Optimization Modeling for Clustering a Design Structure Matrix

Chamath Gunawardena

University of Wisconsin-Whitewater

Abstract

Design Structure Matrix (DSM) is a two-dimensional matrix representation of the structural or functional interrelationships of objects, tasks or teams. Used primarily in engineering management, DSM is increasingly being applied to complex issues in health care management, financial systems, public policy, natural sciences, and social systems. A DSM can also be treated as an adjacency matrix of a directed graph in which the edges represent the dependencies. A DSM is used to identify program complexity and inherent relationships as well as more proactive task sequence managing. A main part of the DSM model is to cluster the tasks in the DSM with a clustering which results in a block lower triangular matrix (i.e., no cycles among blocks) so that each task within a tightly coupled cluster of iterative tasks creates rework for the other linked tasks. The path searching algorithm found in the literature is a clustering algorithm that involves tracing information flow backwards or forwards until a task is encountered twice to identify a cycle of information flow. When all cycles have been identified, the tasks have scheduled or permuted in the matrix signifying the end of the sequencing and resulting in a matrix in block triangular form. Another clustering algorithm is the Powers of the Adjacency Matrix Method which involves a binary DSM that is raised to the n-th power showing tasks that may be involved in a n-step cycle. We implement DSM sequencing algorithm using mixed integer linear and quadratic models which are used to find optimal clusters. We implemented our optimization models on GAMS/CPLEX. The clustering objective functions for DSM analysis trades off minimizing the interactions outside clusters and maximizing the size of the clusters. The linear model provides a global optimal solution while the quadratic model provides a local optimal solution.

1 Introduction

Design Structure Matrix (DSM) is a two-dimensional matrix representation of the structural or functional interrelationships of objects, tasks or teams. Used primarily in engineering management, DSM is increasingly being applied to complex issues in health care management, financial systems, public policy, natural sciences, and social systems. The DSM can be used to identify orderings of tasks and find difficult aspects of the design process [6]. Clustering objective functions for DSM analysis trades off two conflicting goals: (1) minimize the (number and strength of) interactions outside clusters, and (2) maximize the size of the clusters. The following example is taken from [1].

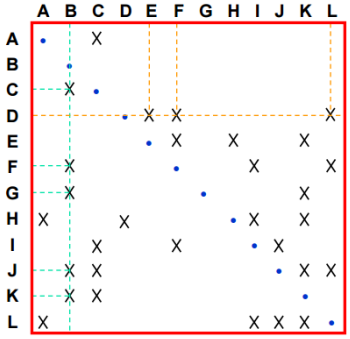

Figure 1: Information Exchange Model [1]

2 Sequencing a DSM

We have a DSM with the tasks labeled alphabetically representing a corresponding row and column. The Xs on the graph represent the dependency of on task to another, i.e., the X at row A, column C represents that task A depends on task C in order to finish.

We can permute this DSM so that the Xs are coupled with each other in different sub matrices (clusters). By doing this tasks are switched around so that they are next to the tasks they depend on which can show us an order of execution of all the tasks. In order to do this we must create a GAMS model on finding a block triangular matrix (i.e, each block represents a cluster) and minimizing the number of X s in the upper part of the resulting block triangular matrix (i.e., Any cluster of tasks will not depend on the tasks of the

future clusters).

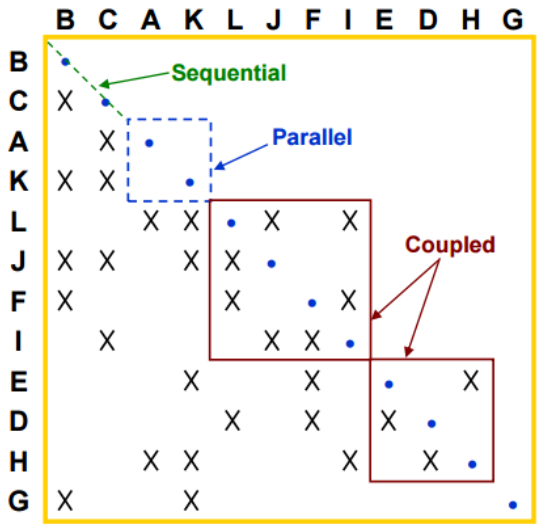

Figure 2: Partitioned or Sequenced DSM [1]

