

# Graph Partitioning-Based Coordination Methods for Large-Scale Multidisciplinary Design Optimization Problems

Zhoujie Lu\*

Joaquim R. R. A. Martins<sup>†</sup>

*Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI*

This paper presents an algorithmic framework that expedites the overall convergence of multidisciplinary design optimization problems through a three-step process: partitioning, reordering and coordination. Calculations of any complexity, as well as coupling information are represented by digraphs with weighted nodes and edges. A partition algorithm divides the MDO problems into multilevel nested clusters of calculations with minimal loss of information while maintaining balanced loads among assigned CPU nodes. The feedback loops within the clusters are further minimized by a reordering algorithm. A new multi-level hybrid multidisciplinary design feasible and individual design feasible coordination method is presented that allows a dynamic MDO architecture based on the computational cost and coupling strength. In addition, this paper describes a real-time visualization for MDO problems with weighted design structure matrix and graph representation. The results of a scalable multidisciplinary design analysis and optimization problem are presented.

## I. Introduction

The intricacy of engineering systems has increased at a tremendous pace over the last two hundred years. To efficiently and effectively manage hundreds and thousands of analyses and disciplines within the design process remains a challenging task. The increasing size of engineering projects has thankfully been accompanied by an exponential increase in computing power, as dictated by Moore's law. However, to take full advantage of this growth in computing power, it is necessary to develop new design optimization algorithms and methods that efficiently handle large-scale problems by utilizing parallel computation, since Moore's law has been increasingly dependent on multi-core processors.<sup>1</sup>

Multidisciplinary design optimization (MDO) has taken the systems engineering design process to another level by enforcing the quantification of every single aspect of the design, considering all the couplings between the various modules or disciplines, and solving an optimization problem in order to yield the best possible design.<sup>2-4</sup> Various MDO architectures have been developed, many of which decompose the problem and are amenable to parallelization.<sup>5-7</sup> However, there is still no MDO architecture that is dynamically created and scales well for large-scale multilevel nested problems.<sup>8</sup>

In order to address such challenges, a procedure to solve large-scale multidisciplinary design optimization problems is presented in this paper; the procedure consists three steps: III.A partitioning, III.B reordering, and III.C coordination. An algorithmic framework is developed that enforces this three-step procedure. Within the framework, a graph handling module dynamically maps the sensitivity and iteration cost of the problem to the digraph before the partition step. In addition, a real-time visualization module displays the weighted design structure matrix (WDSM) and the partitioned digraph at each iteration. These methods

\*PhD Candidate, Department of Aerospace Engineering, University of Michigan, AIAA Student Member

<sup>†</sup>Associate Professor, Department of Aerospace Engineering, University of Michigan, AIAA Senior Member

has been applied to scalable analytical multidisciplinary problems. The results shows a significant reduction in computational cost for large-scale problems, and the total computational cost scales well with the number of CPU nodes used.

This paper is organized as follows. Section II describes the MDO representation method that maps the coupling strengths and iteration costs to a weighted digraph. In addition, the visualization of MDO problems with WDSM and digraphs is described in this section. Section III discusses the partitioning, reordering, and coordination methods in detail. The multilevel hybrid multidisciplinary design feasible (MDF) and individual design feasible (IDF) coordination methods are presented in Section III.C. Section IV presents the results of applying the framework to a scalable analytical MDO test problem.

## II. Representation of Multidisciplinary Design Optimization Problems

One of the challenges in multidisciplinary design optimization is to visualize problems involving large-scale complex systems, and to identify the nested structures as well as the interconnections of disciplines or computations. In this section, weighted digraphs are introduced to represent MDO problems. This provides a mathematically sound graph representation of the connections within the system, and provides solid ground for applying graph partitioning techniques. Two visualization methods for MDO problems are discussed in this section.

### II.A. Graph Representation

The basis for this framework is built on graph theory, which was first introduced by Leonhard Euler in 1736, with the famous Seven Bridges of Königsberg problem.<sup>9</sup> The original problem was to find a path through the city of Königsberg that would cross each bridge once and only once among all seven bridges. Over the last centuries, the concept of graph theory was generalized by Cauchy<sup>10</sup> and L'Huillier<sup>11</sup> and is widely used in various fields and applications,<sup>12</sup> including load balancing for parallel computing using graph partitioning,<sup>13</sup> network problems,<sup>14</sup> and scheduling problems.<sup>15</sup> This section describes the representation of multidisciplinary design analysis and optimization problems with weighted digraphs.

In order to describe the MDO problems with the graphs, a set of mappings between the properties of the problems and the properties of the graphs is developed. A graph can be defined as an ordered pair  $G = (V, E)$  consisting of a set  $V$  of vertices or nodes together with a set  $E$  of edges or lines, which are two-element subsets of  $V$ . For the MDO problems, each node of the graph is chosen to represent to a calculation, an analysis, or a discipline with inputs and outputs that connect to other nodes. Each edge of the graph corresponds to the coupling strength or sensitivity between nodes. In order to fully describe the dependence between nodes of the MDO problems, digraphs are used to differentiate feed-forward and feed-backward of the data flows and respective sensitivities. A directed graph or digraph is a pair  $G = (V, E)$ , such that the set  $E$  is ordered pairs of nodes, called arcs, or directed edges. With a directed graph, the structure of the problems can be constructed of any complexity or nesting structure.

In addition to the directed graph representation, weighted edges and weighted nodes in digraphs are necessary to capture the other information within the MDO problems. A weighted graph is defined as a pair  $G = (V, E)$ , such that each edge in  $E$  and each node in  $V$  is associated with a positive weight. In the content of MDO, the weight of nodes is always positive, which corresponds to the computational time or iteration cost involved in each node. The node weights are initialized to unity at the first iteration and are updated based on the computational cost of the corresponding node at each consecutive iteration. The weight of the edges corresponds to the summation of the normalized sensitivity of inputs with respect to the outputs of the connected node. The normalization provides uniform and positive sensitivities for the comparison of all edges in the digraph.

