Proceedings of the 2018 IISE Annual Conference
K. Barker, D. Berry, C. Rainwater, eds.

Clustering Design Structure Matrices: A Comparison of Methods Using Minimum Description Length

Amol Kulkarni, Connor Jennings, Michael Hoffman, Erica Blanco, Janis P. Terpenny and Timothy Simpson
Pennsylvania State University
University Park, Pennsylvania, USA

Abstract

Understanding interactions between components is fundamental in the design of products. Design Structure Matrices (DSMs) are often used to represent the relationships between every component or subsystem in a product. The complex network of interactions can then be clustered into subassemblies and other hierarchies, aiding designers in making critical decisions that will impact assembly, maintenance, and end-of-life disposal. This paper explores three methods for clustering components in a DSM to create a modular product architecture: (1) genetic algorithm, (2) hierarchical clustering, and (3) divisive clustering using a graph. A discussion on each algorithm is followed by an industrial example. This paper leads to the conclusion that genetic algorithm is better at identifying complex structures like bus module, 3D structure and overlapping cluster whereas hierarchical and divisive clustering are computationally inexpensive and are able to find optimal DSMs faster than the genetic algorithm.

Keywords

Design Structure Matrix, Optimization, Genetic Algorithm, Clustering, Modularity, Minimum Description Length.

1. Introduction

Design Structure Matrices (DSMs) are used to create product family architecture [1]. DSMs are useful in situations where determining methods of (re)design and definition of modules in product architecture are required. Product architectures are defined as "the abstract conceptual structures underlying the functioning of engineering artifacts [2]." An architecture shows the relationships between physical components within a product and allows the creation of more modular designs through the use of product families. A product family is defined as "a group of related products that are derived from a common set of components, modules, and/or subsystems to satisfy a variety of market applications where the common 'elements' constitute the product platform" [3]. Creation and development of modules and product platforms allows companies to achieve commonality which is critical to maintaining cost effectiveness. Some benefits of platforming are reducing time-to-market, shortening lead-time, lowering costs, maximizing commonality, reusing resources, standardizing production, increasing efficiency, improving responsiveness, and being able to leverage resources across numerous projects [4].

The process of creating modular designs can be very challenging for many manufacturing industries. Determining effective methods and algorithms for clustering DSMs into modules is necessary in order to understand how modular assemblies can be created when considering interactions between the components in a product. The goal in this work is to examine ways to identify relationships between components and cluster (i.e., group) components in such a way that forms an effective modular architecture.

2. Background

Product modularity is characterized as a product architecture that is based on coupling groups of components that have a high degree of interaction thus creating modules from these groups. To date, effective clustering has been attempted through graph clustering techniques, methods for identifying product modules, and methods for designing product platforms using modules [5]. Others have analyzed complex products as a network of components that share technical interfaces [6]. One group of researchers presented a method to use Generational Variety Index

(GVI), Product Line Commonality Index (PCI), and Design Structure Matrix (DSM) to organize components in such a way that creates groups based on commonality [7].

Platforming has many benefits that can save a company time, money, and resources [4]. Platforming architecture is defined by a trade-off between commonality and distinctiveness among products [8]. Distinct products will have fewer parts in common with other products in the same family. Clustering techniques can be used to create modular architectures that increase the commonality of parts across products. In a good clustering arrangement, items within a group will have a high degree of similarity while being dissimilar from items in other groups. In the case of product design, the objects being clustered are the components of the product and the groups are the corresponding modules in that product. Two components are considered to be "similar" if there are shared interfaces with many of the same components. Thus, components assigned to a module will have similar relationships to other components in the product.

One of the challenges when clustering a DSM is identifying unique structures such as overlapping clusters, bus modules, or three-dimensional clusters [6]. A good clustering algorithm should be able to identify these structures when searching for an optimal clustering arrangement. Figure 3 shows some examples of these structures.

In each arrangement shown in Figure 1, the rows and columns of the matrix represent components of a product. The cell at the intersection of two components contains an "X" if those two components share an interface. In each arrangement, an initial, random arrangement is shown, followed by the clustered version. Figure 1(a) shows the simplest case, where there are two non-overlapping clusters. Figure 1(b) shows three overlapping clusters, while Figure 1(c) shows three overlapping clusters with a bus module. It can be seen in Figure 1(c) that component G interfaces with all other components in this arrangement, making it a bus module. Lastly, Figure 1(d) shows an arrangement overlaps with two other clusters, which can be particularly hard to identify by some clustering methods.