2.1 Other Sequencing Algorithms

2.1.1 Path Searching

Path searching deals with finding loops between tasks where a task may depend on a another task that depends on the first task resulting in a loop between them. If a task t_1 does not depend on any other task and no other task depends on t_1, then the row and column corresponding to t_1 is removed from the DSM. If a task t_2 depends on other tasks but no other task depends on t_2 then the row and column corresponding to t_2 are permuted to the edges of the DSM. If certain tasks are part of a dependency loop, then their rows andd columns are combined to make one row and one column [5].

Figure 3(a) shows a DSM and Figure 3(b) shows the final partitioned matrix after applying the Path Search Algorithm on the original DSM in Figure 3.

2.1.2 Powers of the Adjacency Matrix Method

The Adjacency matrix is a binary DSM where dependencies between tasks mark as '1' and no dependency between tasks mark as '0'. This method involves raising the DSM to the n-th power where we can determine which task can be

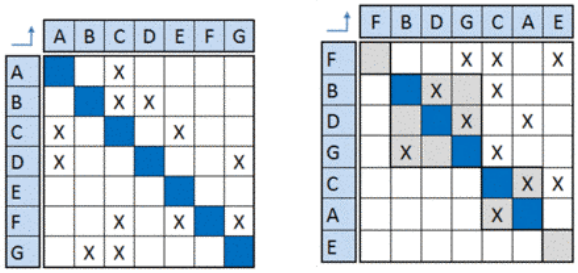

Figure 3: (a) Original DSM and (b) Path Searched DSM [4]

reached from itself in n steps by looking at a non-zero entry for that task in the diagonal of the matrix [4].

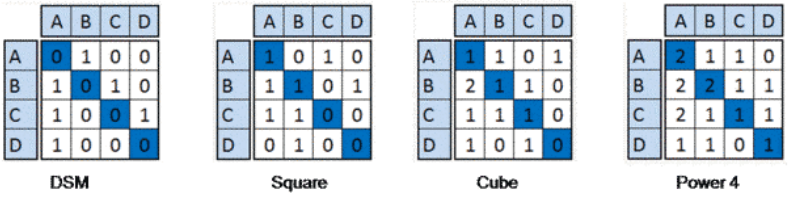

Figure 4: Powers of Adjacency Matrix Method [4]

Figure 4 shows an example of raising the DSM up to the power of 4 where squaring the matrix shows us that there is a feedback loop between task A and B. However, cubing the matrix shows tasks A, B, and C in a feedback loop. This method helps determining whether feedback loops exist within the DSM.

3 Models

Let BDSM be an n x n binary design structure matrix. First we create linear and quadratic models to find a principal submatrix A of BDSM such that the total number of 1s in A, $|A|$, is a maximum. Using a permutation matrix P, we can permute the rows of BDSM so that A can be in the upper left hand corner as shown in Figure 5. At the same time, we want to minimize the number of 1s in the rows of BDSM corresponding to A and to the right of A (i.e., the number of 1s in B, $|B|$). We also want to maximize the number of 1s in the columns of BDSM corresponding to A and below A (i.e., the number of 1s in C, $|C|$). Our

optimization model is:

$$\begin{gathered}\text{Maximize } |A| - |B| + |C| \\ \text{subject to} \\ P^T \times BDSM \times P = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \\ \text{for all permutation matrices } P.\end{gathered}$$

Figure 5: Matrix Permutation

For a given size, our linear and quadratic models with a linear combination of the above objectives find an optimal principal submatrix that corresponds to a cluster. We use these clusters iteratively (the next iteration will start with D) to sequence the Design Structure Matrix to achieve a block lower triangular matrix.