The total edge weight of a node  $i$  is calculated as:

$$W_E(V_i) = \sum_{j=1}^n W_E(E_{ij}) \quad (1)$$

where  $W_E$  indicates the weight of the edges,  $E_{ij}$  denotes the edge between node  $i$  and node  $j$ , and  $n$  is the total number of edge pairs.  $W_E(V_i)$  provides an indication of the degree of node  $i$ , which shows the coupling of node  $i$  relative to other nodes. Such indication provides an estimate of the importance of node  $i$ , which helps to identify node clusters within the graph.

The total edge weight for graph  $G$  is defined as:

$$W_E(G) = \sum_{i=1}^m W_E(V_i) \quad (2)$$

where  $V_i$  denotes the node  $i$ , and  $m$  is the total number of nodes. In addition, the total node weight for graph  $G$  is given by:

$$W_V(G) = \sum_{i=1}^m W_V(V_i) \quad (3)$$

where  $W_V$  indicates the total weight of nodes.  $W_E(G)$  and  $W_V(G)$  provide a comparison between the problems of similar type. In general, the problems with large  $W_E(G)$  is more coupled thus difficult to converge, while large  $W_V(G)$  indicates the problem has higher computational cost. The ratio of  $W_E(G)$  before and after partitioning shows the amount of coupling information that is lost due to the partitioning step.

## II.B. Graph Visualization

Large-scale MDO problems are generally considered difficult to visualize the coupling between the analyses. In order to visualize the effectiveness of the proposed algorithms, two approaches are developed and discussed in this section: graph method and weighted design structure matrix. The two approaches are described in the following sections.

### II.B.1. Graph Method

The graph visualization routine is implemented to directly plot the weighted digraphs of the MDO problems. To maximize the information shown in the digraph, the size of nodes is proportional to the node weights which represents the computational cost; the line thickness of the edges indicates the edge weights which represents the sensitivities. In addition, the different colors show the partitions or the clusters of the systems. The partition method is discussed in more detail in Section III. The gray dashed edge line indicates the cut edge due to partitioning.

This visualization can be dynamically updated at each iteration to provide a realtime feedback to practitioners. The visualization routine is implemented in Python with object-oriented programming. Additional libraries such as NetworkX<sup>16</sup> are used to handles the graph manipulation, and GraphViz<sup>17</sup> to provides graph plotting routines. The spring method is used to minimize the crossed edges in the plot. An example of a problem with 20 disciplines is shown in Figure 1(a).

### II.B.2. Weighted Design Structure Matrix (WDSM)

The graph method is generally more suitable for small-scale problems. For large-scale highly coupled systems, the graph visualization becomes less readable and eventually ineffective. Therefore, in order to visualize systems with large number of nodes and edges, the weighted design structure matrix (WDSM) is preferable. Steward<sup>18</sup> first proposed the design structure matrix (DSM) to illustrate the coupling between design elements or tasks in systems engineering. DSM was further generalized to the visualization any data or process flow. It can also be viewed as the transpose of a square adjacency matrix constructed using graph theory.

Lambe and Martins<sup>19</sup> proposed eXtended Design Structure Matrix (XDSM) to denote the data and process flow for multidisciplinary design analysis and optimization problems, which provides critical insight of the structure of MDO problem. In this paper, WDSM is presented that includes the weight of nodes and edges into the DSM. WDSM focuses on the interaction or the coupling between nodes and is problem

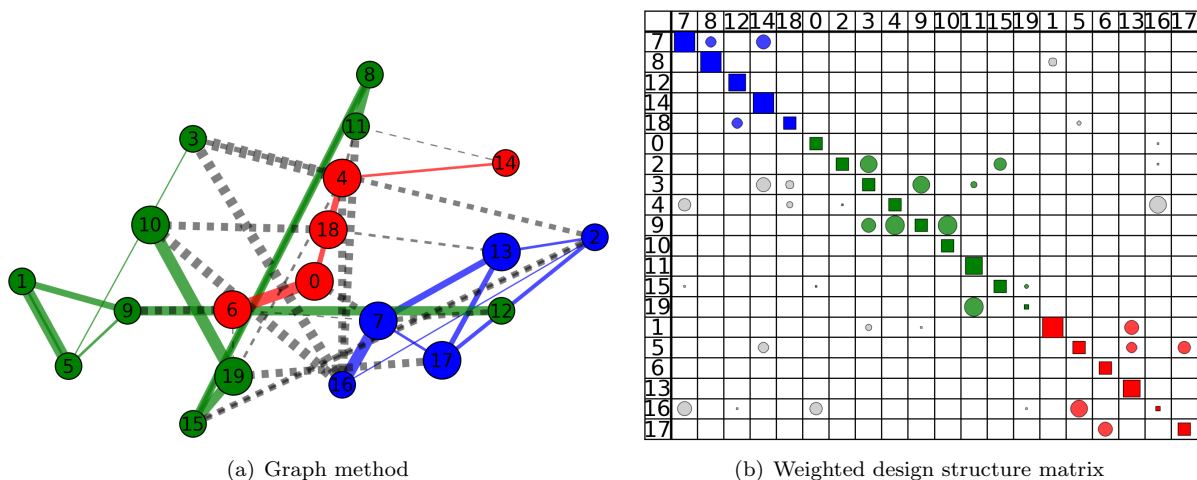


Figure 1. The visualization of a MDO problem with 20 nodes

dependent, while XDMS focuses on the data and process of a MDO architecture and is problem independent. XDMS is also used to visualize the coordination methods in this paper.

The WDSM is also dynamically updated at each iteration. The diagonal entries in WDSM are denoted by squares and the sizes are proportional to the node weights, which correspond to the computational cost of each calculation. The off-diagonal entries are marked by circles and the sizes indicate the coupling strength between the nodes. The same as in the graph visualization, the color denotes the partitions or clusters and the gray color indicates partition cuts. The sequence of nodes in WDSM is reordered to show the clusters of each partitions.

An example of 20 disciplines is shown in Figure 1(b). The WDSM contains the same information as the graph method. However, it is a clearer representation for large-scale highly-coupled MDO problems, and it is easier to identify clusters and the effectiveness of the partitioning step. The WDSM shows a significant advantage in the visualization of large-scale nested systems, while the graph method provides intuitive visualization for small-scale systems.