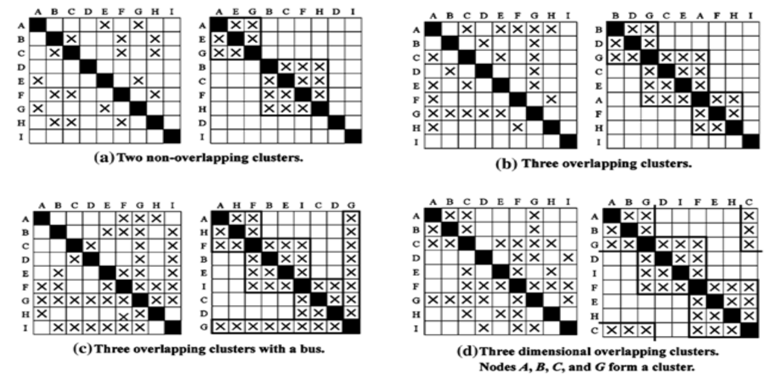
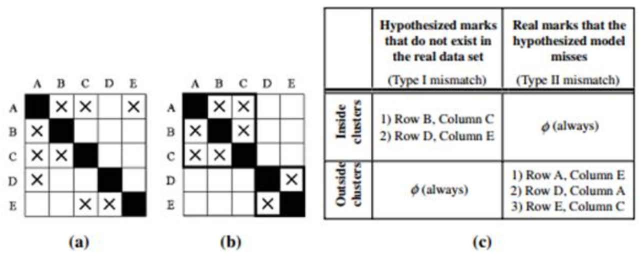

(a) Two non-overlapping clusters.
(b) Three overlapping clusters.
(c) Three overlapping clusters with a bus.
(d) Three dimensional overlapping clusters. Nodes *A*, *B*, *C*, and *G* form a cluster.

Figure 1: DSM cluster structures [6]

Another important consideration is the metric by which the clustering arrangements is evaluated. The metric that is used in this work is the minimum description length (MDL) as first described by Rissanen [9]. MDL measures the length of the description of a model and is optimal when it is at its minimum value. The model description in the case of clustering product components is the assignment of components to a cluster [5]. The MDL metric consists of the model description length and the mismatched data description. The assignment of components to clusters is the model description length. Given this assignment of product components to clusters, a hypothesized model can be created in which every component in a cluster interfaces with every other component in the same cluster.

The difference between the hypothesized model and the actual data is the mismatched data description. These mismatches are classified as either Type I or Type II mismatches. A Type I mismatch indicates that a relationship in the hypothesized model does not exist in the actual data. A Type II mismatch is a relationship in the actual data that is not covered by the hypothesized model. These mismatches are summarized in Figure 2.



	Hypothesized marks that do not exist in the real data set (Type I mismatch)	Real marks that the hypothesized model misses (Type II mismatch)
Inside clusters	1) Row B, Column C 2) Row D, Column E	ϕ (always)
Outside clusters	ϕ (always)	1) Row A, Column E 2) Row D, Column A 3) Row E, Column C

Figure 2: MDL mismatched data description [5]

The actual data is shown in Figure 2(a), and the hypothesized model from a given clustering arrangement is shown in Figure 2(b). We can see that there are some points in the hypothesized model that are not present in the actual data (Type I errors) as well as some points in the actual data that are not captured by the solution (Type II errors). These mismatches are described by the table in Figure 2(c).

Minimizing MDL involves a tradeoff between the description length of model parameters (assignment of components to clusters) and the description length of modeling errors. Given a clustering arrangement for a DSM, MDL is calculated using the following formula [9]:

$$f_{DSM}(M) = (1-\alpha-\beta)(n_c \log_2 n_n + \log_2 n_n \sum_{i=1}^{n_c} cl_i + \alpha[|S_1|(2\log_2 n_n + 1)] + \beta[|S_2|(2\log_2 n_n + 1)] \quad (1)$$

where $f_{DSM}(M)$ is the MDL for a given DSM with model M, n_c is the number of clusters in the model, n_n is the number of components, cl_i is the number of components in cluster i, S_1 and S_2 are the set of Type I and Type II errors, respectively, and α and β are the respective weights for Type I and Type II error description length.

