**INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09**
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

# A REVISED TWO-PHASE METHOD FOR DECOMPOSITION OF DESIGN PROBLEMS

**Simon Li**
Concordia University, Canada

## ABSTRACT

The two-phase decomposition method has two major components: dependency analysis and partitioning analysis, which offer necessary sub-functions for problem decomposition [1]. While dependency analysis focuses on analyzing the dependency structure of a given design problem, partitioning analysis utilizes the results from dependency analysis to identify design sub-problems and their interaction. In the original version of the two-phase method, only two types of coupling in a design problem are considered: coupling between any two parameters and coupling between any two functions. This methodological arrangement has overlooked the coupling between a design function and a design parameter. Thus, this paper proposes a revision of the two-phase method by considering this type of coupling. The revised two-phase method uses the approach of coupling matrix concatenation to unify different types of coupling, and it can simplify and expedite the original decomposition process. The examples of the relief valve and powertrain systems are used to demonstrate the utility of the revised method.

*Keywords: problem decomposition, clustering analysis, matrix-based decomposition*

## 1 INTRODUCTION

### 1.1 Matrix-based Problem Decomposition

Consider a design problem involving $n$ parameters to be configured subject to the satisfaction of $m$ functions. To capture the dependency relationships among these parameters and functions, a rectangular matrix is applied in which the matrix's rows (symbolized $r_i$) and columns (symbolized $c_j$) represent the design problem's functions and parameters, respectively. The value of each matrix entry (symbolized $m_{ij}$) can be either *positive* (indicating the non-zero degree of dependency) or *zero* (indicating the absence of dependency). Mathematically, the rectangular matrix (symbolized as $M$) representing a design problem can be defined as

$$M = [m_{ij}], (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n) \quad (1)$$

Thus, each matrix entry ($m_{ij}$) indicates the strength of dependency between the function corresponding to row $r_i$ and the parameter corresponding to column $c_j$.

Given a rectangular matrix to represent a design problem, problem decomposition is performed to transform an unorganized matrix into a block-angular matrix that, in general, consists of blocks and an interaction part. The blocks represent the sub-problems identified by decomposition, and the interaction part accounts for the coordination imposed on the sub-problems.

In literature, various algorithms have been proposed to decompose a rectangular matrix in order to show the decomposed blocks and their interaction. Initial efforts can be found in the development of matrix-based algorithms to partition a system of equations [2], [3]. In the context of group technology, various matrix-based algorithms have been proposed to find the machine-part matching, including the rank-order clustering algorithm [4], the clustering identification algorithm [5], [6], and the branch-and-bound algorithm [7]. In addition, other formal approaches have been proposed to address problem decomposition specifically, such as the bond energy algorithm [8], the network reliability approach [9], the hypergraph approach [10], and the integer programming approach [11].

Among these matrix-based decomposition methods, the two-phase method is unique since it applies clustering analysis for decomposition. The benefits of the two-phase method can be described in two

major aspects. First, this method decouples the function of matrix-based decomposition into two sub-functions handled by two method components: dependency analysis and partitioning analysis. The function of dependency analysis focuses on probing and analyzing the given dependency structure of the design problem. In contrast, the function of partitioning analysis utilizes the information from dependency analysis to partition the design problem according to some decomposition criteria (such as the number of sub-problems, the size of interactions, etc). This functional decoupling facilitates an expedited means for re-decomposition by simply changing decomposition criteria without resort to dependency analysis [1]. Thus, the two-phase method is flexible to explore different decomposition solutions.

Second, as the core technique of the two-phase method is clustering analysis, its methodological framework is designed as an open architecture, which allows researchers to articulate and reason how to perform matrix-based decomposition systematically [12]. Also, this framework allows us to explore different decomposition-related concepts, such as decomposability and complexity [13]. In the next sub-section, we will provide an overview of the two-phase method.

### 1.2 Overview of the Two-Phase Method

The workflow of the two-phase method is illustrated in Figure 1. The function of the two-phase method is to partition the original design problem, which is represented by a rectangular matrix, into a set of linked sub-problems in the form of block-angular matrix. Since the elements of the input matrix are disorderedly scattered, Phase 1 of the method, (namely, dependency analysis) is invoked to re-arrange the matrix's rows and columns such that the highly-coupled rows and columns are brought close to each other. As a result, a banded diagonal matrix is formed, which is the input of Phase 2 (see Figure 1b). In Phase 2, partitioning analysis is invoked to transform the banded diagonal matrix into a block-angular matrix according to some decomposition criteria, such as the number of blocks and the size of interaction. As a result, the block-angular matrix indicates decomposed blocks and their interaction (see Figure 1c).

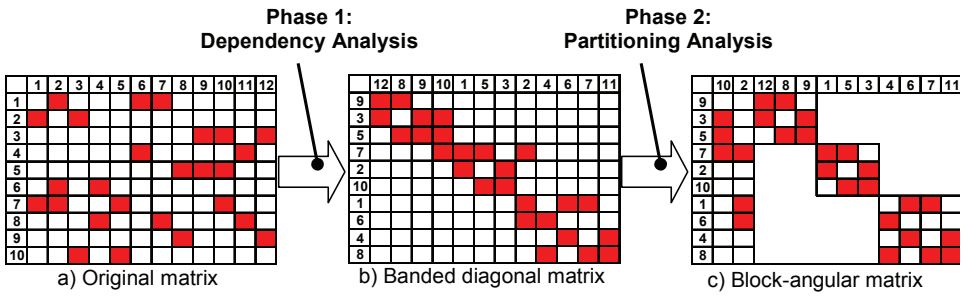



*Figure 1. Workflow of the two-phase decomposition method*

The foundation of dependency analysis (Phase 1) is hierarchical cluster analysis (HCA), which provides a systematic approach to classify objects by comparing their attributes [14]. In the context of problem decomposition, two types of objects need to be clustered according to their coupling information: functions and parameters. Two functions are said coupled when they both depend on the same parameters. Two parameters are said coupled when they both affect the same function(s). For instance, consider the sample matrix in Figure 1a. Functions (rows) #3 and #5 are coupled since they depend on Parameters (columns) #9 and #10. Similarly, Parameters (columns) #1 and #2 are coupled since they both affect Function (row) #7. Such coupling analysis allows us to quantify the coupling values between any two rows and between any two columns.