3.1 Linear Model

Let x(i),a(i,j),b(i,j),c(i,j) be binary variables such that (1 ≤ i,j ≤ n). Assume we have sequenced BDSM up to nextIndex – 1 where nextIndex (NI) is the beginning of the next cluster.We remove the first NI-1 rows and columns by imposing the following constraint:

$$x(i) = 0 \text{ for } 1 \leq i \leq NI - 1 \tag{1}$$

We are going to find an optimal partition as given in Figure 5 for the submatrix BDSM(NI,. . .,n , NI,. . .,n) = B_{NI}. We fix the size of the block A to be $(size \times size)$ where A contains the most optimal rows and columns from the row, column indices $NI, \ldots, n$.

$$\sum_{i=NI}^{NI+size} x(i) = size \tag{2}$$

We capture the submatrix A corresponding to the indices where $a(i,j) = 1$ by following three constraints:

$$\sum_{i=NI}^{n} \sum_{j=NI}^{n} a(i,j) = size^2 \tag{3}$$

$$a(i,j) \leq x(i) \tag{4}$$

$$a(i,j) \leq x(j) \tag{5}$$

We construct the submatrix B corresponding to the indices where $b(i,j) = 1$ with the following three constraints:

$$\sum_{i=NI}^{n} \sum_{j=NI}^{n} b(i,j) = size * (n - NI - size + 1) \tag{6}$$

$$b(i,j) \leq x(i) \tag{7}$$

$$b(i,j) \leq 1 - x(j) \tag{8}$$

Similarly, we construct the submatrix C corresponding to the indices where $c(i,j) = 1$ with the following three constraints:

$$\sum_{i=NI}^{n} \sum_{j=NI}^{n} c(i,j) = (n - NI - size + 1) * size \tag{9}$$

$$c(i,j) \leq 1 - x(i) \tag{10}$$

$$c(i,j) \leq x(j) \tag{11}$$

Finally, we maximize our objective function subject to the above constraints as follows.

$$\text{Maximize} \qquad \lambda_1|A| - \lambda_2|B| + \lambda_3|C| \tag{12}$$

where

$$|A| = \sum_{i=1}^{n} \sum_{j=1}^{n} BDSM(i,j)a(i,j) \tag{13}$$

$$|B| = \sum_{i=NI}^{n} \sum_{j=1}^{n} BDSM(i,j)b(i,j) \tag{14}$$

$$|C| = \sum_{i=NI}^{n} \sum_{j=1}^{n} BDSM(i,j)c(i,j) \tag{15}$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1 \tag{16}$$

3.2 Quadratic Model

Let $x(i)$ be binary variables as in Section 3.1. We can construct a quadratic mixed integer optimization model for sequencing the DSM as follows.

$$\text{Maximize} \qquad \lambda_1|A| - \lambda_2|B| + \lambda_3|C| \tag{17}$$

$$|A| = \sum_{i=1}^{n} \sum_{j=1}^{n} BDSM(i,j)x(i)x(j) \tag{18}$$

$$|B| = \sum_{i=NI}^{n} \sum_{j=1}^{n} BDSM(i,j)x(i)(1-x(j)) \tag{19}$$

$$|C| = \sum_{i=NI}^{n} \sum_{j=1}^{n} BDSM(i,j)(1-x(i))x(j) \tag{20}$$

such that

$$\sum_{i=NI}^{n} x(i) = size \tag{21}$$

4 Sequencing Algorithm

Our sequencing algorithm can use any of the above models for DSM sequencing. It goes through n iterations and at the kth iteration the algorithm has sequenced the previous k-1 rows and columns. At the kth iteration, it calls one of the above models with size equaling 1 to find an optimal row and column to create the new permuted BDSM. The GAMS implementation of the algorithm can be found in [2].

5 Results

For this we have implemented mixed integer linear and quadratic models to find optimal clusters using GAMS (General Algebraic Modeling System) [2]. GAMS is a high-level modeling system for mathematical programming and optimization. GAMS is used for large scale modeling applications and can be used to solve linear, mixed integer, quadratic and nonlinear optimization problems [3] with efficient optimization solvers. The linear model provides a global optimal solution while the quadratic model provides a local optimal solution.