### III. Partitioning, Reordering and Coordination Methods

The algorithm proposed in this paper consists of three steps: [III.A](#) partitioning, [III.B](#) reordering, and [III.C](#) coordination. The partitioning step clusters nodes into  $k$  subsets, while minimizing the weight of the cut edges and balancing the partition loads. The reordering step further reorganizes the sequence of the nodes within each clusters to minimize the feedback loops. Finally, the coordination step superimposes a multilevel hybrid MDF-IDF architecture based on the partitioning information. Each of the three steps is discussed in the following sections.

#### III.A. Partitioning

The graph partitioning problem has been extensively studied over the last 20 years. There are two main types of partitioning algorithms: iterative improvement methods and global methods. The most widely used iterative method is Kernighan–Lin (KL) algorithm,<sup>20</sup> which solves a graph bisection problem to achieve partitioning. Fiduccia and Mattheyses<sup>21</sup> later modified the KL algorithm to handle weighted edges and nodes. Other variations of local search methods extended KL algorithm to achieve various functionalities.<sup>22–24</sup> Global methods, such as spectral-based methods,<sup>25–27</sup> network-based methods,<sup>28</sup> and hybrid genetic algorithms,<sup>29,30</sup> formulate the  $k$ -way partitioning problem as a global optimization problem and solve for the

optimal partitioning to avoid converging to a local optimal partitioning. The global methods are generally used as the initial partitioning for iterative improvement algorithms.

Karypis and Kumar<sup>31,32</sup> developed a software package for partitioning called METIS. They implemented a multilevel  $k$ -way partitioning algorithm with three steps. They first coarsen the graph by collapsing edges and combining vertices. Then, the coarsened graph is partitioned using multilevel bisection methods. Finally, the graph is uncoarsened and refined using global KL refinement algorithm.

For the work presented in this paper, a Python wrapper for METIS is developed based on the previous work by Kloeckner.<sup>33</sup> The graph is handled by NetworkX and converted to METIS for partitioning. In order to achieve multilevel optimal clusters, a top level iterator is developed to determine the optimal number of clusters at each level that allows the minimal amount of cut edge weights.

To initialize the algorithm, edge weights are computed at the starting point of the optimization. Since the sensitivities are often computed in gradient-based optimization, the sensitivities are re-used to update the edge weights without any additional computation cost. If the problem was originally constructed to solve with nongradient-based optimizers, finite-difference,<sup>34</sup> complex-step<sup>35</sup> or adjoint methods<sup>36</sup> can be used to compute the necessary sensitivities; however, it is significantly more costly and may compromise the benefit of the proposed methods. The methods presented in this paper would also work without the sensitivity information to some extent, but the partition and coordination algorithms would be significantly less effective without the sensitivities taken into account.

Since the computational cost of the partitioning algorithms are each analysis of interest, the partitioning is solved multiply times to obtain an optimal clustering. We solved the problems with different number of partitions and record the weight of cut edge. Then, the optimal partition number with minimum cut edge weight is chosen. The partitions shown in Figure 1 are obtained using the algorithm described above. Figure III.A shows an example of partitioning a large-scale problem into two-level clusters.

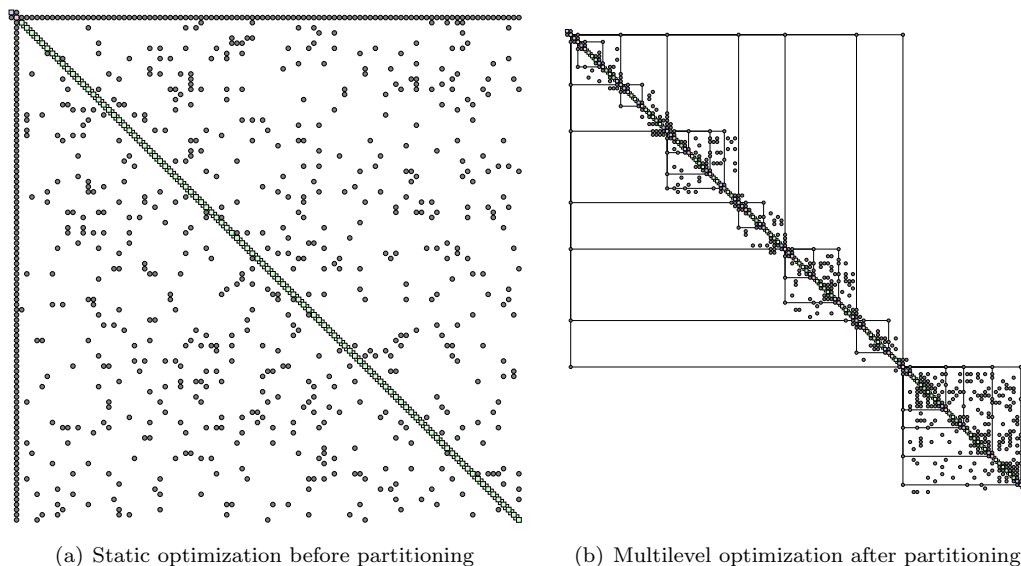


Figure 2. Partitioning of a large-scale optimization problem

### III.B. Reordering

The ordering or the sequencing of a cluster of iterative computations, such as Gauss–Seidel iterations, strongly affects the rate of convergence. Therefore, in order to further expedite the convergence of MDO problems after the partitioning step, it is preferable to reorder the computations within each cluster. The objective of the reordering step is to minimize the feedback loop by changing the execution sequence of the

computations. Rogers proposed DeMAID<sup>37,38</sup> which uses a generic algorithm to perform reordering of tasks. However, using generic algorithm to perform sequencing for each cluster of computations is too costly for the purposes of this paper. For large-scale design optimization problems, the cost of the reordering is equivalent to solving thousands of sequencing problems at each iteration. Thus, following the similar idea of Rogers,<sup>37</sup> a simpler iterative reordering algorithm is developed by examining the total node weights on the upper and lower triangle of the DSM.

