A Cuckoo Search Clustering Algorithm for Design Structure Matrix

Hayam G. Wahdan, Sally S. Kassem and Hisham M. Abdelsalam
Faculty of Computers and Information, Cairo University, Cairo, Egypt

Keywords: Design Structure Matrix, Cuckoo Search, Modularity, Modular Design, Clustering, Optimization.

Abstract: Modularity is a concept that is applied to manage complex systems by breaking them down into a set of modules that are interdependent within and independent across the modules. Benefits of modularity are often achieved from module independence that allows for independent development to reduce overall lead time and to reach economies of scale due to sharing similar modules across products in a product family. The main objective of this paper is to support design products under modularity, cluster products into a set of modules or clusters, with maximum internal relationships within a given module and minimum external relationships with other modules. The product to be designed is represented in the form of a Design Structure Matrix (DSM) that contains a list of all product components and the corresponding information exchange and dependency patterns among these components. In this research Cuckoo Search (CS) optimization algorithm is used to find the optimal number of clusters and the optimal assignment of each component to specific cluster in order to minimize the total coordination cost. Results obtained showed an improved performance compared to published studies.

1 INTRODUCTION

System design involves clustering various components in a product such that the resulting modules are effective for the company. An ideal architecture is one that partitions the product into practical and useful modules. Some successfully designed modules can be easily updated on regular time cycles, some can be made in multiple levels to offer wide market variety, some can be easily removed as they stay, and some can be easily swapped to gain added functionality. The importance of effective product modularity is multiplied when identical modules are used in various different products (Aguwa et al., 2012).

Modular design approach is widely used in consumer products; machinery and software design. In response to the changing market trend of having large varieties within small production, modular design has assumed significant roles in the product development process (Gwangwava et al., 2013). The product is represented in the form of a Design Structure Matrix (DSM) that contains a list of all product components and the corresponding information exchange and dependency patterns. DSM, working as a product representation tool, provides a clear visualization of product design. The transformation of Component-DSM into proposed functional blocks of components is called Clustering. For small problems' components, a Component- DSM may be sorted manually. For larger problems, this is not practical, and at some point, computer algorithms are absolutely necessary (Borjesson and Hölttä, 2012).

The aim of this paper is to develop a cuckoo search (CS) optimization algorithm to find: (1) the optimal number of clusters in a DSM; and (2) the optimal assignment of components to each cluster. The objective function is to minimize the total coordination cost. In this context, the DSM will work as a system analysis tool that provides a compact and clear representation of a complex system. It captures the interactions/interdependencies/interfaces between system elements. It also works as a project management tool which renders a project representation that allows for feedback and cyclic activity dependencies (Abdelsalam et al., 2014).

The following sections of the paper are structured as follows. Section 2 provides a brief introduction on DSM. Section 3 reviews the literature and introduces the previous work this

Wahdan, H., Kassem, S. and Abdelsalam, H.
A Cuckoo Search Clustering Algorithm for Design Structure Matrix.
DOI: 10.5220/0005693000360043
In *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems (ICORES 2016)*, pages 36-43
ISBN: 978-989-758-171-7

research builds on. Section 4 provides the problem definition. Section 5 presents the proposed algorithm. Section 6 discusses the results obtained and, finally, Section 7 provides conclusion and ideas for future research.

2 DESIGN STRUCTURE MATRIX

The design structure matrix (DSM) is becoming a popular representation and analysis tool for system modelling, A DSM displays the relationships between components of a system or product in a compact visualization. Such a system can be, for example, product architecture or an engineering design process or a project.

The basic DSM is a simple square matrix, where *n* is the number of system elements. The DSM has *m* non-empty elements, where *m* is the number of couplings among different system elements. An example of a DSM is shown in Figure 1. Element names are placed on the left hand side of the matrix as row headings and across the top row as column headings in the same order of their execution. A common DSM assumption is that elements are undertaken in the order listed from top to bottom. An off-diagonal mark (**x**) represents dependency between two elements. If an element *i* depends on element *j,* then the matrix element *i j* (row_i , $column_j$) contains an off diagonal mark (**x**), otherwise the cell is empty (Abdelsalam and Bao, 2006).

