An application of Graph Theory to MDAO problem formulation

David Pate, *Dr. Brian German †Justin Gray,‡

blah blah blah

Nomenclature

AAO	All-At-
MDAO	Multidisciplinary Design Analysis and Optimization
FPF	Fundamental Problem Formulation

I. Introduction

As the size and complexity of engineering systems grows the time and expense for setting up analysis models grows with them. Multidisciplinary Design Analysis and Optimization (MDAO) frameworks such as OpenMDAO[1] and ModelCenter have enabled a new level of analysis tool integration and paved the way for models of with more analysis tools and increasing numbers of multidisciplinary couplings. Such complex models present a distinct challenge proper implementation of any given solution strategy . In fact as the size of a problem grows very large even selecting an appropriate solution strategy can be a daunting task.

To help address in order to address the complexity problem we propose the use of graph based problem formulation in a way that allows for both analysis of potential solution strategies and automated implementation of those strategies inside an MDAO framework. In order to serve that purpose, the problem needs to include the following information:

- Discipline Analyses
- Discipline State Variables and Residuals
- Design Variables
- Constraints
- Objective or Objectives
- Coupling Constraints
- Parameters

Notably absent from the preceding list are any kind of solvers, optimizers, or other iterative solution finding tools. At it most basic, a problem formulation includes only information about what is being sought after in a given problem, or what the goals of a given problem are. It need not contain any information about solution paths or strategies to reach those goals. A general specification is one that states nothing specific about a problem solution path.

We define a complete and general problem specification striped down to it's most essential parts to be the Fundamental Problem Formulation (FPF). By definition, the FPF for any given problem will be constant

*Georgia Tech
†Georgia Tech...
‡Aerospace Engineer, MDAO Branch, Mail Stop 5-11, AIAA Member

regardless of which MDAO framework, optimization architecture, optimizer, or solver is used to solve the problem.

In this work we proposed a graph based syntax for the specification of problem formulation. This graph based syntax provides several key features that make it useful for working with large scale MDAO problems. It provides a rigid structure that can be easily manipulated with a wide range of well establish graph-theory algorithms for the purposes of problem decomposition. The graph syntax also provides a means for algorithmically testing a given problem formulation to check if it is the FPF, and if not, to reduce it to the FPF. Lastly, a graph based specification for problem formulation lends itself well to interacting with MDAO frameworks which dramatically increases it's utility for real world applications.

A. Fundamental Problem Formulation

Problem formulation is most commonly given in a mathematical syntax. For a simple, notional problem the problem formulation is as follows:

$$\begin{aligned} given \quad & A : x, y \rightarrow m, z \\ & B : x, z \rightarrow y^t \\ min. \quad & f(m) \\ w.r.t. \quad & x, y, z \\ s.t. \quad & g(y^t, y) = 0 \end{aligned} \tag{1}$$

Where A and B represent analysis tools, f, g are the objective and constraint functions respectively. While this does represent a complete problem formulation, it's not the fundamental one since some assumptions have inherently been made about the solution strategy. Analysis A outputs z which is an input to B. Hence, A should be run before B with the y being iterated on to convergence. However, a different solution could be equally valid and still represent the same fundamental problem:

$$\begin{aligned} given \quad & B : x, z \rightarrow y \\ & A : x, y \rightarrow m, z^t \\ min. \quad & f(m) \\ w.r.t. \quad & x, y, z \\ s.t. \quad & g(z^t, z) = 0 \end{aligned} \tag{2}$$

Equation 2 differs only slightly from Eqn. 1. A is now dependent on the output of B, and z will be iterated on to meet g. Now the problem can be solved by running B first and then A. Since the formulations in Eqn. 1 and 2 both describe the same problem there must be a more fundamental description of the problem that is common between them. We present the FPF as follows:

$$\begin{aligned} given \quad & A : x, y \rightarrow m, z^t \\ & B : x, z \rightarrow y^t \\ min. \quad & f(m) \\ w.r.t. \quad & x, y, z \\ s.t. \quad & g1(z^t, z) = 0 \\ & g2(y^t, y) = 0 \end{aligned} \tag{3}$$

The FPF in Eqn. 3 differs from both Eqn. 1 and Eqn. 2 because it has two constraints which both must be met. The presence of both of these constraints fully decouples the problem so that either A or B could be run first or both could be run simultaneously. By removing either constraint, and replacing it with a direct dependence between the two analyses you could regain the earlier two problem formulations.

B. Problem Formulation Syntax

As shown above, the mathematical language for specifying problem formulations is very general and can be used both for fundamental and specific problem formulations. Tedford and Martins used the above mathematical syntax to specify the FPF for a set of test problems and also to describe specific formulations for solving them with a number of optimization architectures.[2] Their work demonstrates clearly how multiple specific problem formulations can all relate back to a common FPF.