For a given sub-WDSM or subgraph, the objective of reordering is to maximize the node weights in the upper triangle of the WDSM, while minimizing the node weights in the lower triangle by adjusting the sequence of nodes. This corresponds to a more ideal sequence such that the input of a computation only depends on the outputs of the computations that have already been executed. The method presented here processes each node by inserting the node into a new position within the cluster to increase the objective. The new position is determined by the edge weights on the row and column of its current position in DSM. For example, in Figure 3(a), the edge weights in the two spaces above node 3 are empty, while the weights in the two spaces to the left of node 3 are filled. Therefore, node 3 will be placed between node 0 and node 1 in the next iteration. This process is repeated for all the nodes. The iteration terminates when the graph converges to a stable arrangement or it reaches a maximum number of iterations. The iterative reordering algorithm is outlined below.

---

**Algorithm 1** Iterative Reordering Algorithm

---

```

1: for each level  $l$  in the partition do
2:   for cluster  $j$  in level  $l$  do
3:      $i \leftarrow 1$ ,  $Converge = False$ 
4:     while  $i \leq i_{max}$  &  $Converge = False$  do
5:       for node  $k$  in cluster  $j$  do
6:         Determine the new position  $z$  for node  $k$ 
7:         move node  $k$  to new position  $z$ 
8:         Store the sequence  $S_j$  for cluster  $j$ 
9:       end for
10:      if  $S_j$  is stable then
11:         $Converge$  is  $True$ 
12:      end if
13:       $i \leftarrow i + 1$ 
14:    end while
15:  end for
16: end for

```

---

An example of reordering step with a cluster of 5 computations is shown in Figure 3. This iteration method is efficient in finding an improved sequence with minimum feedback loop. The computational cost can be controlled by setting maximum iteration for each cluster. By default, the maximum iteration is set to be proportional to the number of nodes in each cluster. For a cluster of 20 nodes, 5 iterations usually gives a stable and improved sequencing. In addition, unlike the partitioning step, the reordering for each cluster and each level can be computed in parallel, which further reduce the computational cost. The cost of the reordering step is on the same order of the cost of the partitioning algorithm.

### III.C. Coordination

The first two steps create an effective graph partition for the MDO problems, and this last step coordinates the partitioned clusters of computations to solve the multidisciplinary design analysis and optimization problem with an dynamic MDO architectures. The coordination steps for Multidisciplinary Analysis (MDA) and Multidisciplinary Optimization (MDO) are presented separately in the following sections.

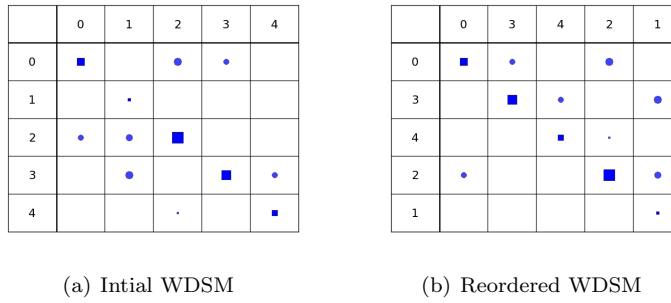


Figure 3. Iterative reordering algorithm applied to a cluster of 5 computations

### III.C.1. Multidisciplinary Design Analysis Problems

If the problem does not involve optimization, the MDA can be solved readily after partitioning with a number of iterative methods. Currently, two methods are implemented: block Gauss–Seidel iterations or Jacobi iterations. The Gauss–Seidel iterations converge faster than Jacobi iterations on a single processor, while Jacobi iterations can be parallelized to take advantage of multiple cores. For a large-scale problem, a multilevel MDA can be achieved with the partitioning algorithm by further partitioning the clusters of nodes into sub-clusters. The number of levels is set before the partition. The optimal number of partitions,  $k$ , at each level is determined by multiple evaluation of partitioning algorithm with different  $k$ 's to determine the minimum cut edge weights. This is feasible since the computational cost of the partitioning algorithm is significantly lower than the cost of the actual analysis. Other approaches include formulating the optimal  $k$  problem as an integer optimization problem, and is not implemented in the framework. Figure 4 shows the XDSM for a single level multidisciplinary design analysis with Gauss–Seidel iterations.

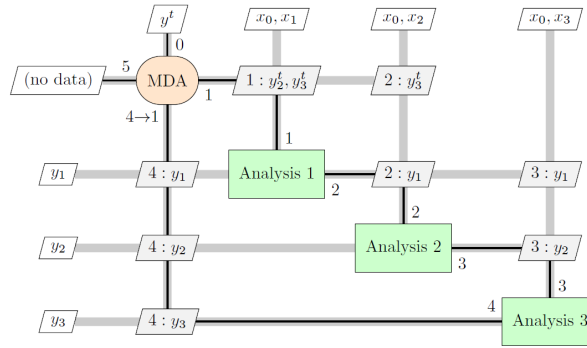


Figure 4. Single level multidisciplinary design analysis with Gauss–Seidel iterations

### III.C.2. Multidisciplinary Design Optimization Problems

The coordination algorithms for the MDO problems define the interactions between clusters of nodes, which is equivalent to defining an MDO architecture for the optimization problem. This paper is focus on coordination algorithms that result in multilevel monolithic architectures. The monolithic approach is known for its simplicity and faster convergence. One of the most popular monolithic architectures, such as Individual Design Feasible (IDF), and Multidisciplinary Design Feasible (MDF),<sup>3,4,39</sup> are considered here. In the monolithic approach, the MDO problem is solved as a single optimization problem. On the other hand, the



performance of distributed architectures on practical problems is still largely unknown. Studies suggest that the distributed architectures are generally less efficient than the monolithic architectures.<sup>41–43</sup> Therefore, due to the difficult and the complex nature of the distributed architectures, only monolithic architectures are explored in this paper.

Two key information about the partitioned problem is utilized in the coordination step: the partition structure of the problem, and the cut edge weights and total edge weights at each level and each cluster. After the partitioning and the reordering steps, highly coupled nodes in the MDO problem form clusters (or a hierarchy of clusters) with minimal dependence on the nodes outside the clusters. The coupled clusters can be identified by examining the total edge weight of the this cluster and the cut edge weight to form this cluster. Furthermore, the total cut edge weights at each level provide quantification of the strength of coupling that has been disconnected due to partitioning. The coordination step is constructed based on these two piece of information about the MDO problems.

