ensure that a *singular* entity is always grouped with the only entity it is connected with. If it were not grouped with that entity, the "fully connected" principle would be violated. Hence, the next step of the computerized algorithm is to identify and remove each singular entity along with its associated relationship from M_1. Sometimes, when a singular entity is removed from M_1, a previously non-singular entity becomes a singular entity in the reduced matrix. This happens when a singular entity's connected entity is related to only one other entity. Hence, the process of identification and removal of singular entities is repeated until there are no singular entities in the modified matrix. Simultaneously, in the order of their removal, a list of all the singular entities and their associated relationships is created, so that at the appropriate time, they can be reinserted in the matrix, in the reverse order of their removal. The resulting matrix is labeled M_2. Clearly, if one or more weak, subtype, or singular entities are removed, M_2's dimension, $m \times n$, will be smaller than M_0's dimension ($p \times q$).

5.4. *Compute the distance between each pair of entities in M_2*

Now, using Eq. (1), our computerized algorithm computes the distance between each possible pair of rows (entities) in M_2. As explained in Section 4, the shorter the distance between a pair, the more appropriate it is to put the two entities in the same cluster, and the larger the distance, the more appropriate it is to put the two entities in different clusters.

5.5. *Rearrange the rows in M_2 to juxtapose pairs of entities with smallest distances creating M_3*

Next, our algorithm constructs a matrix, M_3, where we leave the relationship columns in the same order as in M_2, but rearrange the rows so that pairs of entities with least distances are closest to each other. The algorithm compares all pairwise row distances, and copies the pair of rows with the least distance next to each other in M_3 (*arbitrarily one above the other*). Among the *UnusedRows*, the algorithm finds the one with the smallest distance from one of the *current edge rows*. The identified row is added to M_3 next to that edge row (above the top edge or below the bottom edge). This process is repeated until there are no *UnusedRows* in M_2. Thus, at this point, M_3 represents the rows in M_2, in an order such that the pairs of entity rows with the smallest distances are closest together.

5.6. *Identify N, the largest number of potential clusters to consider*

The next step of the automated algorithm is to find the range for K, the number of clusters to consider. As suggested earlier, every cluster must contain at least two entities. Hence, the maximum number of clusters, N, is the integer value of the number of entities divided by two. We also assume that, when an ER matrix is to be clustered, a database manager is looking for at least two clusters. Therefore, our algorithm assumes that the desired number of clusters, K, is in [2, N].

5.7. *Identify the "best" K-cluster solution for every possible value of K*

This is the most involved part of the algorithm with several sub-steps detailed below. First, for each value of K ($= 2$ to N), the algorithm identifies all admissible sets (say S) of K groups of entities as described in Section 5.7.1. Next, for each one of the S sets for a given value of K, the algorithm carries out the sub-steps described in Section 5.7.2 creating S matrices that are labeled M_4. Then, each one of the M_4's is used to create a feasible and promising K-cluster solution as described in Section 5.7.3. Once all S feasible and promising K-cluster solutions are created, as described in Section 5.7.4, we calculate the goodness of fit, G, for each one of them. Among the S solutions, the solution with the largest value of G is identified as the best K-cluster solution.

5.7.1. *Identify all S admissible sets of K groups of entities in M_3*

Since a matrix's own boundaries (i.e. the first and the last rows) also serve as the boundaries for the first and the last sub-matrices, respectively, to construct a partition resulting in K groups of entities (rows), we need $K-1$ other horizontal dividers between rows. It is desirable to draw the dividers between the $K-1$ pairs of adjacent rows that have the greatest distances. However, since a cluster must have at least two entities, divider locations immediately after the first row, or immediately before the last row, cannot be considered selectable. Initially, all other possible divider locations are considered as *selectable*. The pair of adjacent rows with the largest distance is found and the first divider is placed between those two rows. Since an entity group must contain at least two entities, entity pairs involving the rows immediately adjacent (on either side) to a selected divider are no longer selectable. Among the remaining pairs of entities, the pair of adjacent rows with the largest distance is found and the next divider is placed between those two rows. This process is repeated $K-2$ times to find *all but the last* divider. Up until this point, ties in largest distances are broken arbitrarily by choosing the first of the tied divider locations.

For the ($K-1$)st divider, often, there are several selectable divider locations with the same and largest distance between adjacent rows. Hence, the last divider is handled differently to ensure that arbitrary tie-breaking does not cause a better solution to be missed. Suppose that there are S candidates for the ($K-1$)st divider. Together, the first $K-2$ dividers and *one of* the S candidates produce an admissible set of K groups of entities in M_3. Thus, there are S sets of K groups of entities to consider. Once all S sets of K groups of entities are identified, for each one of these S sets, the algorithm carries out the sub-steps described in Sections 5.7.2 and 5.7.3.