Adapting the HCA techniques, a row tree and a column tree (or dendrograms) can be constructed to present a nested structure to group the objects, as shown in Figure 2a & 2b for the sample matrix. The tree's leaves are labeled to represent the corresponding rows or columns (e.g., r9 represents Row #9). Given a tree structure, each branch represents a set of rows or columns. For instance, the top left branch of the row tree in Figure 2a shows that Functions (rows) #9, #3 and #5 should be put in one group, while the others should be put in another group. Inspecting the branches further down can reveal more groupings in a nested structure.

However, the matrix-based problem decomposition is different from the problems traditionally addressed by HCA since it requires *simultaneous clustering* of rows and columns. By simply handling rows and columns separately does not yield the banded diagonal matrix. For instance, suppose that a row tree and a column tree are generated separately for the sample matrix, and their results are shown in Figure 2a & 2b. Based on the tree information, the rows and columns of the original matrix can be re-arranged based on the sequences of the leave labels. Figure 2c shows the re-arranged matrix, which only presents scattered clusters. Thus, three subsequent algorithms have been developed for Phase 1 (namely, branch node sequencing, tree branch association, and tree association). The purpose of these algorithms is to compact the formed clusters and align these clusters along the diagonal direction [15].
Given the banded diagonal matrix as the input, Phase 2 is invoked to partition the matrix by considering the decomposition criteria. The core of matrix partitioning is the systematic placement of partition points to the diagonal matrix to define the boundary of the blocks and identify the interactions among blocks. Our past research has proposed the enumerative approach (which identifies all block-angular matrices that satisfy decomposition criteria) [1] and the heuristic approach (which quick identifies a satisfactory decomposition solution) [16].

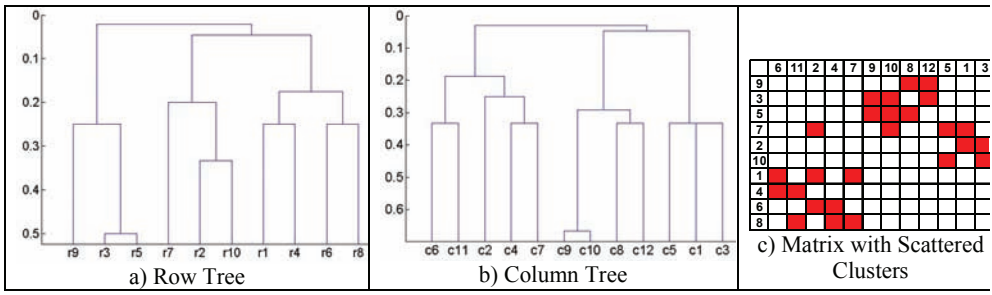



*Figure 2. Illustration of trees and the sample matrix with scattered clusters*

### 1.3 Paper's Motivation and Outline

As discussed before, the two-phase method offers a novel framework to partition a design problem systematically. However, one deficient area has been found in the algorithmic design. As the tree construction algorithm only focuses on the coupling measures between rows and between columns to form clusters, the grouping according to row couplings does not necessarily correspond to the grouping according to column couplings. As a result, the clusters after the tree construction algorithm are often scattered and subsequently three more algorithms are required in Phase 1 to compact and align the clusters to form the banded diagonal matrix.
Furthermore, even the row and column trees are obtained after Phase 1, the given tree information does not often suggest a proper way to partition the diagonal matrix. The reason behind is also due to the lack of the coupling measure between rows and columns. As a result, a partition point search method (both enumerative and heuristic) is required to yield decomposition solutions.
Therefore, it is motivated to derive a coupling measure between a row and a column in the context of problem decomposition. This effort is expected to simplify the Phase 1 procedure by eliminating three subsequent algorithms. Also, the tree information can be greatly utilized to partition a design problem. Then, any partition point search method can be viewed as an assistant tool (rather than necessity) to yield decomposition solutions.
The rest of the paper is organized as follows. Section 2 will introduce the revised two-phase method with the detailed description of methodological components. Section 3 will test the revised method via two case studies: relief valve system and powertrain system. Section 4 will provide the closing remarks of this paper.

## 2 REVISION OF THE TWO-PHASE METHOD

### 2.1 Coupling Matrix Concatenation

In the original version of dependency analysis (Phase 1), we have developed a rational foundation to evaluate the coupling of two elements of the same type. That is, we have studied the coupling of any

two functions (rows) and any two parameters (columns). If two rows (or columns) are said highly coupled, they would have a high chance to be grouped in the same block after decomposition. Accordingly, a coupling matrix is employed to record the coupling values of any two objects. As such, two types of coupling matrix have been classified: row coupling matrix (which records coupling values between rows) and column coupling matrix (which records coupling values between columns).
To measure the coupling between rows and between columns, the min/max coefficient (symbolized as $R_{min/max}$) has been derived, which utility has been justified in [16]. The formulations of the min/max coefficient for rows and columns are given in (2) and (3) as follows:

$$R_{\min/\max}(r_i, r_j) = \frac{\sum_{k=1}^{n} \min(m_{ik}, m_{jk})}{\sum_{k=1}^{n} \max(m_{ik}, m_{jk})} \qquad i, j \in [1, m] \tag{2}$$