Once the DSM for a product is constructed, it can be analyzed for identifying modules, a process referred to as clustering. The goal of DSM clustering is to find a clustering arrangement where modules minimally interact with each other while components within a module maximally interact with each other. As an example, consider the DSM shown in Figure 1(a). One can see from Figure 1(b) that the DSM is rearranged by permuting rows and columns to contain most of the interactions within two separate modules: {*A, F, E*} and {*D, B, C, G*}. However, three interactions are left out of any modules.

	A	B	C	D	E	F	G
A		x			x	x	
B				x			x
C		x		x			x
D		x	x		x		x
E				x		x	
F	x				x		
G		x	x	x			

(a)

	A	F	E	D	B	C	G
A		x	x				
F	x		x				
E		x		x			
D			x		x	x	x
B				x			x
C				x	x		x
G				x	x	x	

(b)

Figure 1: Design Structure Matrix.

3 RELATED WORK

The idea of maximizing interactions within modules and minimizing interactions between modules within a DSM was proposed by (Eppinger et al., 1994). A stochastic clustering algorithm using this principle operating on a DSM was first found in (Idicula, 1995), with subsequent improvements presented by (Gutierrez, 1998). The proposed algorithm can find clustering solutions to architecture and organization interaction problems modelled using DSM method. Gutierrez, (1998) developed a mathematical model to minimize the coordination cost to find the optimal solution for a given number of clusters. A Simulated annealing algorithm was performed by (Thebeau, 2001) to find clustered DSM with cost minimization as an objective.

Yassine et al. (2007) used the design structure matrix (DSM) to visualize the product architecture and to develop the basic building blocks required for the identification of product modules. The clustering method was based on the minimum description length (MDL) principle and a simple genetic algorithm (GA).

Borjesson (2009) proposed a method for promoting better output from the clustering algorithm used in the conceptual module generation phase by adding convergence properties, a collective reference to data identified as option properties, geometrical information, flow heuristics, and module driver compatibility.

Van Beek et al. (2010) developed a modularization scheme based on the functional model of a system. The k-means clustering was adopted for DSM based modularization by defining a proper entity representation, a relation measure and an objective function. A novel clustering method utilizing Neural Network algorithms and Design Structure Matrices (DSMs) was introduced by (Pandremenos and Chryssolouris, 2012). The algorithm aimed to cluster components in DSM with predetermined number of clusters and clustering efficiency as an objective function.

Borjesson and Hölttä (2012) used IGTA (Idicula-Gutierrez-Thebeau Algorithm) for clustering Component-DSM as the basis for their work. They provided some improvement named IGTA-plus. IGTA-plus represented a significant improvement in speed and quality of the solution obtained.

Borjesson and Sellgren (2013) presented an efficient and effective Genetic clustering algorithm, with the Minimum Description Length measure. To significantly reduce the time required for the algorithm to find a good clustering result, a

knowledge aware heuristic element is included in the GA process. The efficiency and effectiveness of the algorithm is verified with four case studies.

Yang et al. (2014) provided a systematic clustering method for organizational DSM. The proposed clustering algorithm was able to evaluate the clustering structure based on the interaction strength.

Jung and Simpson (2014) introduced simple new metrics that can be used as modularity indices bounded between 0 and 1, and also utilized as the objective functions to obtain the optimal DSM. The optimum DSM was the one with the maximized interactions within modules and the minimized interactions between modules.

Kim et al. (2015) provided a new approach for product design by integrating assembly and disassembly sequence structure planning.

We conclude from all the above that there are few techniques to cluster DSM for modularity which differ mainly in the clustering objective. Cost minimization is one of the first clustering objectives in which each DSM element is placed in an individual cluster and components are then, coordinated across modules to minimize the cost of being inside and outside a cluster. The maximum number of components in a cluster is predetermined to prevent forming large clusters. A clustered DSM can be compared to a targeted DSM topology using another objective function called Minimal Description Length (MDL). MDL finds mismatching elements between the two topologies. The objective of clustering is to minimize MDL. The number of clusters is determined a priori based on the DSM structure. Clustering Efficiency (CE) index is another clustering objective that evaluates a weighed count of zero elements inside clusters and non-zero elements outside clusters with a predefined number of clusters.

In this research, a new optimization algorithm, called the cuckoo search algorithm (CS) algorithm, is introduced for solving the clustering problem of DSM. To the best of our knowledge, this is the first time CS is used to design products under modularity with variable number of clusters, while prohibiting overlapping between clusters.