When, for a given value of K, it is *impossible* to select *any* admissible ($K-1$)st divider, that value of K is eliminated from further consideration.

5.7.2. Reinsert all singular, weak, and subtype entities and their connecting relationships in M_3 to create M_4

First, for each admissible set, $f \in S$, of K groups of entities, the algorithm constructs a new matrix M_4 by reinserting, in the reverse order of their removal, each singular, weak, and subtype entity immediately above its respective connected, strong or supertype entity. If necessary, a divider location is adjusted to ensure that the reinserted entity and its relevant entity are in the same group. All the relationships are reinserted as the last columns of the matrix and any altered relationship column ‘1’s are adjusted back to their original entity rows. Thus, at the end of this procedure, for each one of the S sets, the corresponding M_4 is a $p \times q$ matrix consisting of all the entities and relationships in M_0 and for a given value of K, the total number of M_4’s is S.

5.7.3. Create M_5, a feasible and promising K-cluster solution from a given M_4

Now, the columns in a given M_4 need to be rearranged so that each group of entities is clustered with suitable relationships. Our aim is to maximize the goodness of fit of the resulting K-cluster solution. Given the mathematical formula (Eq. (2)) for G, to maximize G, one should cluster the largest possible number of relevant relationships with the smallest group of entities.

Hence, for each M_4, the computerized algorithm creates a blank matrix, M_5, of the same dimensions ($p \times q$) as M_4. It copies into M_5 all the entity (row) names in the same order as in M_4. The chosen $K-1$ dividers are also copied in M_5. Along with the matrix boundaries, the dividers help identify the K groups of entities in M_5. Then, an iterative process of column rearrangement and cluster identification begins.

Each iteration involves first identifying the smallest group of entities (rows) that has not yet been put into a cluster. In case of a tie in the entity group size, the first group is selected. The algorithm considers each relationship that has not already been copied to M_5. If a relationship has at least as many 1’s in the rows of the selected group of entities, as it has in the rows of any of the other un-clustered groups of entities, that relationship column is copied to matrix M_5 in the next available column spot. When consideration of all relationships is done, the copied relationships and the selected group of entities are declared as a cluster, and the algorithm moves on to the identification of the next cluster. When all K clusters are constructed, M_5 represents the feasible and promising K-cluster solution resulting from a given M_4.

As can be noted, this is a greedy procedure for maximizing the G-value of the solution. However, there is no guarantee that an exhaustive approach would not have found a clustering arrangement with a higher value of G for this set of K groups of entities. That is why we call this solution as a “feasible and promising” solution.

5.7.4. Calculate G and identify the best K-cluster solution

Once all S feasible and promising K-cluster solutions have been created, the automated algorithm calculates the goodness of fit, G, for each one of them. The solution with the largest value of G is identified as the best K-cluster solution.

5.8. Identify the optimal solution to a problem

Finally, all the best K-cluster solutions for the entire range of K’s (2 to N) are compared based on their G values, and the solution with the highest G is chosen as the “optimal” clustering solution. Although we use the word “optimal” to refer to the best of the best solutions for various values of K, it should be clear that we are not claiming that our solution is globally optimal. Our algorithm is a heuristic that employs a greedy approach. Thus, we use the word “optimal” simply to avoid the use of the awkward phrase, "best of the best K-cluster solution."

5.9. A special feature of the algorithm

While our algorithm is designed to consider all possible values of the number of clusters, $K = 2$ to N, we have also incorporated a special feature in it. A manager may ask for just the best solution for a given value of K. This is a useful feature for a manager who wants to allocate the database development and maintenance work to a given number of individuals or teams.

6. Algorithm complexity

While our algorithm is complex, it is polynomial in time. In terms of number of solutions evaluated for a $p \times q$ matrix containing no weak/subtype or singular entities, our algorithm's Big O efficiency is $O(p^2)$.

Note that, K, the number of clusters in a solution can range from 2 to $p/2$ (Section 5.6). In the worst-case scenario (if all distances are tied), for each K, there are $S = (p–K)$ admissible sets of K entity groups (Section 5.7.1). Although each one of these sets must go through the steps of reinserting weak/subtype and singular entities (Section 5.7.2) followed by creating a feasible and promising solution by clustering each relationship with its appropriate entity group (Section 5.7.3), these steps are carried out only once for each admissible set. Therefore, in terms of the number of solutions evaluated, the worst-case efficiency is $\sum_{K=2}^{p/2}(p-K)$. It can be shown that

$$\sum_{K=2}^{p/2}(p-K) = \left(\frac{p}{2}-1\right)\left(\frac{3p}{4}-1\right). \qquad (5)$$

Assuming that p is large, this is roughly equal to $(\frac{3}{8})p^2$. Hence, the order of magnitude is $O(p^2)$. Of course, the actual number of solutions evaluated is often considerably smaller. In the introductory example, with $p = 11$, the Big O analysis suggests that, in the worst case, we have to evaluate 121 possible solutions, or more precisely (using Eq. (5)) 33 possible solutions. In its actual execution for the introductory example, the algorithm evaluated only 4 possible solutions.