$$R_{\min/\max}(c_i, c_j) = \frac{\sum_{k=1}^{m} \min(m_{ki}, m_{kj})}{\sum_{k=1}^{m} \max(m_{ki}, m_{kj})} \qquad i, j \in [1, n] \tag{3}$$

where $m$ and $n$ denote the numbers of rows and columns, respectively. For the sample matrix in Figure 1, the corresponding row and column coupling matrices are given in Figure 3a & 3b.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | - | 0 | 0 | 0.25 | 0 | 0.25 | 0.17 | 0.2 | 0 | 0 |
| 2 | 0 | - | 0 | 0 | 0 | 0 | 0.2 | 0 | 0 | 0.33 |
| 3 | 0 | 0 | - | 0 | 0.5 | 0 | 0.17 | 0 | 0.25 | 0 |
| 4 | 0.25 | 0 | 0 | - | 0 | 0 | 0 | 0.25 | 0 | 0 |
| 5 | 0 | 0 | 0.5 | 0 | - | 0 | 0.17 | 0 | 0.25 | 0 |
| 6 | 0.25 | 0 | 0 | 0 | 0 | - | 0.2 | 0.25 | 0 | 0 |
| 7 | 0.17 | 0.2 | 0.17 | 0 | 0.17 | 0.2 | - | 0 | 0 | 0.2 |
| 8 | 0.2 | 0 | 0 | 0.25 | 0 | 0.25 | 0 | - | 0 | 0 |
| 9 | 0 | 0 | 0.25 | 0 | 0.25 | 0 | 0 | 0 | - | 0 |
| 10 | 0 | 0.33 | 0 | 0 | 0 | 0 | 0.2 | 0 | 0 | - |

a) Row coupling matrix ($CM_r$)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | - | 0.25 | 0.33 | 0 | 0.33 | 0 | 0 | 0 | 0 | 0.25 | 0 | 0 |
| 2 | 0.25 | - | 0 | 0.25 | 0.25 | 0.25 | 0.25 | 0 | 0 | 0.2 | 0 | 0 |
| 3 | 0.33 | 0 | - | 0 | 0.33 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0.25 | 0 | - | 0 | 0 | 0.33 | 0 | 0 | 0 | 0.33 | 0 |
| 5 | 0.33 | 0.25 | 0.33 | 0 | - | 0 | 0 | 0 | 0 | 0.25 | 0 | 0 |
| 6 | 0 | 0.25 | 0 | 0 | 0 | - | 0.33 | 0 | 0 | 0 | 0.33 | 0 |
| 7 | 0 | 0.25 | 0 | 0.33 | 0 | 0.33 | - | 0 | 0 | 0 | 0.33 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - | 0.33 | 0.25 | 0 | 0.33 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.33 | - | 0.67 | 0 | 0.33 |
| 10 | 0.25 | 0.2 | 0 | 0 | 0.25 | 0 | 0 | 0.25 | 0.67 | - | 0 | 0.25 |
| 11 | 0 | 0 | 0 | 0.33 | 0 | 0.33 | 0.33 | 0 | 0 | 0 | - | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.33 | 0.33 | 0.25 | 0 | - |

b) Column coupling matrix ($CM_c$)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0.33 | 0 | 0 | 0 | 0.4 | 0.4 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0.5 | 0 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.4 | 0.33 | 0 | 0.4 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0 | 0 | 0 | 0 | 0.5 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.4 | 0.4 | 0.33 | 0 | 0 |
| 6 | 0 | 0.4 | 0 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0.33 | 0.29 | 0 | 0 | 0.33 | 0 | 0 | 0 | 0 | 0.29 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0.4 | 0 | 0 | 0.4 | 0 | 0 | 0 | 0.4 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.5 | 0 | 0 | 0 | 0.5 |
| 10 | 0 | 0 | 0.5 | 0 | 0.5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

c) Row-column coupling matrix ($CM_{rc}$)

*Figure 3. Coupling matrices of the sample matrix*

However, the above coupling analysis does not explicitly consider the coupling between a row and a column, and such coupling *does* affect the formation of blocks during decomposition. The omission of such coupling can be used to explain why we need subsequent algorithms after tree construction to align the clusters in a proper way. In the following, we will discuss how to measure the coupling between a row and a column.
Consider the sample matrix in Figure 1. The 1st row and the 1st column are not coupled because the corresponding matrix entry is zero. In contrast, the 1st row and the 2nd column are coupled due to the presence of the non-zero element at (1,2). Then, the 1st row and the 2nd column may be grouped in one

block after matrix partitioning. At this point, we want to quantify the corresponding coupling value so that we can study how *likely* this row and this column are grouped in one block after matrix partitioning.

In this context, let us consider the non-zero element at (1,2) (1st row, 2nd column). The 1st row has three non-zero elements. Besides the 2nd column, the 1st row may be grouped with the 6th and 7th columns. Similarly, besides the 1st row, the 2nd column may be grouped with the 6th and 7th rows. For comparison, consider the non-zero element at (2,1) (2nd row, 1st column). The 2nd row may be grouped with the 1st and 3rd columns, and the 1st column may be grouped with the 2nd and 7th rows only. Since the 2nd row and 1st column have *less* choice to be combined with other columns and rows respectively, the coupling between them (compared with the 1st row and the 2nd column) is *higher*.