3.Clustering Algorithms

Three algorithms are used to cluster DSMs: (1) hierarchical clustering, (2) divisive clustering, and (3) a genetic algorithm. These three algorithms were chosen based on their commonality [6,10] and, flexibility [5]. The objective for each method is to find a clustering arrangement that minimizes MDL. These methods are introduced next and then applied to a set of sample DSMs so that solutions can be compared across algorithms.

3.1 Hierarchical Clustering

Hierarchical clustering is one of the most common data mining clustering algorithms [10]. The algorithm is built on the concept of finding the relationships between each component and building a hierarchy to visualize the relationships. The algorithm works by first taking the original DSM and making each row a distinct instance. Every column in the row becomes a direction and all the directions combine together to make a coordinate. The distance between each component is calculated using standard distance metrics (Euclidean distance), and a distance matrix is produced. The components with the shortest distance between them are selected and put into a cluster together. The two rows from the DSM of the components are then averaged together to develop a coordinate for the new cluster. Then a new distance matrix is calculated with the new cluster substituted in for the two components. The shortest distances are again combined into a cluster. This continues until all of the components are combined together into one big cluster. Every time two components are combined into a cluster, the distance at which the combination occurs is recorded. These distances are then used as thresholds for when two components or clusters group together. These relationships can then be visualized by a plot called a dendrogram. A dendrogram is a hierarchical tree graph that shows the breakdown of the clusters at different distances. Figure 3 is an example of a dendrogram. On the x axis are the components contained in the DSM, and the y axis represents the distance when a group forms. It can be seen in Figure 3 that gear 4 and shaft 3, at a distance of 1, group together to form a cluster, along with gear 1 and shaft 1. As the distance metric increases, the components and clusters begin to combine into clusters containing more and more components. As shown, as the distance metric increases, in this case until a distance of about 3, all of the components combine to form one large cluster.

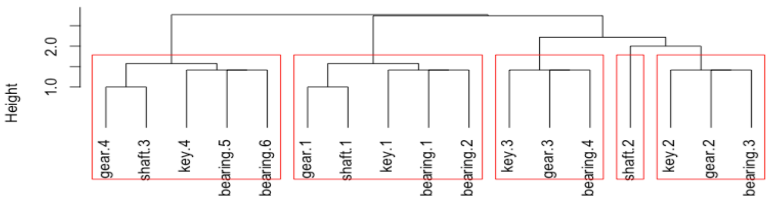


Figure 3: Example Dendrogram

Once a dendrogram is calculated, a relationship between the gears is known, but not necessarily which are or are not in the same cluster. For example, Figure 3 shows a threshold cut-off of a distance of about 1.75, which results in 5 clusters (denoted by the red boxes). However, if a cut-off of 2.5 is selected, then the last three clusters of the 1.75 cut-off would combine into one large cluster, resulting in a total of 3 clusters. Hierarchical clustering only shows the hierarchy of the data, but the algorithm does not show how to calculate the correct distance threshold. This is largely due to the "correct" threshold being largely relative to the problem. However, the order of the names in the bottom of the dendrogram is one of the most optimal ways of ordering the components of a DSM. For this case study, the total within sum of squares along with the elbow method was used to find the optimal number of clusters and thus the cut off threshold for the dendrogram.

3.2 Divisive Clustering

Divisive clustering is a type of hierarchical clustering technique. Divisive clustering techniques are largely ignored in the literature, primarily due to the computational complexity, as splitting a set of *n* observation into two non-empty subsets by considering all divisions of the set would lead to 2^{n-1}-1 possibilities [11]. The algorithm starts by splitting a cluster into two and proceeding until each observation is its own cluster. This means that the hierarchy of *n* observations is built in *n*-1 steps. A similarity or a distance measure is essential in clustering or pattern recognition. A similarity measure referred to as cosine similarity is used to obtain the affinity/similarity matrix. The cosine similarity between two vectors is a measure that calculates the cosine of the angle between the vectors. As the distance decreases, the matrix is further split until each leaf of the tree is in its own cluster.