The challenge with using this traditional mathematical syntax is that it is not easily manipulated or analyzed. A number of matrix based methods have been used successfully to translate the mathematical syntax into a more useful computational form. Steward's Design Structure Matrix (DSM) is a square adjacency matrix which captures the relationship between analysis tools where off diagonal elements of the matrix indicate coupling.[3] Figure 1 shows a DSM for the above notional example with and without the coupling cycle identified. In order to derive the formulations in Eqn. 1 and Eq. 2 you would remove one of the output target variables (y^t or z^t) and replace it with a direct connection which would select either A or B to be executed first. For more complex problems, choosing the proper order to run analysis tools is a non-trival task. Rogers et. al developed DeMAID to manipulate a DSM to find an ordering for analysis tools that reduces the cost of solving highly coupled systems.[4] This re-ordering yields multiple specific problem formulations which all solve the same FPF. In other words, manipulation of the DSM does not fundamentally alter the problem formulation, which makes DSM an excellent foundation for specifying the FPF itself. Notice however, that the DSM in Figure 1 contains less information than the Eqn. 3. While it does fully capture the coupling information, objective and constraint information is missing.

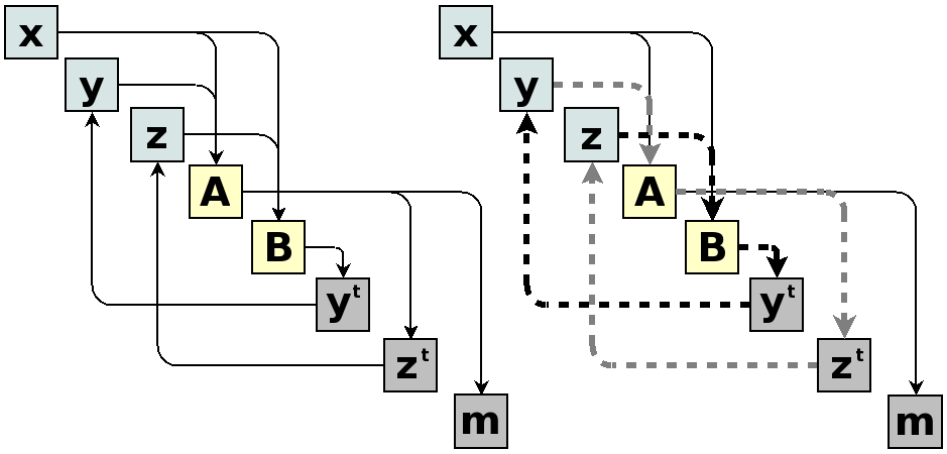

Figure 1: DSM specifications for Eqn. 3. Coupling cycle is identified by dashed lines.

Since a DSM describes a square adjacency matrix, it can be represented in an equivalent directed graph where nodes represent analysis tools and edges represent information exchange between those tools. An alternate matrix based syntax, called a Functional Dependence Table (FDT), was proposed by Michelena and Papalambros. FDT represents the relationship between functions, including objectives and constraints, and their values.[5] Similar to DSM FDT also describes an adjacency matrix of a graph. Unlike the DSM graph, however, a FDT graph is an undirected graph where nodes can represent analysis tools, objectives, or constraints. Edges between nodes represent a dependence on the same variable value. By searching the FDT graph for clusters of totally connected nodes Wagner and Papalambros identified groups of analysis tools that were all dependent on the same input variables and used that to make partitioning decisions.[6] FDT retains a greater portion of the information in the problem formulation, but it is still not complete. It ignores information about coupling between analyses.

It is possible to combine the syntax of an FDT and a DSM by making a small extension to a traditional DSM specification. Normally, a DSM is given with analysis tools as nodes and variable dependencies given as edges. Lamb and Martins included the variables, objectives, and constraint functions as nodes in an Extended DSM (XDSM)[7] in order to capture a more complete problem formulation for MDAO problems. Lu made use of XDSM to perform ordering and partitioning on the fundamental problem formulation specified by XDSM in order to achieve significant reductions in computational costs for large scale optimization problems.

Figure 2 illustrates the same notional analysis as in Figure 1 represented as an XDSM.