Accordingly, the likelihood of the 1st row and the 2nd column being grouped in one block is weakened by other non-zero elements along the corresponding rows and columns. Let $M = [m_{ij}]$ be the input matrix. The coupling between the $i$th row and the $j$th column ($a_{ij}$) is quantified as follows.

$$a_{ij} = \frac{2m_{ij}}{\sum_{k=1}^{m} m_{ik} + \sum_{k=1}^{n} m_{kj}} \tag{4}$$

The value of $a_{ij}$ is bounded between zero (no coupling) and one (the $i$th row and the $j$th column have to be grouped in the same block). Then, we can get three *coupling matrices* to record dependency values between two rows ($CM_r$), between two columns ($CM_c$), and between a row and a column ($CM_{rc}$). Using the sample matrix in Figure 1, these three types of matrices are provided in Figure 3. Then, we can construct a square, symmetric $CM$ (i.e., concatenated coupling matrix) that combines these three types of coupling matrices as follows.

$$CM = \begin{bmatrix} w_c \cdot CM_c & w_{rc} \cdot CM_{rc}^{T} \\ w_{rc} \cdot CM_{rc} & w_r \cdot CM_r \end{bmatrix} \tag{5}$$

This combined $CM$ can be used to construct a tree using the existing tree construction algorithm. Figure 4a shows the constructed tree of the sample matrix based on $CM$. Based on the sequence of leaf labels of the tree in Figure 4a, we can obtain the diagonal matrix, as shown in Figure 4b.

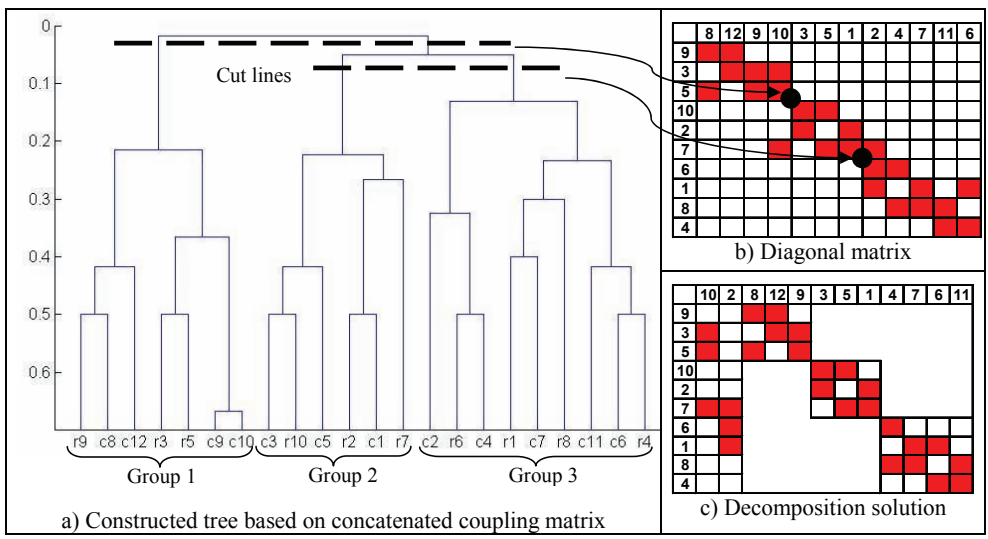



*Figure 4. Tree and diagonal matrix according to the concatenated coupling matrix*

## 2.2 Revised Tree Construction Algorithm

Through coupling matrix concatenation, a banded diagonal matrix can be obtained after the tree construction algorithm without resort to the subsequent algorithms. This is the original contribution of this paper. By referencing the original tree construction algorithm [1], the procedure of the revised tree construction algorithm is provided below, along with the sample matrix for illustration.

Step 1: Evaluation of coupling

Compute the coupling between rows according to Equation (2), the coupling between columns according to Equation (3), and the coupling between rows and columns according to Equation (4). The coupling matrices of the sample matrix have been recorded in Figure 3.

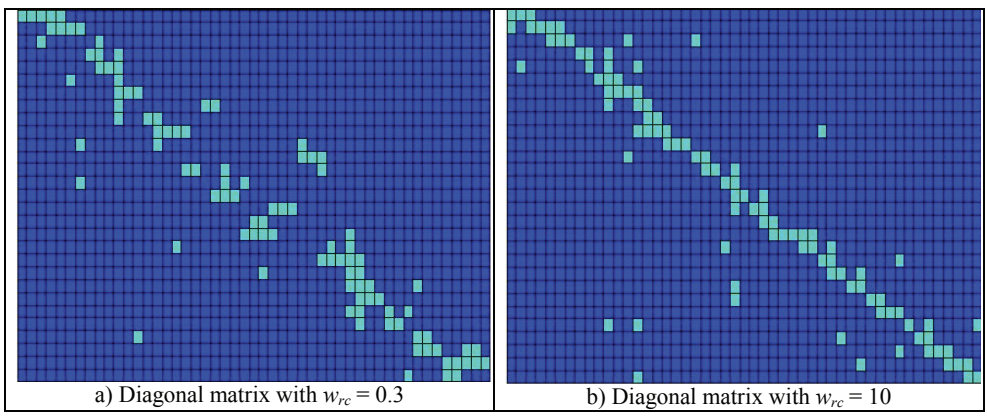
Step 2: Concatenation of coupling matrices

Construct the concatenated coupling matrix according to Equation (5). Assign the weights. In this paper, we set $w_r = w_c = 1$, and set $w_{rc}$ according to the criterion formulated below.

$$w_{rc} \geq \frac{\frac{1}{2}\left(\sum CM_r / 2m + \sum CM_c / 2n\right)}{\sum CM_{rc} / m+n} \tag{6}$$

This criterion essentially indicates that the weighted average coupling of $CM_{rc}$ should be larger than or equal to the average coupling of both rows ($CM_r$) and columns ($CM_c$).