The complexity of our algorithm is kept manageable by a series of strategies including: definition of a distance metric to assess the closeness of entities, arbitrary tie-breaking in various steps of the algorithm, and using the properties of the G metric in clustering relationships with entity groups. Several of these strategies are greedy and global optimality of our solution is not guaranteed. However, our algorithm is practical even for large enterprise data models with hundreds of entities and relationships.

On the other hand, a brute force search for the optimal solution would be exponential in time. Remember that in a $p \times q$ matrix, there can be K (= 2 to $p/2$) clusters. For a given K, there are ${}_pC_K$ entity groups and ${}_qC_K$ relationship groups. Multiplying these independent possibilities, we get $({}_pC_{Kq}C_K)$ possible clustering arrangements. Thus, the total number of clustering arrangements to consider is $\sum_{K=2}^{p/2}({}_pC_{Kq}C_k)$. We cannot present a simple formula for this total number. However, for the introductory example involving $p = 11$ and $q = 14$, according to this formula, a brute force approach will require an evaluation of 1,320,319 possible solutions! When p and q are larger than those in our introductory example, a brute force approach would require the evaluation of exponentially greater number of possible solutions. In comparison, our algorithm's worst-case scenario of evaluating 33 solutions for the introductory example seems to be considerably more desirable! The fact that our algorithm actually evaluated only 4 possible solutions for the introductory example shows the true efficiency of our algorithm.

Some readers may suggest that evaluating a couple of million solutions (required by a brute force approach to a small problem) would be feasible and would allow a comparison of our greedy solution with the globally optimal solution. However, by now, the reader also knows that significant computation is needed in creating and evaluating each one of millions of solutions required for the brute force approach. Hence, identifying the globally optimal solution for even a small problem is not practical.

7. An assessment of the effectiveness of our algorithm

We use the criteria discussed in Section 3 to assess the effectiveness of our algorithm. Some criteria are qualitative and difficult to judge objectively. For example, one criterion is that the clusters should be semantically meaningful. Our procedure for removing and reinserting weak and subtype entities is designed to partly fulfill this criterion. However, another interpretation of this criterion would require that the resulting clusters should be capable of being identified as meaningful modules or subsystems of the larger system. An automatic algorithm cannot show the semantic meaningfulness of the resulting clusters by naming the clusters as meaningful modules. That is a manual task for practicing database engineers and designers. All we can report is that we worked on numerous ERDs, and in every case, we have been able to find semantically meaningful names for the clusters resulting from our algorithm.

Some criteria discussed in Section 3 are absolute. That is, they represent requirements that must be met by any good clustering solution. Absolute criteria include "completeness," "non-redundancy," and "fully connected" principles. Our approach is algorithmically forced to meet these criteria. Indeed, it was for the "fully-connected" principle that we developed a special procedure to deal with the *singular* entities.

Some criteria are relative. They are best discussed in comparison to other methods and in the context of specific problems. Ideally, we should compare the solutions to numerous problems obtained by our algorithm with the solutions to the same problems obtained by other algorithms. Unfortunately, most of the literature discusses general principles of clustering, using small size examples (involving 8 or 9 entities and an even smaller number of relationships). We have encountered only two sizable ER clustering problems in the published literature. Next, we present our algorithm's solutions to each one of those problems and compare them with the solutions to the same problems obtained by other algorithms.