We propose a coordination method based on a multilevel hybrid MDF-IDF architecture. The problem formulation of single-level MDF and IDF is shown in Table 1 and the corresponding XDSDM diagrams<sup>19</sup> are shown in Figure 5. Due to the additional information obtained during the partitioning and the reordering step, we can formulate the MDO architecture more wisely and dynamically. The hybrid architecture is designed to avoid the disadvantages of choosing either MDF and IDF by enforcing different sub-architectures at each level and/or at each cluster based on the nature of the sub-clusters. Therefore, the coordination algorithm chooses to enforce MDA or to enforce the coupling of the clusters as constraints in the optimizations at each level. The critical metrics are the cut edge weights and the total edge weights at each level. For a highly coupled problem, the MDF architecture generally has a faster convergence than the IDF architecture. Thus, our strategy is to choose the sub-architecture by comparing the cut edge weight of a sub-level  $W_E(C_{l,cut})$  with the total edge weight  $W_E(G_l)$ . In this algorithm,  $\omega_l$  is an adjustable bias parameter between 0 and 1 to determine the choice of sub-architectures at each level  $l$ . The multilevel hybrid MDF-IDF coordination algorithm is detailed in Algorithm 2.

	MDF		IDF
minimize	$f_0(x, \bar{y}(x, \bar{y}))$	minimize	$f_0(x, \bar{y}(x, \bar{y}^t))$
with respect to	$x$	with respect to	$x, \bar{y}^t$
subject to	$c_0(x, \bar{y}(x, \bar{y})) \geq 0$ $c_i(x_0, x_i, \bar{y}_i(x_0, x_i, \bar{y}_{j \neq i})) \geq 0$	subject to	$c_0(x, \bar{y}(x, \bar{y}^t)) \geq 0$ $c_i(x_0, x_i, \bar{y}_i(x_0, x_i, \bar{y}_{j \neq i}^t)) \geq 0$ $c_i^c = \bar{y}_i^t - \bar{y}_i(x_0, x_i, \bar{y}_{j \neq i}^t) = 0$

Table 1. Problem formulation of MDF and IDF

In this problem formulation,  $f_0$  is the objective function;  $x$  is the vector of design variables;  $\bar{y}$  is the vector of state variables.  $c_0$  and  $c_i$  denote the constraints. For the IDF,  $\bar{y}^t$  is the vector of target state variables, and  $c_i^c$  is the consistency constraint.

---

**Algorithm 2** Multilevel Hybrid MDF-IDF Coordination Algorithm

---

- 1: **for** each level  $l$  in the partition **do**
  - 2:   Determine  $w_l$
  - 3:   **if**  $W_E(C_{l,cut})/W_E(G_l) \leq \omega_l$  **then**
  - 4:     Enforce MDA for level  $l$  with either Gauss–Seidel or Jacobi iterations
  - 5:   **else**
  - 6:     Establish IDF for level  $l$
  - 7:     Add constraint and target variable to optimizer to ensure consistency within level  $l$
  - 8:     Level  $l$  will be iterated through optimizers and does not participate convergence at local levels
  - 9:   **end if**
  - 10: **end for**
-



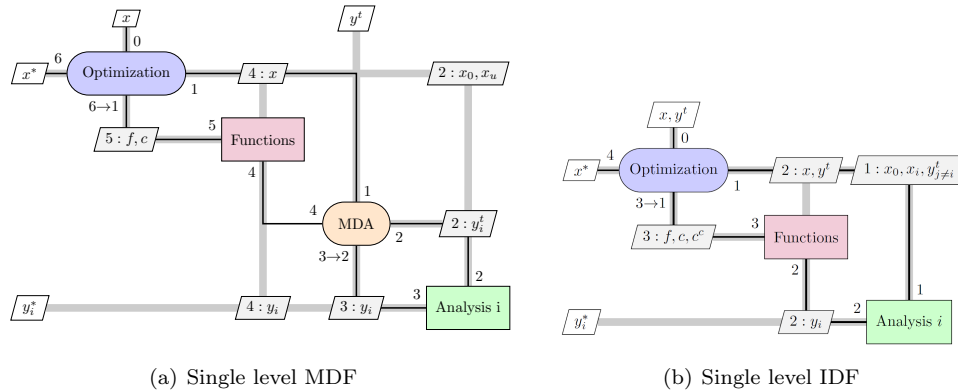


Figure 5. XDSM of single level MDF and IDF

In order to visualize the process of hybrid MDF-IDF, two examples of a two-level hybrid MDF-IDF architecture are shown in Figure 6 and Figure 7. Both examples are plotted in XDSM diagram format. Example 1 has MDF at both levels and all clusters, which results in a nested MDF architecture. Example 2 has IDF at the top level and one MDF and one IDF at the second level for each cluster. Since IDF can not have nested structure, the multilevel IDF will collapse into a single level IDF, as shown in this example.

With the coordination algorithm, dynamic and hybrid MDO architectures become possible. Each sub-architecture is chosen according to the coupling strength, as well as the node weights of each sub-level and cluster in order to expedite convergence. It is also possible to incorporate other monolithic or even distributed architectures with a more complicated coordination strategy. Such coordination algorithms are still under development. The goal of this three-step procedure is to allow dynamic self-organized MDO architectures that are problem independent. The concept of self-organization is that the solution strategies are dynamically adjusted to the properties of MDO problems.

#### IV. Scalable Analytical Multidisciplinary Problems

In order to study the effectiveness of the three-step procedure on large scale multidisciplinary problems, the scalable problem purposed by Tedford and Martins<sup>43</sup> is chosen to benchmark this method and to examine the effect of increasing dimensionality. This scalable problem was designed to allow researchers to examine the effects of increasing dimensionality while keeping manageable computational requirements. This scalable problem has the complete freedom to select the number of disciplines, the number of local and global design variables, the number of coupling variables, and the strength of the coupling. The problem formulation is stated as follows.

$$\begin{aligned}
 & \text{minimize} && z^T z + \sum_{i=1}^{N_y} y_i^T y_i \\
 & \text{with respect to} && x, z \\
 & \text{subject to} && 1 - \frac{y_i}{c_i} \leq 0, i = 1, \dots, N_y
 \end{aligned} \tag{4}$$

where  $N_y$  is the number of computations.  $z$  is the vector of the global design variables.  $x$  is the vector of the local design variables.  $y_i$  is the state variable. The governing equation for the state variable  $y_i$  is:

$$y_i = C_{x,i}x_i + C_{z,i}z + C_{y,i}y \tag{5}$$

In order to fully examine the effectiveness of this approach on different problems, the constants  $C_{x,i}$ ,  $C_{z,i}$ , and  $C_{y,i}$  are randomly generated to simulate different coupling strength and number of design variables. All  $C$ 's are adjusted within certain bounds to ensure stability.