It has been briefly observed that the weight assignment will affect the shape of the banded diagonal matrix. In general, if a heavier weight is assigned for $CM_{rc}$, more non-zero matrix entries are gathered along the diagonal with few of them *remotely* scattered from the diagonal. Figure 5 shows two diagonal matrices, which are obtained according to two different weights. Figure 5a represents the light weight assignment for $w_{rc}$ so that the non-zero matrix entries are somehow scattered around the main diagonal. For the case of heavy weight assignment for $w_{rc}$, Figure 5b shows that most of the non-zero matrix entries are gathered along the diagonal with few entries scattered far from the diagonal. It should be mentioned that how diagonal shapes affect the final decomposition solutions is still an open research question. The criterion formulated in (6) works fine with our examples.



a) Diagonal matrix with $w_{rc} = 0.3$ b) Diagonal matrix with $w_{rc} = 10$

*Figure 5. Two diagonal matrices based on two different weights*

Step 3: Tree branch construction

Pick two objects that yield the highest value from the concatenated coupling matrix (*CM*). Then, label the leaves of the tree according to the picked objects. Form the branch of the tree by combining the leaves (or branches). The vertical axis of the tree is labeled with the coupling values. The leaves (or

branches) are merged to form a new branch at their coupling value. For instance, the highest coupling value in Figure 3 is the coupling between Columns #9 and #10, which value is 0.67. Accordingly, their leaves are combined into a branch at 0.67 according to the vertical axis, as shown in Figure 4a.

Step 4: Coupling matrix update
Modify the coupling matrix (*CM*) to represent the newly formed branch. Such modification is achieved through the average distance formulation as follows [1]:

$$r_{(ij)k} = \frac{r_{ik} + r_{jk}}{2} \qquad 1 \le k \le n \quad k \ne i \quad k \ne j \tag{7}$$

where $r_{ik}$ is the coupling value between the *i*th and *k*th objects in *CM*, and the subscript *ij* refers to the newly combined branch.

Step 5: Iteration check
Repeat Step 3 and Step 4 until the concatenated coupling matrix (*CM*) cannot be further reduced.

The resulting tree of the sample matrix is obtained and shown in Figure 4a. The label order at the bottom of the final tree basically indicates the order of rows and columns. Re-arrange the rectangular matrix using this order. A banded diagonal matrix can be obtained accordingly. The resulting banded diagonal matrix of the sample matrix according to the constructed tree is shown in Figure 4b.

### 2.3 Tree-based Partitioning Analysis

The benefit of considering the coupling between rows and columns are not just the avoidance of subsequent algorithms after tree construction to form a banded diagonal matrix. Also, it allows valuable tree information to facilitate matrix partitioning. The resulting algorithm can yield decomposition solutions quicker and of better quality.
The resulting tree after dependency analysis essentially captures a nested structure, reflecting how objects should be clustered according to their coupling relationships. For instance, according to the resulting tree of the sample matrix (i.e., Figure 4a), the top branch reflects the composition of two clusters. By cutting the top branch as indicated in Figure 4a, two clusters of objects are identified. These two clusters in facts represent a partition point on a diagonal matrix, as shown in Figure 4b. If three clusters are desired, we can subsequently cut the second top branch, which can lead to three branches, indicating the composition of three clusters (or groups).
Based on the property of the tree after dependency analysis, we can consider that each branch corresponds to a partition point on the banded diagonal matrix. By checking the tree's branches from top to bottom, we can generate a list of partition points with priority sequence. The partition point according to the top branch represents a good location to divide the banded diagonal matrix based on the coupling information. For the sample matrix, the three-block decomposition solution according to tree-cutting is shown in Figure 4c, which is considered a good decomposition solution.
In addition to the coupling information, the matrix partitioning algorithm should also be capable to generate decomposition solutions according to some criteria specified by the users, such as the size of blocks and interaction. Given below is the algorithm of matrix partitioning, which utilizes the tree-based coupling information.

Step 1: Specification of decomposition criteria
The decomposition criteria considered in the algorithm include the number of blocks, the maximum size of the interaction (number of interaction columns), and the minimum size of the block.

Step 2: Collection of partition points
By breaking the branches of the final tree from top to down, collect the corresponding partition points in the same order, which indicates the priority of applying the partition points. For instance, the top two branches of the sample tree correspond to two partition points, as shown in Figure 4b.

Step 3: Application of partition points

Apply partition points subsequently to the banded diagonal matrix according to their priority order. Each time when a partition point is applied, we check the tentative solution with the decomposition criteria. If the tentative solution satisfies the criteria, we keep the point as part of the solution. Otherwise, we discard the point and search for the next point on the priority list.

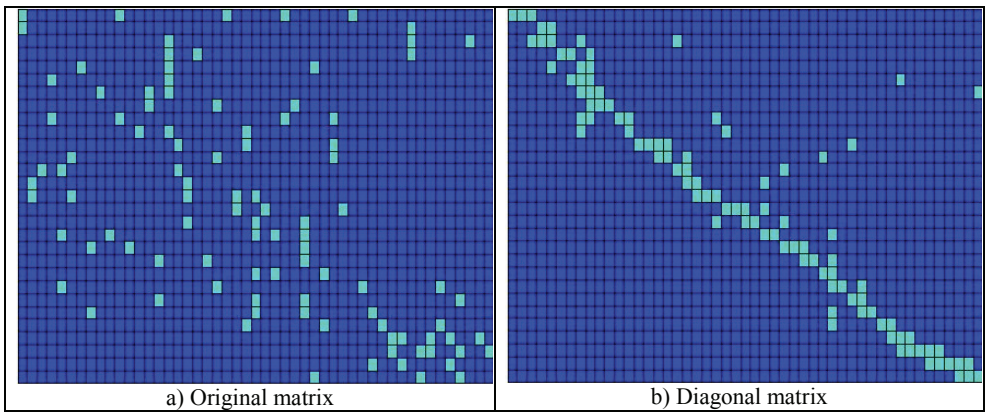
In the past, two versions of matrix partitioning have been proposed: enumerative approach [1] and heuristic approach [16]. Comparatively, the proposed tree-based matrix partitioning algorithm has several benefits. First of all, the tree-based matrix partitioning is able to "short-list" a set of good partition points, which can be applied subsequently to search for desirable decomposition solutions. This practice can largely reduce the search space, as compared to the enumerative approach. Also, the tree-based approach has directly utilized the tree-based coupling information from dependency analysis, whereas the heuristic approach needs to "re-analyze" the coupling information from dependency analysis to collect a set of good partition points. The solution quality according to the tree-based partitioning approach will be illustrated in Section 3.