7.1. Feldman and Miller problem

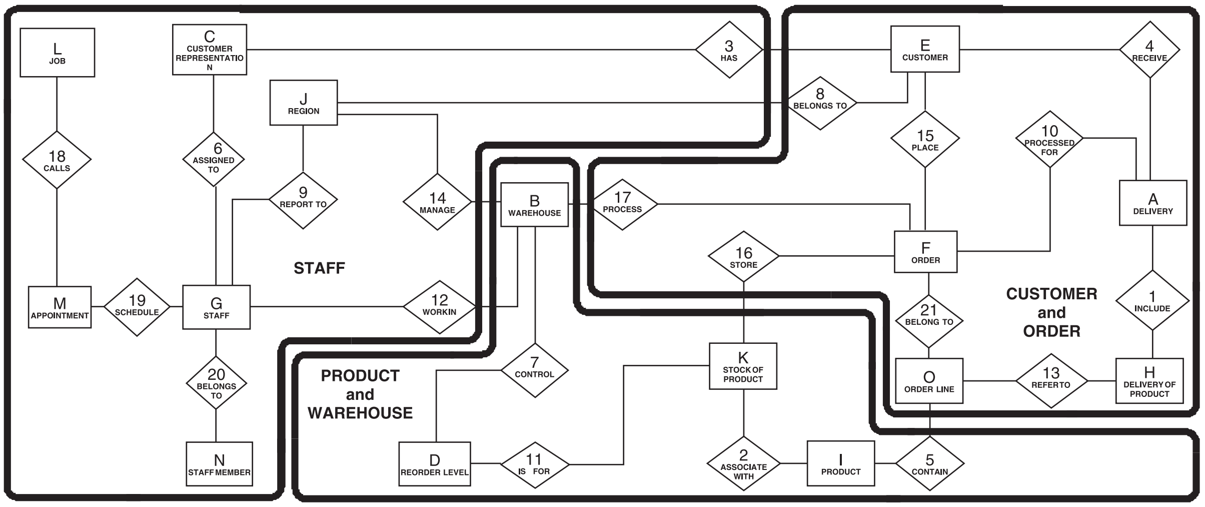
First, we use Feldman and Miller's [3, p. 352] problem (Fig. 3). Their solution to this problem is presented in Fig. 3a. Using Feldman and Miller's initial ERD, we constructed a matrix M_0 and a table T_0 and used our algorithm to obtain the solution shown in Fig. 3b.

Let us compare the two solutions in Figs. 3a and b on the criteria we have established earlier. First, note that both solutions satisfy the absolute criteria of completeness, non-redundancy, and fully connectedness. Feldman and Miller's [3] solution consists of 3 clusters, while our solution consists of 4 clusters. Thus, our solution has a higher degree of modularity. Thus, our solution offers greater flexibility to a database manager in allocating modular tasks to information technology personnel (or teams). Of course, if a manager so desires, two or more of our clusters could always be assigned to a single team.

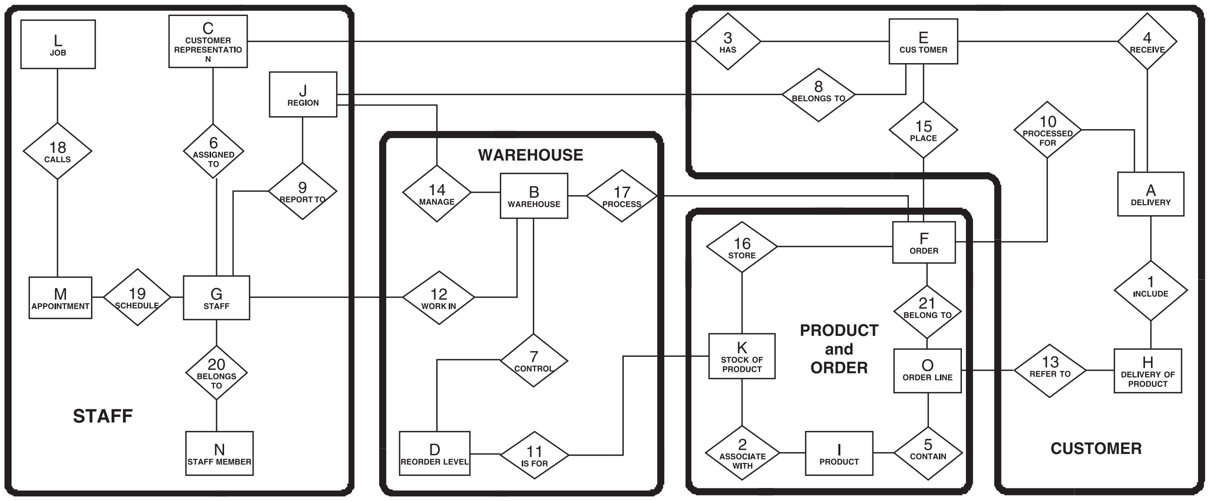
One dimension of semantic meaningfulness is whether or not every weak and subtype entity is clustered with its strong or supertype entity. Our algorithm is designed to ensure that. Feldman and Miller's solution also fulfills this requirement. Another dimension of this criterion is whether or not the various clusters in a solution are semantically meaningful. We have taken the liberty to name the clusters in Figs. 3a and b in a semantically meaningful way. We find that on both diagrams, all clusters are semantically meaningful. However, notice that in labeling two of the clusters in Feldman and Miller's solution, in order to be adequately descriptive, we had to use the names of two entities each. In contrast, only one of our four clusters needed the names of two entities. Thus, our solution fulfils Moody and Flitman's "centered" criterion a little better. However, because in our view, the "centered" criterion is not very important, and because the semantic meaningfulness of a naming scheme lies in the eye of the beholder, we do not want to attach too much importance to these differences. We are satisfied that our clusters are semantically meaningful.