The partitioning, reordering, and coordination strategy described in Section III.C.1 is applied to the scalable problem 4. Jacobi iteration is used to allow parallel computing capability. The computation time was

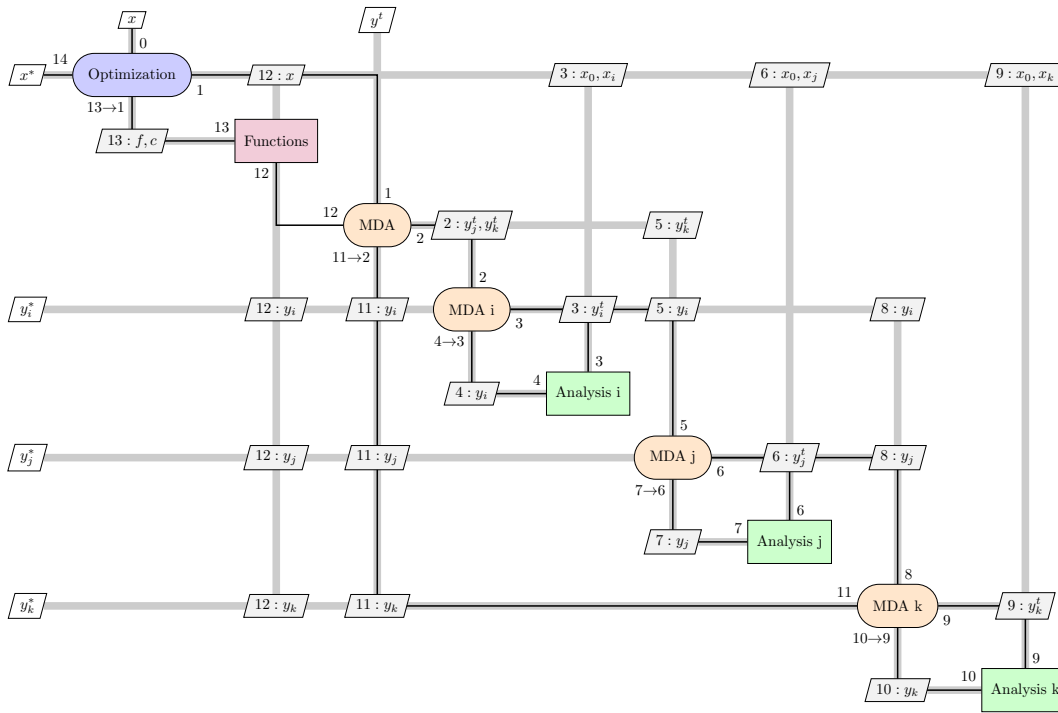


Figure 6. Example 1: Two level architectures with MDF at both levels

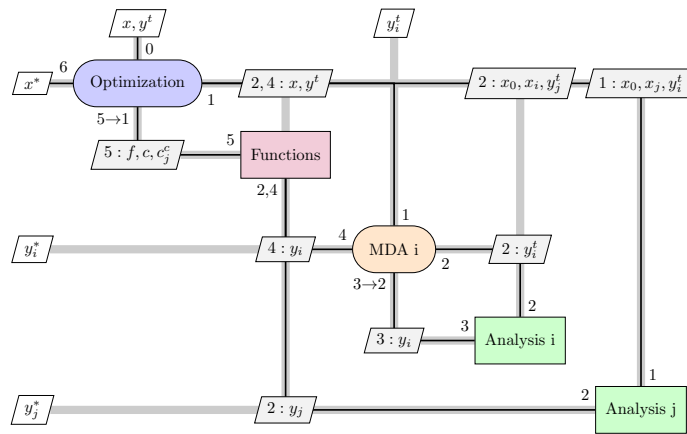


Figure 7. Example 2: Two level architectures with IDF at level top and one MDF and one IDF at second level

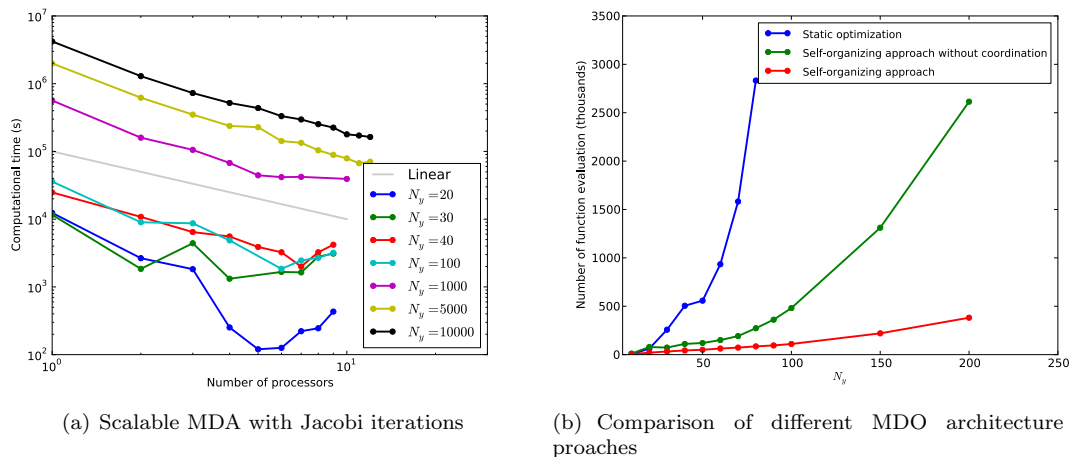


Figure 8. Scalable multidisciplinary design problem results

plotted as the dimensionality of the problem increases. For each dimension, the problem is randomly generated and computed 50 times to ensure an accurate representation. Then, the average of the computational time is recorded. Figure 8(a) shows the computational time of the random generated scalable problems. As can be seen on the plot, this method achieved linear scalability. The large scale problems obtain more consistent partitions than the small scale problems. The algorithm maintains a linear trend until a critical point that too much partitions lead to significant loss of coupling information, and the linear systems become very difficult to converge with the Jacobi iterations. This study shows the importance of obtaining a optimal number of partitions at each level of MDO problems with minimal loss of coupling information.