## 3 CASE STUDIES AND COMPARISON

The purpose of this section is to illustrate and justify the proposed coupling matrix concatenation and the corresponding tree-based approach for matrix partitioning. For comparison purpose, two examples found in matrix-based decomposition literature are used: relief valve system and powertrain system.

### 3.1 Relief Valve System

The relief valve system is adapted from [17], which consists of 49 parameters and 29 functions. Figure 6a shows the rectangular matrix that captures the dependency relationships among these parameters (represented in columns) and functions (represented in rows). In this application, a two-phase decomposition method is used to divide the original system into sub-systems for design purpose. In dependency analysis, the revised tree construction algorithm according to coupling matrix concatenation is applied. The resulting diagonal matrix is obtained and shown in Figure 6b. As seen, the revised tree construction algorithm is able to yield a banded diagonal matrix without resort to the subsequently algorithms that are present in the previous version of dependency analysis.



a) Original matrix b) Diagonal matrix

*Figure 6. Original matrix and diagonal matrix of the relief valve system*

In Phase 2, the tree-based matrix partitioning is applied. For the decomposition criteria, the minimum block size is set as 4 columns, and the maximum interaction size is 20 columns. Note that these criteria are loosely set. If these loosely-set criteria are applied for the enumerative partitioning approach, the computational time will become very long (e.g., several hours). In contrast, the computational time of the tree-based approach is relatively short (e.g., several seconds) and is not sensitive to the decomposition criteria. We will search the decomposition solutions with different numbers of blocks (from two to six blocks). For comparison, the decomposition solutions via the enumerative approach are also obtained.

To estimate the quality of a decomposition solution, the matrix-based complexity metric is used [13]. This metric approximates the complexity entailed in a block-angular matrix by measuring the size of each block and the size of an interaction part. In general, a good decomposition solution has a low complexity value, which is obtained due to smaller blocks and less interaction among them. In contrast, a decomposition solution with larger blocks and larger interaction part will lead to a higher complexity value. Further details of the matrix-based complexity metric (e.g., formulation and justification) can be found in [13].

As a result, the complexity values of the decomposition solutions via different approaches are tabulated in Table 1. The images of decomposition solutions (from two blocks to six blocks) obtained from the revised two-phase method are shown in Figure 7. As seen from Table 1, the revised two-phase method can generally yield the solutions that have a lower complexity value compared to the solutions obtained by the previous approach. This justifies the utility of the revised two-phase method.

*Table 1. Comparison of decomposition solutions of the relief valve system*

| | Previous Approach | | Revised Approach | |
|---|---|---|---|---|
| Solution Type | Complexity value | # of interaction columns | Complexity value | # of interaction columns |
| Two-block | 0.6995 | 6 | 0.6502 | 5 |
| Three-block | 0.5799 | 9 | 0.4954 | 8 |
| Four-block | 0.5250 | 10 | 0.4912 | 10 |
| Five-block | 0.5208 | 12 | 0.4659 | 11 |
| Six-block | 0.4961 | 13 | 0.4497 | 13 |