Referring again to Figure 3, a dendrogram shows a relationship between the components, but it does not necessarily show components belonging to the same cluster. In terms of a network model, modularity can be defined as the expected fraction of edges from a randomized null model of network subtracted from the fraction of edges within the groups. Considering that a dendrogram is a graph/network represented as a tree structure, an optimization algorithm known as spectral optimization [6,12] can be used to obtain the optimal number of clusters. The algorithm is based on the approach proposed by Newman and Girvan in [12] wherein the optimal number of clusters *k* are found by maximizing the "modularity function" *Q*, in contrast to the objective functions used for other graph clustering methods. The network is divided recursively based on the signs of the eigenvector corresponding to the most positive eigenvalue of the modularity matrix [13].

3.3 Genetic Algorithm

Genetic algorithms can be used to solve a wide variety of problems. Genetic algorithms mimic reproduction in nature whereby a population of solutions produces a population of offspring, and the cycle continues as the fitness of the solution improves. In the case presented here, a solution is a clustering arrangement for a DSM and is considered to be more "fit" if it has a lower MDL. Russel and Norvig describe the basic procedure of a genetic algorithm [14]. An important consideration is the encoding of solutions to chromosomes so that more offspring can be produced. In this paper, binary variables are used to indicate membership of each component to a cluster. For example, given five components and a maximum of three clusters, one possible solution is shown in Table 1.

Table 1: Sample clustering arrangement solution

	A	B	C	D	E
Cluster 1	1	1	1	0	0
Cluster 2	0	0	0	1	1
Cluster 3	0	0	0	0	0

To encode this solution, the rows of the table are concatenated, which results in the following chromosome: 111000001100000. Thus, the length of the chromosome is the product of the number of components and the maximum number of clusters. The parameters of a genetic algorithm should be chosen carefully, as there is a significant impact on the ability of the algorithm to find a solution. Such parameters include the population size, crossover probability, mutation rate, and termination criteria. For the results shown in the next section, the population size was 100, and pairs of parent chromosomes were chosen randomly from the population without replacement to generate offspring chromosomes. Uniform crossover was used, meaning that each gene of an offspring was selected randomly from one of the two parents. The mutation rate was 0.05, meaning there was a 5% probability that each gene of an offspring would be changed from 0 to 1 or vice versa. The termination criteria were a maximum of 500 generations or 50 generations without improvement. These parameters were similar to those used in similar work found in the literature (e.g., [9]) and adjusted through trial and error.

4. Results and Analysis

The three clustering algorithms were applied to five different DSMs. The first three DSMs were from [8] (shown in Figure 1) and were chosen for their unique structure. The last two DSMs (DSM4 and DSMShaver) were larger, representing components with 17 and 53 components, respectively taken from [15]. Each of these algorithms was applied to each DSM, then the MDL and convergence rate of each algorithm were recorded and compared.
Figure 4 shows the MDL for the solutions found for each DSM. DSM1 had the simplest structure, and each of the three algorithms arrived at the same solution. DSM2 had a bus module, and DSM3 had a 3D cluster arrangement. Divisive clustering performed better for DSM3 as a result of an additional optimization algorithm used. The genetic algorithm was able to find much more optimal solutions than either hierarchical or divisive clustering for these two. For DSM4, the genetic algorithm found the best solution, followed by hierarchical, then divisive. The shaver DSM was by far the largest (53 x 53), and the genetic algorithm was unable to converge to a solution before reaching the termination criteria of 50 iterations without improvement. Since the solution space for the shaver DSM is very large, it may take more than 50 iterations to find an improvement. However, both the hierarchical and divisive clustering algorithms were able to find solutions, with the divisive having a better solution than the hierarchical clustering. Due to the large solution space, the true optimal clustering arrangement is difficult to identify. Therefor the clustering algorithms used are considered to be unsupervised methods.