4 PROBLEM DEFINITION

The problem presented in this work considers two decision variables: (1) the number of clusters to be formed and (2) the optimal assignment of elements to each cluster. The objective function is to minimize the total coordination cost. The total coordination cost of the DSM to be clustered is based on IntraClusterCost and ExtraClusterCost as shown in equations 1 and 2,

$$\text{intraClusterCost} = \sum_{j=1}^{\text{ncluster}} (\text{ClusterSize}\,j^{\text{powcc}} \sum_{i,k \in \text{Clusterj}} (\text{DSM}_{ik} + \text{DSM}_{ki})) \quad (1)$$

$$\text{ExtraClusterCost} = \sum_{i,k \notin \text{clusterj}} (\text{DSM}_{ik} + \text{DSM}_{ki})\text{DSMSize}^{\text{powcc}}, \quad j = 1 \ldots \text{ncluster} \quad (2)$$

where DSM_{ik} is the coupling between elements i and k, DSMSize is the number of elements (rows) in the matrix, powcc is the exponent used to penalize the size of clusters, and ncluster is the total number of clusters. clustersize is the number of elements in cluster j (Borjesson and ltta, 2014).

Total coordination Cost = ExtraClusterCost + IntraClusterCost

Subject to the constraint that each element is assigned only to one cluster, in other words, overlap between clusters is not allowed. Prohibiting overlap, or multi-cluster elements, is important for the following reasons: when allowing elements to be assigned in multiple clusters, the importance and usefulness of the clustering algorithm will be diminished or eliminated. If elements exist in more than one cluster, this forces interactions between these clusters on multi levels. We would like the elements to be placed with other elements that are very similar (Pandremenos and Chryssolouris, 2012).

Modularity affects both the profit and the sustainability of the product. A modular product contains modules that can be removed and replaced. The manufacturer can develop new modules instead of entirely new products. Therefore, customers buying upgraded modules only dispose of a portion of the product, thus reducing the total amount of waste. Hence, a customer upgrading a module does not have an entirely new product.

5 PROPOSED ALGORITHM

Yang and Deb (2009) proposed a new Meta heuristic algorithm called cuckoo search (CS). They tried to simulate the behaviour of cuckoos to examine the solution space for optimization. The algorithm was

inspired by the obligate interspecific brood parasitism of some cuckoo species that lay their eggs in the nests of host birds of other species. The aim is to escape the parental investment in raising their offspring. This strategy is also useful to minimize the risk of egg loss to other species, as the cuckoos can distribute their eggs amongst a number of different nests.

Of course, sometimes it happens that the host birds discover the alien eggs in their nests. In such case, the host bird takes different responsive actions varying from throwing such eggs away, to simply leaving the nest and building a new one elsewhere. On the other hand, the brood parasites have their own sophisticated characteristics to ensure that the host birds will care for the nestlings of their parasites. Examples of these characteristics are shorter egg incubation periods, rapid nestling growth, and egg coloration or pattern mimicking their hosts (Li and Yin, 2015).

Many testing functions are used to prove the effectiveness of the algorithm, for example, Michaelwicz function, Rosenbrock's function, etc. They prove that the CS algorithm is efficient. When comparing results with existing GA and PSO's, cuckoo search performs better (Yang and Deb, 2010). Another major advantage of CS when compared to other metaheuristic algorithms, is its simplicity since it requires only two parameters. This feature reduces the effort of adjustment and fine tuning of parameter settings.

In cuckoo search, each egg can be regarded as a solution. In the initial process, each solution is generated randomly. When generating the i^{th} solution in t + 1 generation, denoted by X_i^{t+1} a levy flight is performed as shown in equation 3,

$$X_i^{t+1} = X_i^t + \alpha \oplus \text{Levy}(\lambda) \quad (3)$$

Where $\alpha > 0$ is a real number denoting the step size, which is related to the sizes of the problem of interest, and the $\oplus$ product denotes entry-wise multiplications. A Levy flight is a random walk where the step-lengths are distributed according to a heavy-tailed probability distribution as shown in equation 4.

$$\text{Levy} \sim u = t^{-\lambda}, (1 < \lambda \leq 3) \quad (4)$$

The CS algorithm is based on three idealized rules (Navimipour and Milani, 2015).