Multilevel hybrid MDF-IDF coordination with scalable multidisciplinary design optimization problems is also studied. The optimization problem is constructed with pyOpt<sup>44</sup> framework. SNOPT<sup>45</sup> is used as the optimizer for the problems. The optimization tolerance is  $10^{-6}$ . For the levels with MDF, Gauss–Seidel iterations are used with an under-relaxation parameter of 0.7. Three approaches are investigated: static optimization with one-level structure, self-organization approach without coordination (only using 2-level nested MDF), and self-organizing with coordination (using 2-level hybrid MDF/IDF). The parameter  $\omega$  is adjusted to ensure an efficient convergence. For this problem, we found that the problems converge faster if  $\omega$  is biased toward MDF, i.e.,  $\omega = 0.08$ .

Figure 8(b) shows the comparison between the three approaches with varying problem size  $N_y$ . Those large-scale problems require significantly longer time to run. Therefore, we chose to plot the number of function evaluation for each approach in order to perform a more accurate comparison. Similar to the study of MDA, at each dimension, the scalable problem is randomly generated and computed for 50 times, and the average number of function evaluation is plotted here. The static optimization formates the problem with a single MDF architecture. As the dimensionality of the problem increases, the cost immediately becomes prohibitive. By only using the self-organization approach without coordination, the computational cost is reduced dramatically by about 70%. The savings mostly due to the use of nested structures and by allowing high-coupled disciplines to be solved together. The number of iterations significantly reduced. Finally, by adding the coordination method, the computational time is further improved by dynamically choosing between MDF and IDF approach for each level and each cluster. Therefore, some clusters or computations only need to be iterated at the top level by the optimizers. The total number of iterations for optimization increases slightly, but the number of iterations in each level and cluster is reduced.

## V. Conclusion and Future Work

A novel three-step procedure to construct a dynamic MDO architecture is discussed in this paper. The main concept of this method is to use weighted digraph to represent MDO problems. Key information, such as computational cost and sensitivities, is mapped to the digraph. The representing graph is then partitioned to nested clusters of analyses or computations, while minimizing the information loss due to partition. The reordering step construct new sequences of the analyses within each level and each cluster, such that feedback loops are reduced. Finally, a multi-level hybrid multidisciplinary design feasible and individual design feasible coordination method is superimposed on the partitioned problems. This approach allows dynamic construction of a MDO architecture based on the properties of the problems. This three-step procedure is tested with a scalable MDA and MDO problem. The results show a 70% reduction in the total number of function evaluations. This approach proves the possibility of mixed implementation of different MDO architectures. The future work includes: extending the coordination method with more monolithic or even distributed architectures; and incorporating sub-optimizers into this approach.

## References

- <sup>1</sup>Sutter, H., "The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software," *Dr. Dobbs's Journal*, Vol. 30, No. 3, March 2005.
- <sup>2</sup>Alexandrov, N. M., "Multilevel Methods for MDO," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. M. Alexandrov and M. Y. Hussaini, SIAM, Philadelphia, 1997.
- <sup>3</sup>Cramer, E., Dennis, J., Frank, P., Lewis, R., and Shubin, G., "Problem formulation for multidisciplinary optimization," *SIAM Journal on Optimization*, Vol. 4, No. 4, 1994, pp. 754–776.
- <sup>4</sup>Sobieszczanski-Sobieski, J. and Haftka, R., "Multidisciplinary aerospace design optimization: survey of recent developments," *Structural Optimization*, Vol. 14, No. 1, 1997, pp. 1–23.
- <sup>5</sup>Sobieszczanski-Sobieski, J., Agte, J. S., and Sandusky, R. R., "Bilevel Integrated System Synthesis," *AIAA Journal*, Vol. 38, No. 1, 2000, pp. 164–172.
- <sup>6</sup>Haftka, R., Sobieszczanski-Sobieski, J., and Padula, S., "On options for interdisciplinary analysis and design optimization," *Structural Optimization*, Vol. 4, 1992, pp. 65–74.
- <sup>7</sup>Kroo, I. M., "MDO for Large-Scale Design," *Multidisciplinary Design Optimization: State-of-the-Art*, edited by N. M. Alexandrov and M. Y. Hussaini, SIAM, 1997, pp. 22–44.
- <sup>8</sup>Tedford, N. P. and Martins, J. R. R. A., "Comparison of MDO Architectures within a Universal Framework," *Proceedings of the 2nd AIAA Multidisciplinary Design Optimization Specialist Conference*, Newport, RI, May 2006, AIAA 2006-1617.
- <sup>9</sup>"Problem of Seven Bridges of Konigsberg," *Encyclopedia of GIS*, edited by S. Shekhar and H. Xiong, Springer, 2008, p. 913.
- <sup>10</sup>F. Harary, *Graph Theory*, Addison-Wesley, 1969.
- <sup>11</sup>Lhuillier and Gergonne, "Géométrie. Mémoire sur la polyédrométrie; contenant une démonstration directe du théorème d'Euler sur les polyèdres, et un examen des diverses exceptions auxquelles ce théorème est assujetti," 1812-1813.
- <sup>12</sup>Wilson, R. J. and Beineke, L. W., editors, *Applications of Graph Theory*, Academic Press, London, 1st ed., 1979.
- <sup>13</sup>Conrad, J. M. and Agrawal, D. P., "A Graph Partitioning-Based Load Balancing Strategy for a Distributed Memory Machine," *Proceedings of the International Conference on Parallel Processing*, 1992, pp. II-74–II-81.
- <sup>14</sup>Sanchis, L. A., "Multiple-Way Network Partitioning," *IEEE Transactions on Computers (TOC)*, Vol. C-38, No. 1, Jan. 1989, pp. 62–81.
- <sup>15</sup>McCreary, C. L. and Gill, D. H., "Automatic Partitioning and Virtual Scheduling for Efficient Parallel Execution," Tech. Rep. CSE91-02, Auburn Univ., March 1991.
- <sup>16</sup>Hagberg, A., Schult, D., and Swart, P., "NetworkX, High productivity software for complex networks," *Webová stránka https://networkx.lanl.gov/wiki*, 2006.
- <sup>17</sup>Ellson, Gansner, Koutsofios, North, and Woodhull, "Graphviz – Open Source Graph Drawing Tools," *GDRAWING: Conference on Graph Drawing (GD)*, 2001.
- <sup>18</sup>Steward, D., "Design structure system: A method for managing the design of complex systems," *IEEE TRANS. ENG. MGMT.*, Vol. 28, No. 3, 1981, pp. 71–74.
- <sup>19</sup>Lambe, A. B. and Martins, J. R. R. A., "Extensions to the Design Structure Matrix for the Description of Multidisciplinary Design, Analysis, and Optimization Processes," *Structural and Multidisciplinary Optimization*, 2012, (In press).
- <sup>20</sup>Kernighan, B. and Lin, S., "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, Vol. 49, No. 2, 1970, pp. 291–307.
- <sup>21</sup>Fiduccia, C. and Mattheyses, R., "A linear-time heuristic for improving network partitions," *Papers on Twenty-five years of electronic design automation*, ACM, 1988, pp. 241–247.