Compared to our solution, Feldman and Miller's solution has a smaller number of relationships that are connected outside their own clusters. This is reflected in the coupling metric, G_2, which is 0.167 for Feldman and Miller's solution and 0.214 for our solution. However, our solution has a substantially larger proportion of within-cluster entity pairs that are directly connected to each other. This is reflected in the cohesion metric, G_1, which is 0.429 for our solution but only 0.321 for Feldman and Miller's solution. Thus, our solution provides a better balance between cohesion and coupling, reflected in the fact that Feldman and Miller's solution has a G of only 0.154, whereas our solution attains a G of 0.215.

We also showed Figs. 3a and b (with their titles changed to "Solution A1" and "Solution A2," respectively, and the labeling of the clusters removed to avoid any bias to our expert database engineers and designers and asked them to assess each solution on Moody and Flitman's [7] nine criteria as well as



(a)



(b)

Fig. 3. Feldman and Miller's problem: (a) Feldman and Miller's solution ($G_1 = 0.321$, $G_2 = 0.167$, $G = 0.154$); (b) our solution ($G_1 = 0.429$, $G_2 = 0.214$, $G = 0.215$).

overall (scale: 1 = very low fulfillment and 5 = Very high fulfillment). As shown by the mean scores in Table 2a, our algorithm's solution scored consistently better than Feldman and Miller's solution on each criterion as well as overall (except on "non-redundancy," where both scored a 5). We used a two-population t-test ($\alpha = 0.05$ level of significance) to see if the differences in the mean scores of the two solutions are statistically significant. On seven of the nine criteria, and also for the overall score, the differences in the mean scores were statistically significant.

Based on the foregoing discussion, we believe that our algorithm has produced a solution for Feldman and Miller's [3] problem that is superior to their own solution.

7.2. *Moody and Flitman problem*

For a second comparison, we use Moody and Flitman's [14] example. Moody and Flitman's [14, p. 9] optimal solution to the problem is presented in Fig. 4a. Using Moody and Flitman's initial ERD, we constructed a matrix M_0 and a table T_0 and used

Table 2
A comparison of our solutions with alternative methods' solutions for two problems

(a) Feldman and Miller's problem

Criteria	Expert database engineers' mean ratings		
	Feldman and Miller's solution (A1)	Our solution (A2)	Significance
Semantically meaningful	2.75	4.69	Y
Completeness	4.31	4.50	N
Manageable size	3.31	4.75	Y
Fully connected	4.00	4.56	Y
Maximal cohesion	2.75	4.75	Y
Non-redundancy	5.00	5.00	N
Minimal Coupling	2.56	4.75	Y
Modularity	2.56	4.81	Y
Centered	2.50	4.13	Y
Overall score	3.36	4.68	Y

(b) Moody and Flitman's problem

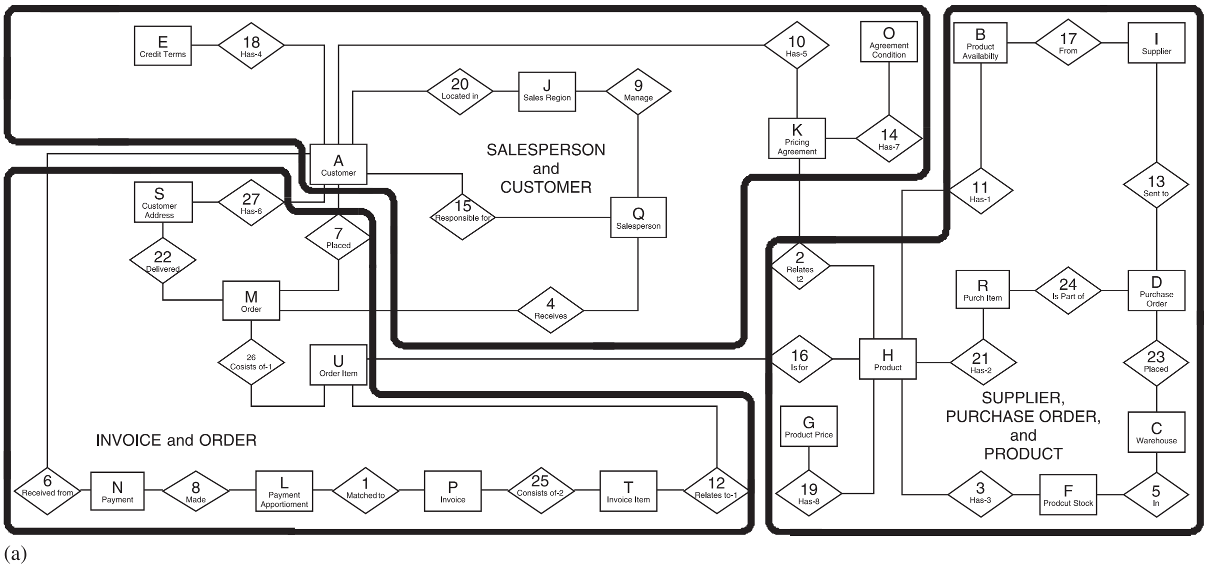
Criteria	Expert database engineers' mean ratings		
	Our solution (B1)	Moody and Flitman's solution (B2)	Significance
Semantically meaningful	4.88	2.19	Y
Completeness	4.94	3.81	Y
Manageable size	4.88	2.56	Y
Fully connected	4.88	3.75	Y
Maximal cohesion	4.75	2.25	Y
Non-redundancy	5.00	5.00	N
Minimal Coupling	4.69	2.00	Y
Modularity	4.94	1.75	Y
Centered	4.56	2.69	Y
Overall score	4.84	2.94	Y