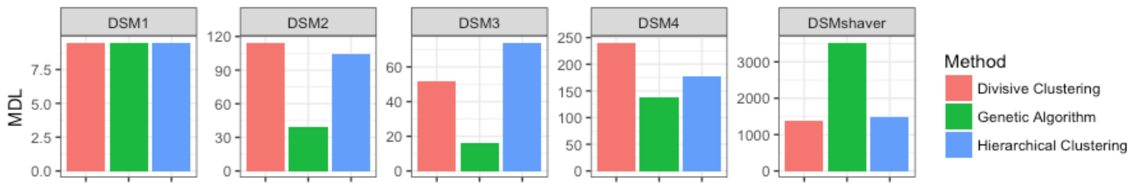


Figure 4: Algorithm Solution MDL for each DSM

The next metric used to evaluate the algorithms was the time it took to find an optimal solution. Figure 5 shows the runtime for each algorithm for every DSM. The genetic algorithm method took the longest with some of the runtimes coming close to 3 hours. Although divisive and hierarchical clustering had much lower runtimes in comparison, the divisive clustering algorithm ran between 2-23 seconds while hierarchical algorithm converged in under 0.1 seconds. This is due to hierarchical clustering only requiring the repeated calculation of distance matrices and the averaging of components in a cluster which are all relatively computationally inexpensive.

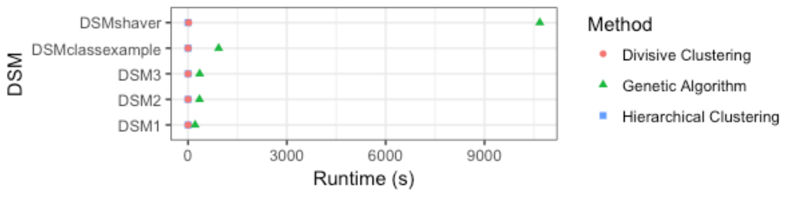


Figure 5: Algorithm Runtime for each DSM

Some interesting differences and limitations of the algorithms can be seen in the differences in the solutions that were found. For example, the optimal matrix for DSM3 is shown in in Figure 6 for the three different clustering algorithms. Since the genetic algorithm was able to identify overlapping clusters, it was able to find an extremely well-organized DSM solution when compared to hierarchical or divisive clustering.

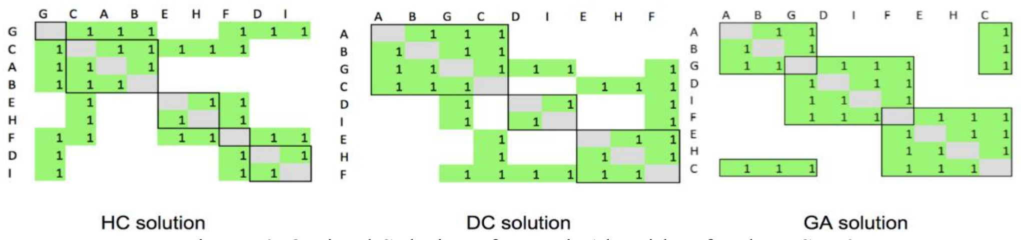


Figure 6: Optimal Solutions for Each Algorithm for the DSM 3

The DSM4 clustering arrangements (see Figure 7) illustrate how both hierarchical and divisive clustering algorithms were able to find similar clusters. For example, the first three clusters of the divisive clustering as a similar structure as the second cluster of the hierarchical cluster. Additionally, the last three clusters of the divisive clustering algorithm have a similar structure to the first cluster of the hierarchical cluster. Also, the last three clusters of hierarchical clustering are the same as the second and third-to-last cluster in the genetic algorithm solution. Again, since the genetic algorithm can have components that can identify as being a member to more than one cluster, the genetic algorithm was able to explain this relationship with only two clusters instead of the three needed for the hierarchical algorithm.