Figure 6 shows an example of the original 12x12 Design Structure Matrix given in [1] used as the input for the algorithm for both linear and quadratic versions.

Figure 7 shows the resulting sequenced matrix written to the text file created in the linear version. It has one 2x2 parallel block, 4x4 and 3x3 coupled blocks. We received a similar sequencing for the quadratic version which is shown in

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	1	0	0	0	0	0	1
5	0	0	0	0	0	1	0	1	0	0	1	0
6	0	1	0	0	0	0	0	0	1	0	0	1
7	0	1	0	0	0	0	0	0	0	0	1	0
8	1	0	0	1	0	0	0	0	1	0	1	0
9	0	0	1	0	0	1	0	0	0	1	0	0
10	0	1	1	0	0	0	0	0	0	0	1	1
11	0	1	1	0	0	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	1	1	1	0

Figure 6: Original Design Structure Matrix

	2	3	11	1	7	10	9	12	6	4	8	5
2	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0
11	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0
7	1	0	1	0	0	0	0	0	0	0	0	0
10	1	1	1	0	0	0	0	1	0	0	0	0
9	0	1	0	0	0	1	0	0	1	0	0	0
12	0	0	1	1	0	1	1	0	0	0	0	0
6	1	0	0	0	0	0	1	1	0	0	0	0
4	0	0	0	0	0	0	0	1	1	0	0	1
8	0	0	1	1	0	0	1	0	0	1	0	0
5	0	0	1	0	0	0	0	0	1	0	1	0

Figure 7: Resulting Sequenced Matrix from Mixed Integer Linear Model

figure 6. But the two solutions are different as shown in the index permutations. If we take the linear solution, task 2, 3, and 11 must be performed sequentially. Task 1 and 7 can go in parallel. The block 10, 9, 12, 6 has coupled tasks. The block 4, 8, 5 also has coupled tasks.

	2	3	11	1	7	12	10	6	9	8	5	4
2	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0
11	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0
7	1	0	1	0	0	0	0	0	0	0	0	0
12	0	0	1	1	0	0	1	0	1	0	0	0
10	1	1	1	0	0	1	0	0	0	0	0	0
6	1	0	0	0	0	1	0	0	1	0	0	0
9	0	1	0	0	0	0	1	1	0	0	0	0
8	0	0	1	1	0	0	0	0	1	0	0	1
5	0	0	1	0	0	0	0	1	0	1	0	0
4	0	0	0	0	0	1	0	1	0	0	1	0

Figure 8: Resulting Sequenced Matrix from Mixed Integer Quadratic Model

We also have tried a 34x34 matrix given in [7]. We have noticed that the linear version gave a better sequencing for this case. This may be due to locally optimal solutions given by the quadratic model instead of the global solution

given by the linear model. Although we used binary DSM matrices, the same algorithm can be used for real valued matrices. In the future we would like to compare the performance of our linear and quadratic models to the existing DSM sequencing algorithms [5, 4].

References

[1] Design structure matrix. https://ocw.mit.edu/courses/engineering-systems-division/esd-36-system-project-management-fall-2012/lecture-notes/MITESD_36F12_Lec04.pdf.

[2] Gams files from github repository. https://github.com/chamathg21/DSMGAMS.

[3] An introduction to gams. https://www.gams.com/products/introduction/.

[4] Sequencing a dsm. https://dsmweb.org/sequencing-a-dsm/.

[5] E. Gebala. Methods for analyzing design procedures. In *Proceedings of 3rd International ASME Conference on Design Theory and Methodology*, page 227:223, 1991.

[6] S. E. Robert Smith. Identifying controlling features of engineering design iteration. In *Management Science*, page 276, 1997.

[7] E. S. Thomas Black, Charles Fine. A method for systems design using precedence relationships: An application to automotive brake systems. 1990.