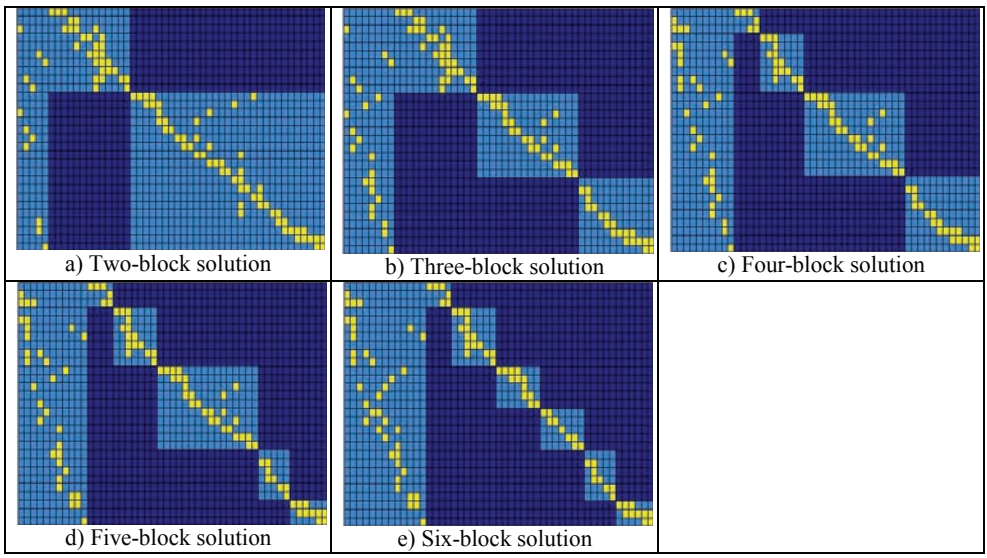


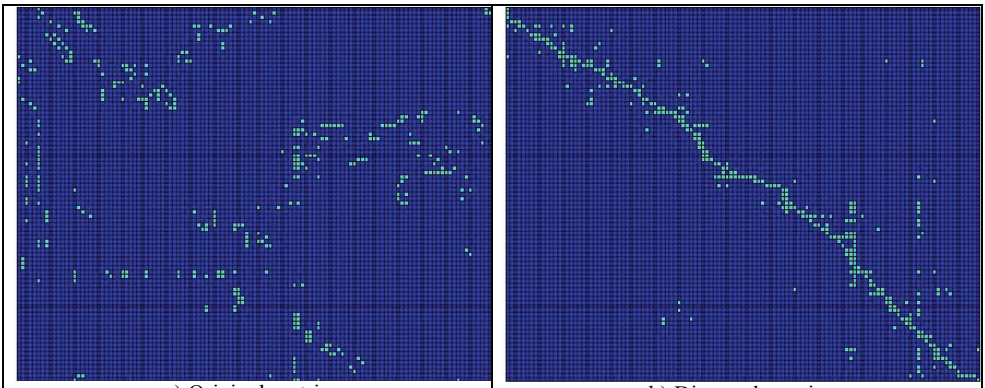

*Figure 7. Decomposition solutions of the relief valve system*

### 3.2 Powertrain System

The powertrain system, adapted from [18], is another test problem to justify the utility of the revised two-phase method. Figure 8a shows the rectangular matrix that captures the dependency relationships among 119 parameters (represented in columns) and 87 functions (represented in rows) in the powertrain system. Again, in this example, a two-phase decomposition method is used to divide the original system into sub-systems for design purpose.

In dependency analysis, the revised tree construction algorithm is applied. The resulting diagonal matrix is obtained and shown in Figure 8b. Given this diagonal matrix, the tree-based matrix

partitioning is applied. For the decomposition criteria, the minimum block size is set as 4 columns, and the maximum interaction size is 50 columns.
Accordingly, we generate the decomposition solutions starting with two blocks. Then we seek for a new decomposition solution by incrementing the required number of blocks by one until we cannot get a viable solution based on the given criteria. The comparison of complexity values and the images of actual solutions are shown in Table 2 and Figure 9, respectively. Again, it has been observed that the revised two-phase method can generally yield the solutions with lower complexity values.



a) Original matrix b) Diagonal matrix

*Figure 8. Original matrix and diagonal matrix of the powertrain system*

## 4 CLOSING REMARKS

In this paper, we have simplified the two-phase decomposition method via the approach of coupling matrix concatenation. The results from the relief valve and powertrain examples support that the new approach can provide better decomposition solutions comparatively based on the complexity measure. In addition, the new approach leads to a faster decomposition process due to the simplified dependency analysis and the effective tree-based partitioning analysis.
Directly related to the approach of coupling matrix concatenation, the next step of the research is to investigate the relationship between the weight assignment for Equation (5) and the quality of final decomposition solutions. Then, the corresponding decomposition method can be more flexible and robust to generate solutions for different needs of systems decomposition. Also, software development is to be pursued to implement the proposed decomposition method as a planning tool for engineering design.

*Table 2. Comparison of decomposition solutions of the powertrain system*

| | Previous Approach | | Revised Approach | |
|---|---|---|---|---|
| Solution Type | Complexity value | # of interaction columns | Complexity value | # of interaction columns |
| Two-block | 0.5349 | 5 | 0.5500 | 6 |
| Three-block | 0.4705 | 7 | 0.4985 | 14 |
| Four-block | 0.4100 | 15 | 0.4321 | 20 |
| Five-block | 0.4071 | 20 | 0.3710 | 24 |
| Six-block | 0.4018 | 27 | 0.3567 | 25 |
| Seven-block | 0.4405 | 34 | 0.4037 | 32 |
| Eight-block | N/A | N/A | 0.4117 | 35 |
| Nine-block | N/A | N/A | 0.4136 | 36 |
| Ten-block | N/A | N/A | 0.4236 | 38 |
| Eleven-block | N/A | N/A | 0.4466 | 41 |
| Twelve-block | N/A | N/A | 0.4531 | 43 |