- <sup>22</sup>Vahid, F. and Le, T., "Extending the Kernighan/Lin heuristic for hardware and software functional partitioning," *Design Automation for Embedded Systems*, Vol. 2, No. 2, 1997, pp. 237–261.
- <sup>23</sup>Bui, T., Heigham, C., Jones, C., and Leighton, T., "Improving the performance of the Kernighan-Lin and simulated annealing graph bisection algorithms," *Proceedings of the 26th ACM/IEEE Design Automation Conference*, ACM, 1989, pp. 775–778.
- <sup>24</sup>Dhillon, I., Guan, Y., and Kulis, B., "A fast kernel-based multilevel algorithm for graph clustering," *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, ACM, 2005, pp. 629–634.
- <sup>25</sup>Hagen, L. and Kahng, A., "New spectral methods for ratio cut partitioning and clustering," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Vol. 11, No. 9, 1992, pp. 1074–1085.
- <sup>26</sup>Filippone, M., Camastra, F., Masulli, F., and Rovetta, S., "A survey of kernel and spectral methods for clustering," *Pattern recognition*, Vol. 41, No. 1, 2008, pp. 176–190.
- <sup>27</sup>Hendrickson, B. and Leland, R., "An improved spectral graph partitioning algorithm for mapping parallel computations," *SIAM Journal on Scientific Computing*, Vol. 16, No. 2, 1995, pp. 452–469.
- <sup>28</sup>Van Den Bout, D. and Miller III, T., "Graph partitioning using annealed neural networks," *Neural Networks, IEEE Transactions on*, Vol. 1, No. 2, 1990, pp. 192–203.
- <sup>29</sup>Bui, T. and Moon, B., "Genetic algorithm and graph partitioning," *Computers, IEEE Transactions on*, Vol. 45, No. 7, 1996, pp. 841–855.
- <sup>30</sup>Kang, S. and Moon, B., "A hybrid genetic algorithm for multiway graph partitioning," *Proceedings of the Genetic and Evolutionary Computation Conference*, Citeseer, 2000, pp. 159–166.
- <sup>31</sup>Karypis, G. and Kumar, V., "METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," *University of Minnesota*, Vol. 102, 1998.
- <sup>32</sup>Karypis, G. and Kumar, V., "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, Vol. 20, No. 1, 1999, pp. 359.
- <sup>33</sup>Klockner, A., Warburton, T., and Hesthaven, J., "High-Order Discontinuous Galerkin Methods by GPU Metaprogramming," 2011.
- <sup>34</sup>Chapra, S. and Canale, R., "Numerical methods for engineers," *New York*, 1990.
- <sup>35</sup>Martins, J., Sturdza, P., and Alonso, J., "The complex-step derivative approximation," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 29, No. 3, 2003, pp. 245–262.
- <sup>36</sup>Martins, J., Alonso, J., and Reuther, J., "A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design," *Optimization and Engineering*, Vol. 6, No. 1, 2005, pp. 33–62.
- <sup>37</sup>Rogers, J., "DeMAID: A Design Manager's Aide for Intelligent Decomposition User's Guide," 1989.
- <sup>38</sup>Rogers, J., "DeMaid/GA-an enhanced design manager's aid for intelligent decomposition," *AIAA Paper*, Citeseer, 1996.
- <sup>39</sup>Sobieszczanski-Sobieski, J., James, B., and Riley, M., "Structural optimization by generalized, multilevel decomposition," 1985.
- <sup>40</sup>Cramer, E., Dennis Jr, J., Frank, P., Lewis, R., and Shubin, G., "Problem formulation for multidisciplinary optimization," *SIAM Journal on Optimization*, Vol. 4, 1994, pp. 754.
- <sup>41</sup>Perez, R. E., Liu, H. H. T., and Behdina, K., "Evaluation of Multidisciplinary Optimization Approaches for Aircraft Conceptual Design," Aug. 2004, AIAA 2004-4537.
- <sup>42</sup>Yi, S. I., Shin, J. K., and Park, G. J., "Comparison of MDO methods with mathematical examples," *Structural and Multidisciplinary Optimization*, Vol. 35, No. 5, 2008, pp. 391–402.
- <sup>43</sup>Tedford, N. and Martins, J., "Benchmarking multidisciplinary design optimization algorithms," *Optimization and Engineering*, Vol. 11, No. 1, 2010, pp. 159–183.
- <sup>44</sup>Perez, R. E., Jansen, P. W., and Martins, J. R. R. A., "pyOpt: a Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization," *Structural and Multidisciplinary Optimization*, 2011, (In press).
- <sup>45</sup>Gill, P., Murray, W., and Saunders, M., "SNOPT: An SQP algorithm for large-scale constraint optimization," *SIAM Journal of Optimization*, Vol. 12, No. 4, 2002, pp. 979–1006.