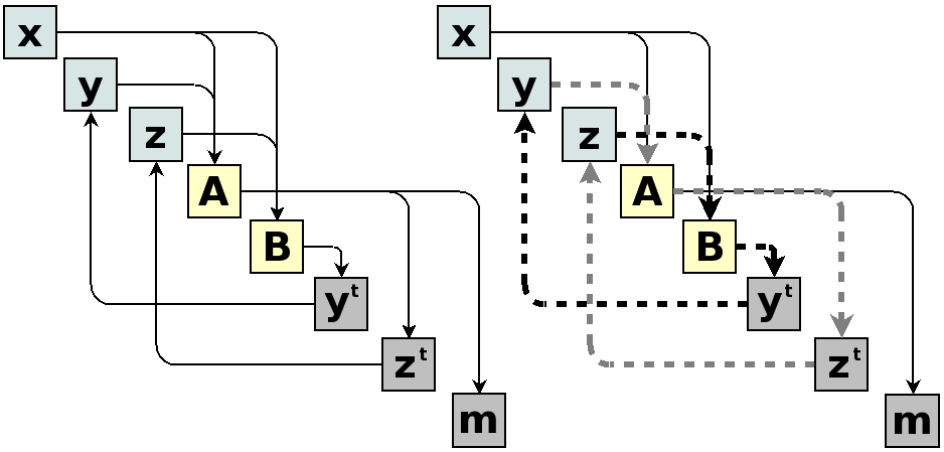

Figure 2: [TODO: Generate XDSM]Extended DSM graph for Eqn. 3

II. Graph Based Problem Formulation Syntax

A. Formulation Graph Syntax

Rather than start with an adjacency, we chose to work directly with a directed cyclic graph to develop a syntax for the FPF. The following information must be provided to start:

- Analysis blocks: an analysis block represents any calculation, and each comes with
  - local inputs
  - local outputs
  - execution properties: the properties associated with running the code, such as run time.
- Global parameters: these may serve as fixed inputs to the local inputs of analysis blocks
- Global outputs: these may represent
  - objectives
  - constraints
  - residuals: (not sure)

The graph representation of a data flow is cast to utilize the extensive library of algorithms in graph theory to analyze a directed weighted graph. Edge weights are used to represent the metrics associated with a data flow

- Run time: this metric is a property of an analysis block
- Fidelity: this metric is a property of an individual local output
- Expected Convergence

The key assumption is that identical variables are recognized as such. This serves as the basis for creating a data flow by connecting compatible input and output nodes with a directed edge. In order to represent the fact that execution of an analysis code does not depend on the number of outputs being used, it is represented as depicted in the following figure (not made yet).

This information immediately leads to the maximal connectivity graph, which is formed by placing a directed edge from each local output or global parameter to each matching local input or global output. Whenever multiple edges are connected to a single input a conflict occurs because only one may be used. Resolving these conflicts is one key challenge in creating a data flow.

B. Solution Graph Syntax

What is the difference between a problem formulation and a problem solution method? Convert from a cyclic graph, to an acyclic graph

- cycles indicate convergence loops or design varaible loops
- Problem can't be solved until all loops are *removed* by adding in solvers/optimizers
- *special* nodes for solvers and optimizers that *break* loops (from an algorithmic point of view)
- FPF represents the minimal amount of information necessary to define a problem
- Any solution path grows the graph complexity by adding edges and nodes (or possibly have an empty solution graph, which you build up as you remove edges from problem formulation graph?)

III. Example Problem

IV. Applications

V. Conclusions

References

[1]Gray, J., Moore, K. T., Hearn, T. A., and Naylor, B. A., "A Standard Platform for Testing and Comparison of MDAO Architectures," *8th AIAA Multidisciplinary Design Optimization Specialist Conference (MDO)*, Honolulu, Hawaii, 2012, pp. 1–26.

[2]Tedford, N. P. and Martins, J. R. R. a., "Benchmarking multidisciplinary design optimization algorithms," *Optimization and Engineering*, Vol. 11, No. 1, March 2009, pp. 159–183.

[3]Steward, D. V., "The Design Structure System: A Method for Managing the Design of Complex Systems," *IEEE Transactions on Engineering Management*, Vol. EM-28, No. 3, 1981, pp. 71–74.

[4]Rogers, J., McCulley, C., and Bloebaum, C., *Integrating a genetic algorithm into a knowledge-based system for ordering complex design processes*, NASA Technical Memorandum, Hampton Virginia, 1996.

[5]Michelena, N. F. and Papalambros, P. Y., "A Hypergraph Framework for Optimal Model-Based Decomposition of Design Problems," *Mechanical Engineering*, Vol. 8, No. 2, 1997, pp. 173–196.

[6]Wagner, T. C. and Papalambros, P. Y., "A general framework for decomposition analysis in optimal design," *Advances in Design Automation*, Vol. 2, 1993, pp. 315–325.

[7]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.