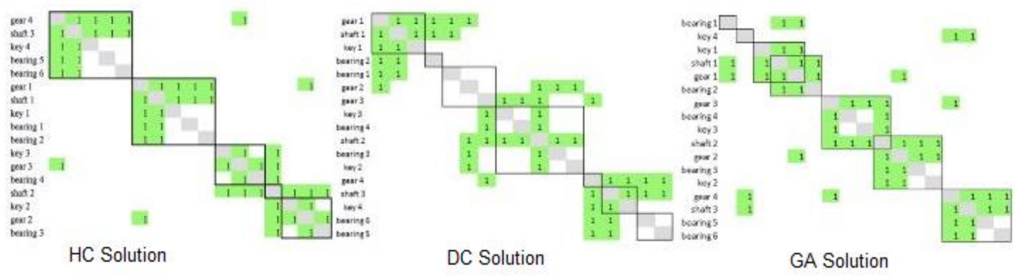


Figure 7: Optimal Solutions for Each Algorithm for DSM 4

7. Conclusion

This paper compared three different algorithms for clustering DSMs. Across most examples, the genetic algorithm produced the lowest MDL, and it was able to identify the optimum number of clusters. It was also capable of detecting bus modules, overlapping structures, and overlapping clusters. However, hierarchical and divisive clustering have the distinctive advantage over genetic algorithms, being faster and allowing for clusters that can be visualized in the form of a dendrogram, which has a multitude of distance measures available to compute differences between components. It should be noted however, that components in these methods can only be a member of one cluster and cannot overlap. This means that the clusters have less flexibility between modules and fail to identify bus modules. To identify overlapping clusters, use of soft clustering techniques like fuzzy clustering algorithm which allow cluster members to be present in more than one cluster may be helpful. Comparing hierarchical clustering to divisive clustering, it is found that it is faster, much more efficient, although divisive clustering performs better on the chosen metric, the improvement is negligible given the trade-off with computational complexity. Apart from computational complexity, the modularity maximization algorithm used to identify the optimal number of clusters uses only the leading eigenvector of the modularity matrix and ignores all the other values thus losing valuable information in the process. To identify overlapping clusters, use of soft clustering techniques like fuzzy clustering algorithm may be helpful.

References

1. T. R. Browning, “Applying the design structure matrix to system decomposition and integration problems: a review and new directions,” *IEEE Trans. Eng. Manage.*, vol. 48, no. 3, pp. 292–306, 2001.

2. D. F. Wyatt, D. C. Wynn, J. P. Jarrett, and P. John Clarkson, “Supporting product architecture design using computational design synthesis with network structure constraints,” *Res. Eng. Des.*, vol. 23, no. 1, pp. 17–52, 2011.
3. T. Simpson, A. Bobuk, L. Slingerland, S. Brennan, D. Logan, and K. Reichard, “From User Requirements to Commonality Specifications: An Integrated Approach to Product Family Design,” in *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, 2010.
4. T. Simpson, “Platform Fundamentals: Examples, Definitions, and Pitfalls,” Penn State University, 2017.
5. T.-L. Yu, A. A. Yassine, and D. E. Goldberg, “An information theoretic method for developing modular architectures using genetic algorithms,” *Res. Eng. Des.*, vol. 18, no. 2, pp. 91–109, 2007.
6. M. E. Sosa, S. D. Eppinger, and C. M. Rowles, “A Network Approach to Define Modularity of Components in Complex Products,” *J. Mech. Des.*, vol. 129, no. 11, p. 1118, 2007.
7. S. Jung and T. W. Simpson, “An Integrated Approach to Product Family Redesign Using Commonality and Variety Metrics,” in *Volume 2B: 41st Design Automation Conference*, 2015.
8. T. Simpson, “Platform Architecting: Example, Definition, Planning, Leveraging,” The Pennsylvania State University, 2017.
9. J. Rissanen, “The MDL principle, universal coding, and modeling,” in *1986 25th IEEE Conference on Decision and Control*, 1986.
10. Clark F. Olson, “Parallel algorithms for Hierarchical Clustering”, *Parallel Computing*, vol. 21(8), pp.1313-1325, 1995.
11. A. J. Scott and M. J. Symons, “297. Note: On the Edwards and Cavalli-Sforza Method of Cluster Analysis,” *Biometrics*, vol. 27, no. 1, p. 217, 1971.
12. M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Physical Review E*, vol. 69, no. 2, 2004.
13. M. E. J. Newman, “Finding community structure in networks using the eigenvectors of matrices,” *Phys. Rev. E Stat. Nonlin. Soft Matter Phys.*, vol. 74, no. 3 Pt 2, p. 036104, Sep. 2006.
14. S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Third. 2003, pp. 126–129.
15. S. Jung, “Methods and Tools for Product architecting”, The Pennsylvania State University, 2017.