$\alpha = 0.05$ level of significance.

our algorithm to obtain the optimal solution shown in Fig. 4b.

Note that both solutions satisfy the absolute criteria of completeness, non-redundancy, and fully connectedness. Moody and Flitman's solution consists of 3 clusters, while our solution consists of 6 clusters. Thus, our solution has a much higher degree of modularity and offers substantially greater flexibility to a database manager in allocating modular tasks to various information technology personnel (or teams). Of course, if a manager so desires, two or more of our clusters could always be assigned to a single team. Also, as explained in Section 5.9, if a manager has a predetermined K, the number of clusters he/she desires, our algorithm can also produce the best solution for that K.

Second, from the module labels in Fig. 4b, note that the clusters produced by our algorithm are semantically meaningful and five of the six clusters can be adequately described by single entity names. Although in presenting their solution, Moody and Flitman [14] use single entity names for each one of their clusters, we believe the names we have used in Fig. 4a are better descriptors of the major entities in those clusters. As can be seen, each one of Moody and Flitman's clusters need two or more entity names. Thus, in spite of their explicit criterion of "centered" clusters [7], their solution is not as centered as ours. Of course, as we have said before, the semantic meaningfulness of a naming scheme is always subjective. Hence, one need not attach too much significance to these differences. We are satisfied that our clusters are semantically meaningful, and that our algorithm has ensured that any weak/subtype entities are suitably clustered with their respective strong/supertype entities.



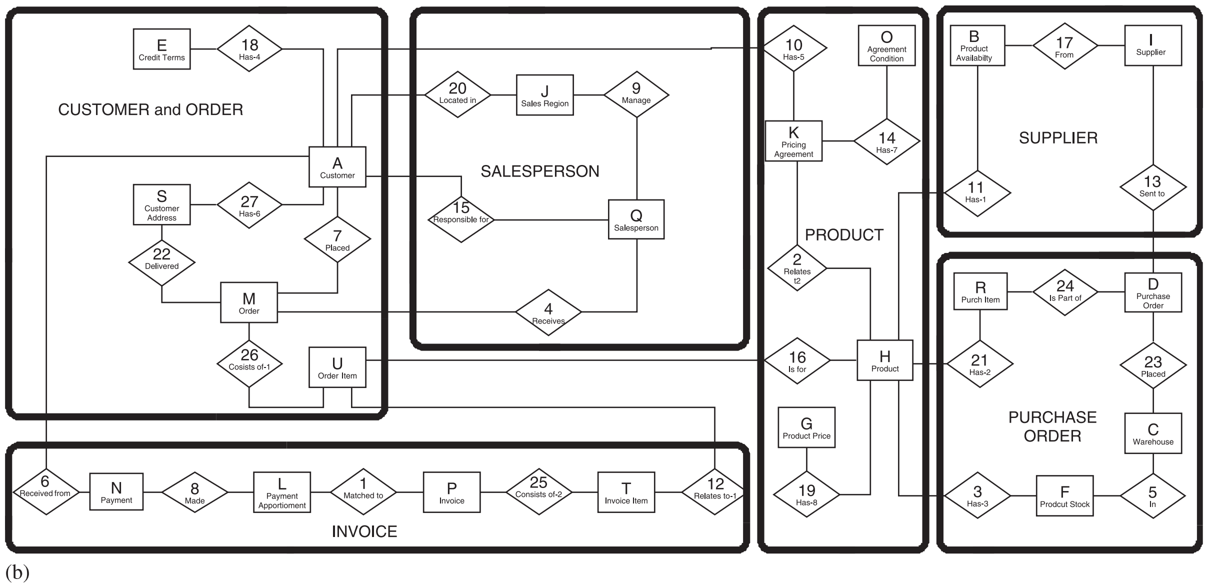


Fig. 4. Moody and Flitman's problem: (a) Moody and Flitman's solution ($G_1 = 0.244$, $G_2 = 0.130$, $G = 0.114$); (b) our solution ($G_1 = 0.434$, $G_2 = 0.203$, $G = 0.231$).