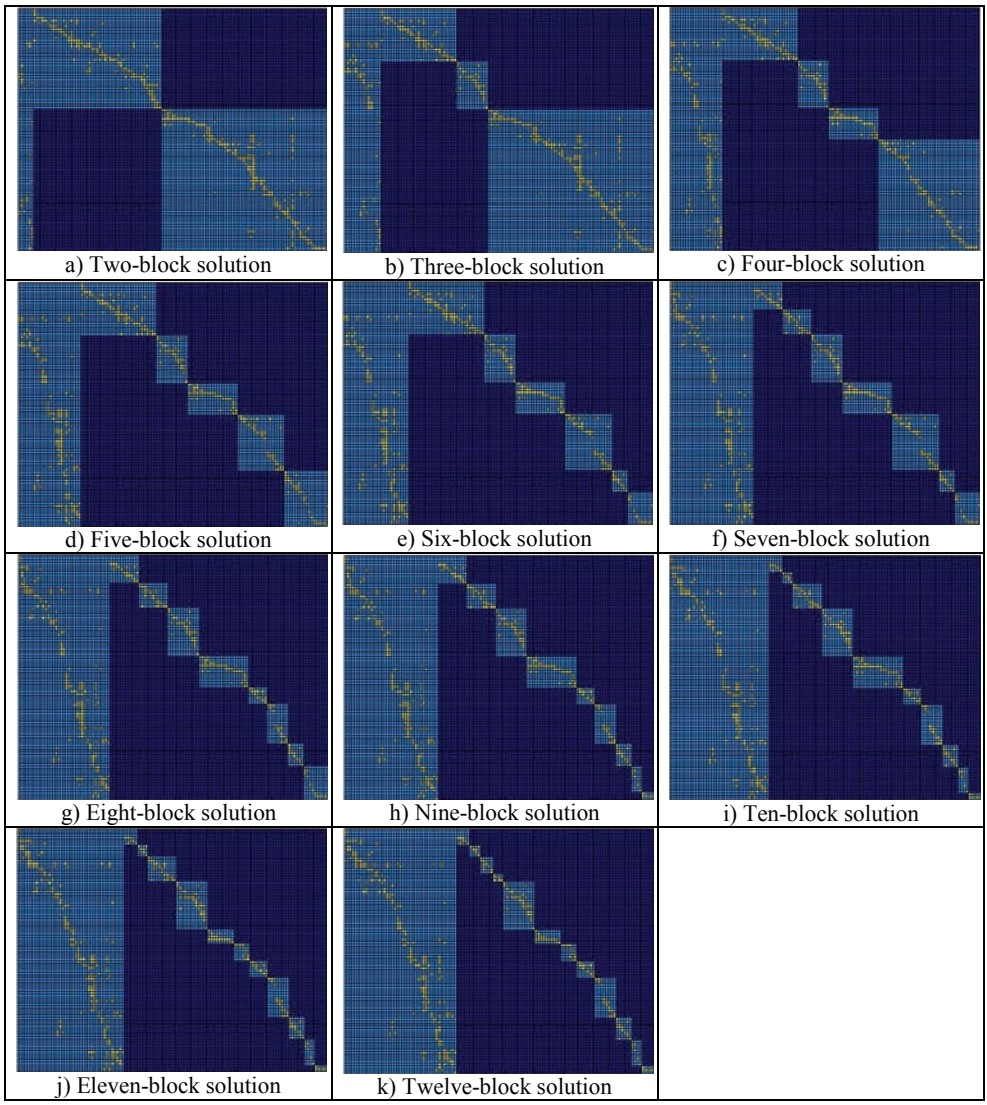



*Figure 9. Decomposition solution of the powertrain system*

## REFERENCES


[1] Chen, L., Ding, Z. and Li, S., A Formal Two-Phase Method for Decomposition of Complex Design Problems, *ASME Journal of Mechanical Design*, 2005, 127(2), pp. 184-195.

[2] Steward, D.V., Partitioning and Tearing Systems of Equations, *Journal of the SIAM: Series B, Numerical Analysis*, 1965, 2(2), pp. 345-365.

[3] Weil, R.L. and Kettler, P.C., Rearranging Matrices to Block-Angular Form for Decomposition (and other) Algorithms, *Management Science*, 1971, 18(1), pp. 98-108.

[4] King, J.R., Machine-Component Group in Production Flow Analysis: An Approach using a Rank Order Clustering Algorithm, *International Journal of Production Research*, 1980, 18(2), pp. 213-232.

[5] Kusiak, A. and Chow, W.S., Efficient Solving of the Group Technology Problem, *Journal of Manufacturing Systems*, 1987, 6(2), pp. 117-124.

[6] Kusiak, A., *Computational Intelligence in Design and Manufacturing*, 2000 (John Wiley & Sons, New York).
[7] Kusiak, A. and Cheng, C.H., A Branch-and-Bound Algorithm for Solving the Group Technology Problem, *Annals of Operations Research*, 26, 1990, pp. 415-431.
[8] McCormick, Jr., W.T., Schweitzer, P.J., and White, T.W., Problem Decomposition and Data Reorganization by a Clustering Technique, *Operations Research*, 1972, 20(5), pp. 993-1009.
[9] Michelena, N.F. and Papalambros, P.Y., A Network Reliability Approach to Optimal Decomposition of Design Problems, *ASME Journal of Mechanical Design*, 1995, 117(3), pp. 433-440.
[10] Michelena, N.F. and Papalambros, P.Y., A Hypergraph Framework for Optimal Model-Based Decomposition of Design Problems, *Computational Optimization and Applications*, 1997, 8(2), pp. 173-196.
[11] Krishnamachari, R.S. and Papalambros, P.Y., Optimal Hierarchical Decomposition Synthesis Using Integer Programming, *ASME Journal of Mechanical Design*, 1997, 119(4), pp. 440-447.
[12] Li, S. and Chen, L., Decomposition-based Analysis Framework for Complex Product Development, in *CD-ROM Proceedings of ASME Design Engineering Technical Conferences*, DETC2008/49542, New York City, New York, Aug. 3-6, 2008 (ASME, New York).
[13] Chen, L. and Li, S., Analysis of Decomposability and Complexity for Design Problems in the Context of Decomposition, *ASME Journal of Mechanical Design*, 2005, 127(4), pp. 545-557.
[14] Romesburg, H.C., *Cluster Analysis for Researchers*, 1990 (Robert E. Krieger, Malabar).
[15] Chen, L., Ding, Z. and Li, S., Tree-based Dependency Analysis in Decomposition and Re-decomposition of Complex Design Problems, *ASME Journal of Mechanical Design*, 2005, 127(1), pp. 12-23.
[16] Li, S. and Chen, L., Model-based Decomposition using Non-binary Dependency Analysis and Heuristic Partitioning Analysis, in *CD-ROM Proceedings of ASME Design Engineering Technical Conferences*, DETC2006/99175, Philadelphia, Pennsylvania, Sept. 10-13, 2006 (ASME, New York).
[17] Kannapan, S.M. and Marshek, K.M., An Approach to Parametric Machine Design and Negotiation in Concurrent Engineering, in *Concurrent Engineering: Automation, Tools and Techniques*, A. Kusiak (Ed.), 1992, pp. 509-533 (John-Wiley, New York).
[18] Michelena, N.F. and Papalambros, P. Y., Optimal Model-Based Decomposition of Powertrain System Design, *ASME Journal of Mechanical Design*, 1995, 117(4), pp. 499–505.

Contact: Simon Li
Concordia University
Concordia Institute for Information Systems Engineering
1455 de Maisonneuve Blvd. West, EV 7.648
Montreal, H3G 1M8
Canada
Phone: 1-514-848-2424 (extension: 5621)
Fax: 1-514-848-3171
E-mail: lisimon@ciise.concordia.ca
URL: http://www.ciise.concordia.ca/~lisimon

Simon is Assistant Professor in the Concordia Institute for Information Systems Engineering, Concordia University. His research interests include complex product development, computer-aided design, collaborative design methodology, and design change management. He received his bachelor (1998), master (2000), and doctoral (2005) degrees from University of Toronto.