(1) Each cuckoo lays one egg at a time and dumps it in a randomly chosen nest.

(2) The best nests with high quality eggs (solutions) will be carried over to the next generations.

(3) The number of available host nests is fixed, and a host can discover an alien egg with a probability $pa \in [0, 1]$. In this case, the host bird can either throw the egg away or abandon the nest to build a completely new nest in a new location.

For simplicity, the third assumption can be approximated by a fraction pa of the n nests being replaced by new nests (with new random solutions at new locations). For a maximization problem, the quality or fitness of a solution could be proportional to the objective function. However, other more sophisticated expressions for the fitness function can also be defined.

Based on these three rules, the basic steps of the CS algorithm are summarized in the pseudo code in Figure 2.

```
Objective function f(x), x = (x1, ..., xd)^T ;
Initial population of n host nests xi (i = 1, 2, ..., n);
while (t <MaxGeneration) or (stop criterion);
Get a cuckoo (i) randomly using Levy flights;
Evaluate its quality/fitness Fi;
Choose a nest among n (j) randomly;
if (Fi > Fj),
Replace j with the new solution;
end
Abandon a fraction (pa) of worse nests
and build new ones at new locations via Levy flights;
Keep the best solutions (or nests with quality solutions);
Rank the solutions and find the current best;
end while
Postprocess results and visualisation;
```

Figure 2: pseudo code of CS (Yildiz, 2013).

5.1 Solution Representation

The CS algorithm will be used to solve the problem defined in Section 4. Solution representation of the problem is a vector of size equals to the number of elements in the DSM. Each cell in the vector takes an integer value between 1 and the number of clusters, as show in Figure 3. The vector in Figure 3 with size 7 represents a solution, where the DSM Contains 7 elements. Elements 1 and 7 belong to cluster 1, elements 2, 3, 4 belong to cluster 2, and elements 5, 6 belong to cluster 3. This solution representation forces the element to be a member of only one cluster.

1	2	2	2	3	3	1

Figure 3: Solution representation vector.

Assume that we start with the maximum possible number of clusters, which equals to the number of elements in the DSM. The next step is to try to find the optimal number of clusters after deleting empty clusters. Such representation of the problem will not allow multi-clustering, which means each element will be assigned to only one cluster.

This problem will be solved using Cuckoo search algorithm (CS). CS solves continuous types of variables. Since the problem in hand is categorized as a discrete variable problem, the solutions should be converted from continuous to discrete. This is done by the discretization of the continuous space by transforming the values into a limited number of possible states. There are several discretization methods available in the literature, for example: random key technique is used to transform from continues space to discrete integer space, to decode the position, the nodes are visited in ascending order for each dimension (Chen et al., 2011). The smallest position value (SPV) method maps the positions of the solution vector by placing the index of the lowest valued component as the first item on a permutated solution, the next lowest as the second, and so on (Verma and Kumar, 2012). The nearest integer method is another technique, to transform continuous variables to integer variables. In the nearest integer method, a real value is converted to the nearest integer (NI) by rounding or truncating up or down (Burnwal and Deb, 2012).

Considering the above mentioned methods, SPV, and random key methods, are not suitable for the problem presented in this work. This is because integer value(s) need to be repeated, while these methods result in unique values. Therefore, the suitable method for the problem in hand is the nearest integer method since it allows the repetition of values by truncating to the higher or lower value.

In CS, we start with a set of nests; each nest is a vector of length that equals to the number of elements in the DSM. This vector contains random numbers following uniform distribution in the range from lower and upper limits, these random numbers are converted to integer values using the nearest integer method. Each one of these integer numbers represent a solution that could be sent for the evaluation function. The evaluation function returns the total coordination cost.

5.2 Solution Evaluation

The total coordination cost of the DSM to be clustered is based on IntraClusterCost and ExtraClusterCost as explained in section 4. Regarding intracluster cost, if interaction DSM_{ik} belongs to cluster j, then calculate intra cluster cost. On the other hand, if interaction DSM_{ik} does not belong to cluster j, calculate the extra cluster cost. The first step in calculating the total coordination cost is to start with the total number of interactions in the DSM multiplied by the size of the DSM raised to the power powcc. This is the highest value of total coordination. This value will be minimized in subsequent steps of the algorithm after forming clusters. After completion of the evaluation step, select the best solution and go to the next best solution using Levy flight carrying the best nests with high quality of eggs (solutions) over to the next generations. Continue till the stopping condition is reached.