Third, compared to our solution, Moody and Flitman's solution has a smaller number of relationships that are connected outside their own clusters. This is reflected in the coupling metric, G_2, which is 0.130 for Moody and Flitman's solution and 0.203 for our solution. However, our solution has a substantially larger proportion of within-cluster entity pairs that are directly connected to each other. This is reflected in the cohesion metric, G_1, which is 0.434 for our solution but only 0.244 for Moody and Flitman's solution. Thus, our solution provides a better balance between cohesion and coupling. This is reflected in the fact that Moody and Flitman's solution has a G of only 0.114, whereas our solution attains a G of 0.231.

Finally, we showed Moody and Flitman's solution in Fig. 4a and our solution in Fig. 4b (with their titles changed to "Solution B2" and "Solution B1," respectively, and the labeling of the clusters removed to avoid any bias to our expert database engineers and designers and asked them to evaluate each solution on Moody and Flitman's [7] nine criteria as well as overall. As Table 2b shows, our solution scored consistently better than their solution on each criterion as well as overall (except the "non-redundancy" criterion, where both solutions scored 5). The higher scores of our solution were statistically significant on eight of the nine criteria as well as the overall score.

In view of the above considerations, we believe that our algorithm has resulted in a substantially better solution than Moody and Flitman's solution to their problem.

8. Conclusion and directions for future research

We have described an algorithm for decomposing ERDs into manageable modules. Unlike earlier efforts, our algorithm clusters not only the entities but also the relationships involved. In designing the algorithm, we have capitalized on certain concepts and metrics from research in CM. At the same time, we exploit some of the unique characteristics of ERDs. Our method is fully automated; that is, using only the information contained in an ERD, through an automated computer procedure, it identifies suitable entity and relationship clusters without any further human (subjective) intervention. The algorithm has been implemented as an easy to use Java application, and tested for a variety of sample problems. We have discussed how our algorithm fulfills a comprehensive set of criteria for a good decomposition of ERDs. We have also compared our algorithm's application to two published problems solved by other algorithms. Our algorithm produces a more cohesive set of clusters while keeping inter-cluster coupling small. Our solution also offers a higher degree of modularity than that offered by other algorithms' solutions.

While our algorithm produces very good solutions, it cannot guarantee their global optimality. First, our goodness of fit metric is not necessarily perfect. It is the best of several available metrics that attempt to balance the two objectives of maximizing cohesion while minimizing coupling. Future research may produce a better metric that may lead to a better algorithm. Second, in order to keep the computational time reasonable, in case of ties, our algorithm simply selects one of the choices arbitrarily. We also use a greedy approach to column rearrangement and cluster formation.

Nevertheless, our survey of the expert database engineers and designers at NASA has shown that practicing database engineers and designers find our solutions preferable to those produced by other available algorithms. Hence, throughout the system design, development, and implementation process, practicing database managers should benefit from the manageable modular decompositions of their ERDs produced by our algorithm.

References

[1] P.P. Chen, The entity–relationship model—towards a unified view of data, ACM Trans. Database Syst. 1 (1) (1976) 9–36.

[2] R. Maier, Benefits and quality of data modeling—results of an empirical analysis, in: B. Thalhiem (Ed.), Proceedings of 15th International Conference on the Entity–Relationship Approach, Springer, Berlin, Heidelberg, 1996, pp. 245–260.

[3] P. Feldman, D. Miller, Entity model clustering: structuring a data model by abstraction, Comput. J. 29 (4) (1986) 348–360.

[4] J. Akoka, I. Comyn-Wattiau, Entity–relationship and object-oriented model automatic clustering, Data Knowl. Eng. 20 (1) (1996) 87–117.

[5] C. Francalanci, B. Pernici, Abstraction levels for entity–relationship schemas, in: P. Loucopoulus (Ed.), Proceedings of 13th International Conference on the Entity–Relationship Approach, Springer, Berlin, Heidelberg, 1994, pp. 456–473.

[6] T.J. Teorey, G. Wei, D.L. Bolton, J. Koenig, ER model clustering as an aid for user communication and Documentation in database design, Commun. ACM 32 (8) (1989) 975–987.

[7] D. Moody, A. Flitman, A methodology for clustering entity relationship models—a human information processing approach, in: J. Akoka, M. Bouzeghoub, I. Comyn-Wattiau, E. Metais (Eds.), Proceedings of 18th International Conference on the Entity–Relationship Approach, Springer, Berlin, Heidelberg, 1999, pp. 114–130.