6 EXPERIMENTAL RESULTS AND ANALYSIS

In this section we examine the CS algorithm on different problems. We use 2 small size and 1 large size problem. The first small size problem has a DSM that contains 7 elements as shown in figure 4.The DSM starts initially with a total coordination cost of 68. This cost is based on assigning each element in it is own cluster. No clusters are formed yet, and powcc=0.65.

	1	2	3	4	5	6	7
1		1	0	0	1	1	0
2	0		0	1	0	0	1
3	0	1		1	0	0	1
4	0	1	1		1	0	1
5	0	0	0	1		1	0
6	1	0	0	0	1		0
7	0	1	1	1	0	0	

Figure 4: Original DSM.

After applying the CS clustering algorithm, the clustered DSM is as shown in Figure 5. The Total coordination cost is reduced to 48. Thebeau (2001) solved the same problem and obtained a total coordination cost of 53, hence, our proposed CS is able to obtain better results. The minimum number of clusters using the proposed CS algorithm is 2. Figure 6 shows the total cost as it changes with every iteration. The best solution is obtained in iteration number 575. The CPU run time ranges

from 0.07 seconds to 0.39 seconds for 100 to 1000 iterations, respectively.

It is noticed that, in the clustered DSM two clusters are formed and most interactions are included in clusters. This means that, similar elements are grouped in the same cluster. In this case intracost is larger than the extracost which improves the objective function value. Only 3 interactions are placed outside clusters (number of 1's).

	1	5	6	2	3	4	7
1		1	1	1	0	0	0
5	0		1	0	0	1	0
6	1	1		0	0	0	0
2	0	0	0		0	1	1
3	0	0	0	1		1	1
4	0	1	0	1	1		1
7	0	0	0	1	1	1	

Figure 5: Clustered DSM.

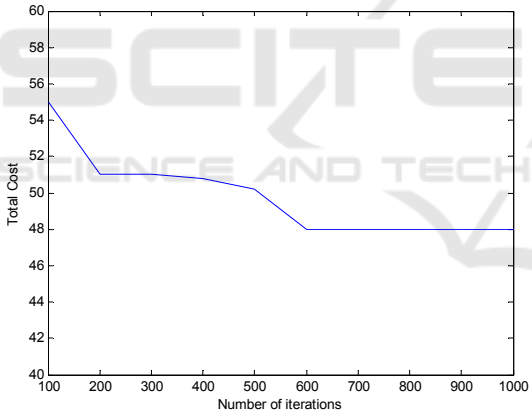

Figure 6: Cost history for CS algorithm-best solution.

We examined the developed algorithm on another problem presented in (Yassine et al., 2007). The DSM of the problem has 9 elements as shown in Figure 7.

Figure 8 shows the clustered DSM after using CS algorithm. The total coordination cost after clustering with CS is 41.8. The corresponding number of clusters is 4. The resulting DSM clustered using our proposed CS algorithm is the same as the one obtained in (Yassine et al., 2007). Figure 9 shows the total cost as it changes with every iteration. The best solution is obtained in iteration number 880. The CPU run time ranges from 1.07 seconds to 7.52 seconds for 100 to 1000 iterations, respectively.

	A	B	C	D	E	F	G	H	I
A	1	0	0	0	1	0	1	0	0
B	0	1	1	0	0	1	0	1	0
C	0	1	1	0	0	1	0	1	0
D	0	0	0	1	0	0	0	0	0
E	1	0	0	0	1	0	1	0	0
F	0	1	1	0	0	1	0	1	0
G	1	0	0	0	1	0	1	0	0
H	0	1	1	0	0	1	0	1	0
I	0	0	0	0	0	0	0	0	1

Figure 7: Original DSM.

We notice from Figure 8 that, in the clustered DSM four clusters are formed, cluster 1 with the most similar 3 elements , cluster 2 with the most similar 4 elements, cluster 3 with 1 element and cluster 4 with 1 element. All interactions are included in clusters. In this case, there are no extra costs because no 1's are outside clusters.

	A	E	G	B	C	F	H	D	I
A		1	1						
E	1		1						
G	1	1							
B					1	1	1		
C				1		1	1		
F				1	1		1		
H				1	1	1			
D									
I									

Figure 8: Clustered DSM.

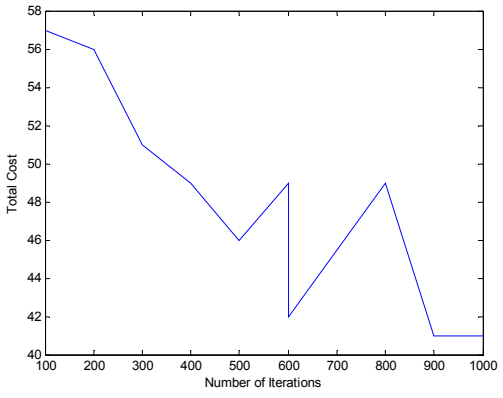

Figure 9: Cost history for CS algorithm-best solution.

To further evaluate the proposed CS algorithm we apply it on a large size problem, available in (Thebeau, 2001) . The DSM contains 61 elements and represents an elevator example. The total coordination cost obtained using the CS algorithm is 4133.25, with a total number of 17 clusters. The total coordination cost obtained in (Thebeau, 2001) is 4433. Accordingly, our proposed CS algorithm is able to obtain superior results when compared to the results obtained by (Thebeau, 2001). Cluster assignments of the elevator example using the CS algorithm are shown in Table 1. The best solution is obtained in iteration number 680. The CPU run time 5461.7 seconds after 1000 iterations.

Figure 10 shows the total cost as it changes with every iteration.

Table 1: results obtained using CS algorithm for the elevator example.

Cluster Number	Elements that cluster contains
1	1,3,5,11,15,17,18,20,22,28,34,35,37,39,40,41,43,47,48,50,59,60,61
2	2,8,12,16,19,21,26,27,32,33,44,46,49,54
3	4
4	6,9,13
5	7
6	10,14,25,55
7	20
8	22,53
9	23,31
10	24
11	30
12	36
13	51
14	52
15	56
16	57
17	58

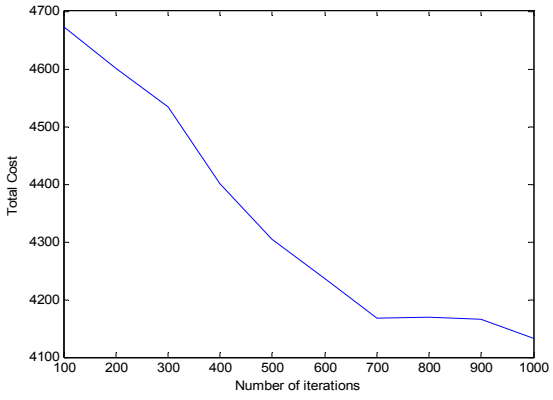

Figure 10: Cost history for CS algorithm–best solution.

7 CONCLUSION AND FUTURE WORK

The design of products under modularity is a problem that captured the attention of many researchers. One method to perform modular product design is through representation and clustering of a DSM. Clustering of a DSM in this work requires solving for 2 decision variables: the number of clusters to form, and the assignment of elements to each cluster. The objective function is to minimize the total coordination cost, subject to one constraint, namely, assigning each element to one cluster and prohibiting clusters' overlap. To perform the clustering of DSM we employed cuckoo search (CS) algorithm. The CS algorithm has proved its efficiency in solving many problems in terms of simplicity, speed, and solution quality. We applied the CS algorithm on a number of DSM test problems available in the literature. Results show that the proposed CS obtained superior or similar results to those available in the literature. Future work includes developing a model that restricts the number of elements within each cluster, incorporating sustainability concepts, and consider the number of clusters as part of the objective function.

REFERENCES

Abdelsalam, H. M., Rasmy, M. H., & Mohamed, H. G. (2014). A Simulation-Based Time Reduction Approach for Resource Constrained Design Structure Matrix. *International Journal of Modeling and Optimization , 4* (1), 51-55.

Abdelsalam, H., & Bao, H. (2006). A Simulation-based Optimization Framework for Product Development Cycle Time Reduction. *IEEE Transactions on Engineering Management , 53* (1), 69-85.

Aguwa, C. C., Monplaisir, L., & Sylajakumar, P. A. (2012). Effect of Rating Modification on a Fuzzy-Based Modular Architecture for Medical Device Design and Development. *Advances in Fuzzy Systems* .