[8] M. Genero, G. Poels, M. Piattini, Defining and validating metrics for assessing the maintainability of entity–relationship diagrams, Working Papers of Faculty of Economics and Business Administration: Ghent University—Belgium 03/199, 2003.

[9] A. Badia, Entity–relationship modeling revisited, ACM SIGMOD Manage. Data Notes 33 (1) (2004) 77–82.

[10] C. Francalanci, A. Fuggetta, Integrating information requirements along processes: a survey and research directions, ACM SIGSOFT Software Eng. Notes 22 (1) (1997) 68–74.

[11] S. Castano, V. De Antonellis, M.G. Fugini, B. Pernici, Conceptual schema analysis: techniques and applications, ACM Trans. Database Syst. 23 (3) (1998) 286–333.

[12] P. Jaeschke, A. Oberweis, W. Stucky, Extending ER model clustering by relationship clustering, in: R. Elmasri, V. Kouramajian, B. Thalheim (Eds.), Proceedings of 12th International Conference on the Entity–Relationship Approach, Springer, Berlin, Heidelberg, 1994, pp. 451–462.

[13] S.B. Huffman, R.V. Zoeller, A rule-based system tool for automated ER model clustering, in: F.H. Lochovsky (Ed.), Proceedings of 8th International Conference on the Entity–Relationship Approach, Springer, Berlin, Heidelberg, 1989, pp. 221–236.

[14] D. Moody, A. Flitman, A methodology for decomposing entity relationship models—a human information processing approach, Department of Information Systems Working Paper, University of Melbourne, Parkville, Victoria, Australia, 1999, pp. 1–43.

[15] M. Halkidi, M. Vazirgiannis, Y. Batistakis, Clustering algorithms and validity measures, Proceedings of 2001 Scientific and Statistical Database Management (SSDBM) Conference, Virginia, 2001, pp. 3–22.

[16] D.L. Moody, A.R. Flitman, A decomposition method for entity relationship models: a systems theoretic approach, International Conference on Systems Thinking in Management 2000, in: G. Altmann, J. Lamp, P.E.D. Love, P. Mandal, R. Smith, M. Warren (Eds.), Proceedings of the International Conference on Systems Thinking in Management, Technical University of Aachen, Geelong, Australia, 2000, pp. 462–469.

[17] J. Akoka, I. Comyn-Wattiau, Framework for automatic clustering of semantic models, in: R. Elmasri, V. Kouramajian, B. Thalheim (Eds.), Proceedings of 12th International Conference on the Entity–Relationship Approach, Springer, Berlin, Heidelberg, 1993, pp. 438–450.

[18] J. Martin, Strategies for Data-Planning Methodologies, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[19] L.J. Campbell, T.A. Halpin, H.A. Proper, Conceptual schemas with abstractions—making flat schemas more comprehensible, Data Knowl. Eng. 20 (1) (1996) 39–85.

[20] D.L. Moody, Entity connectivity vs. hierarchical leveling as a basis for data model clustering: an experimental analysis, Database Expert Syst. Appl. 2736 (2003) 77–87.

[21] J.L. Burbidge, Production flow analysis, Prod. Eng. 42 (12) (1963) 742–752.

[22] M.P. Chandrasekharan, R. Rajagopalan, An ideal seed non-hierarchical clustering algorithm for cellular manufacturing, Int. J. Prod. Res. 24 (2) (1986) 451–464.

[23] A.J. Vakharia, U. Wemmerlov, A comparative investigation of hierarchical clustering techniques and dissimilarity measures applied to cell formation problem, J. Oper. Manage. 13 (2) (1995) 117–138.

[24] J. Miltenburg, W. Zhang, A comparative evaluation of nine well-known algorithms for solving the cell formation problem in group technology, J. Oper. Manage. 10 (1) (1991) 44–72.

[25] P. Joglekar, M. Tavana, S. Banerjee, A clustering algorithm for identifying information subsystems, J. Int. Inform. Manage. 3 (1994) 129–141.

[26] C. Mosier, J. Yelle, G. Walker, Survey of similarity coefficient based methods as applied to the group technology configuration problem, Omega 25 (1) (1997) 65–79.

[27] H. Seifoddini, A note on the similarity coefficient method and the problem of improper machine assignment in group technology applications, Int. J. Prod. Res. 27 (2) (1989) 1161–1165.

[28] P. Joglekar, Q. Chung, M. Tavana, Note on a comparative evaluation of nine well-known algorithms for solving the cell formulation problem in group technology, J. Appl. Math. Decis. Sci. 5 (4) (2001) 253–268.

[29] S.M. Ng, On the characterization and measure of machine cells in group technology, Oper. Res. 44 (5) (1996) 735–744.