Borjesson, F., & Hölttä-Otto, K. (2012). Improved clustering algolrithm for design structure matrix. *ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference* (pp. 1-10). Chicago, IL, USA: IDETC/CIE 2012.

Borjesson, F., & ltta-Otto, K. H. (2014). A module generation algorithm for product architecture based on component interactions and strategic drivers. *Research in Engineering Design , 25* (1), 31-51.

Borjesson, F., & Sellgren, U. (2013). Fast Hybrid Genetic Clustering Algorithm for Design Structure Matrix.

25th International Conference on Design Theory and Methodology. Portland, Oregon, USA: ASME 2013.

Borjesson, F. (2009). Improved output in modular function deployment using heuristics. *International conferance on engineering design*, (pp. 24-27). Stanford,USA.

Burnwal, S., & Deb, S. (2012). Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. *The International Journal of Advanced Manufacturing Technology, 64*, 1-9.

Chen, H., Li, S., & Tang, Z. (2011). Hybrid gravitational search algorithm with random-key encoding scheme combined with simulated annealing. *International Journal of Computer Science and Mobile Computing*, *11* (6), 208-217.

Eppinger, S., Whitney, D., Smith, R., & Gebala, D. (1994). A model based method for organizing tasks in product development. *Research in Engineering Design* , 1–13.

Gutierrez, C. I. (1998). *Integration analysis of product architecture to support effective team co-location.* Cambridge: Masters thesis, Massachusetts Institute of Technology.

Gwangwava, N., Nyadongo, S., Mathe, C., & Mpof, K. (2013). Modular Clusterization Product Design Support System. *International Journal of Advances in Computer Science and Technology (IJACST) , 2* (11), 8-13.

Idicula, J. (1995). *Planning for concurrent engineering.* Singapore: Gintic Institute Research .

Jung, S., & Simpson, T. W. (2014). A Clustering Method Using New Modularity Indices and Genetic Algorithm with Extended Chromosomes. *DSM 14 Proceedings of the 16th International DSM conference: Risk and Change management in complex systems*, (pp. 167-176).

Kim, S., Baek, J. W., Moon, S. K., & Jeon, S. M. (2015). A New Approach for Product Design by Integrating Assembly and Disassembly Sequence Structure Planning. 247-257.

Li, X., & Yin, M. (2015). Modified cuckoo search algorithm with self adaptive parameter method. *Information Sciences , 298*, 80–97.

Navimipour, N. J., & Milani, F. S. (2015). Task Scheduling in the Cloud Computing Based on the Cuckoo Search Algorithm. *International Journal of Modeling and Optimization , 5* (1), 44-47.

Pandremenos, J., & Chryssolouris, G. (2012). A neural network approach for the development of modular product architectures. *International Journal of Computer Integrated Manufacturing*, 1-8.

Thebeau, R. (2001). *Knowledge management of system interfaces and interactions for product development process.* Massachusetts Institute of Technology.

van Beek, T. J., Erden, M. S., & Tomiyama, T. (2010). Modular design of mechatronic systems with function modeling. *Mechatronics , 20* (8), 850–863.

Verma, R., & Kumar, S. (2012). DNA sequence assembly using continuous particle swarm optimization with smallest position value rule. *First International Conference on Recent Advances in Information Technology*, (pp. 410-415).

Yang, Q., Yao, T., Lu, T., & Zhang, B. (2014). An Overlapping-Based Design Structure Matrix for Measuring Interaction Strength and Clustering Analysis in Product Development Project. *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT , 61* (1), 159-170.

Yang, X., & Deb, S. (2009). Cuckoo search via Levy flights. *the World Congress on Nature and Biologically Inspired Computing (NABIC '09)* (pp. 210–214). Coimbatore, India: IEEE.

Yang, X.-s., & Deb, S. (2010). Engineering Optimisation by Cuckoo Search. *International Journal of Math Model Numerical Optimization*, *1* (4), 330-343.

Yassine, A. A., Yu, T.-L., , & Goldberg, D. E. (2007). An information theoretic method for developing modular architectures using genetic algorithms. *Research in Engineering Design , 18*, 91-109.

Yildiz, A. R. (2013). Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *international Journal of Advanced Manufacturing Technology , 64* (1), 55-61.