Enumeration of Architectures with Structured Components

Shangting Li[1]

December 8, 2017

Abstract

Many engineering architectures can be represented as graphs comprised of components and their connections. This project develops the methods for expanding structured components in graphs representing architectures. In addition, we investigate a number techniques to increase the efficiency of the overall approach.

1 Background Information

Graph-based methods can be used to represent design architecture problems. For example, using the work in Ref. [1], the specification of the potential hybrid powertrain architectures can be represented by:

$$C = \{V, G, M, P, E\}, \quad R = [1, 1, 2, 1, 1]^T, \quad P = [1, 3, 1, 3, 1]^T, \quad X = [0, 0, 0, 1, 0]^T \tag{1}$$

where C set is the assigned colored labels for the different component types, R vector is the number of replicates of the colored labels, and P vector represents the number of ports or number of connections for the component type. The labels in this example represent V: Vehicle, G: simple Gear, M: Motor, P: Planetary gear, E: Engine. Using the information from (C, R, P), one potential architecture and its graph-based representation is shown in Fig. 1. Detailed information on how to enumerate all the graphs using (C, R, P) is in Refs. [1, 2].

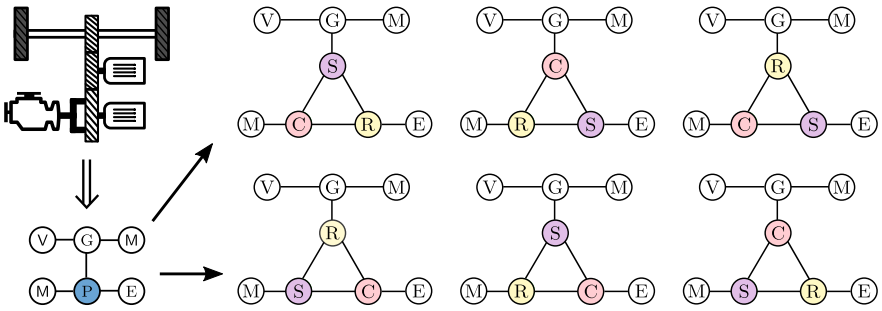


Figure 1: A hybrid powertrain (both simple and structured graph representations).

There are two types of components: simple (0) and structured (1), represented by X. A simple component is defined as a component where the labeling of the individual connections is not important, while the distinction between connections is essential for structured components. The work in Refs. [1, 2] focused on simple components only. In this project, we address the generation of graphs with structured components. We define a structured graph as a graph whose structured components have all their connections labeled (or expanded), and simple graph is when all connections are unlabeled or unexpanded.

2 Expanding All Structured Components

Simple graphs do not provide enough information when some components are structured [1]. For example, in the simple graph in Fig. 1, the connection labels in structured component P (namely, S: sun, R: ring, and C: carrier) need to be defined for a proper architecture. To properly describe the desired architectures,

[1]B.S. candidate in Industrial Engineering, Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Email: sli114@illinois.edu,

we expand all the structured components in the graphs, generating all possible orders of these connection labels. It is possible that multiple connections (i.e., a multiedge) exist between simple components. An example would be two planetary gears that have two distinct connections between them [3]. Furthermore, there may be internal loops where a component is connected to itself. Continuing with our example, the R and S in the planetary gear could be connected, creating an internal loop.

2.1 Total Required Permutations

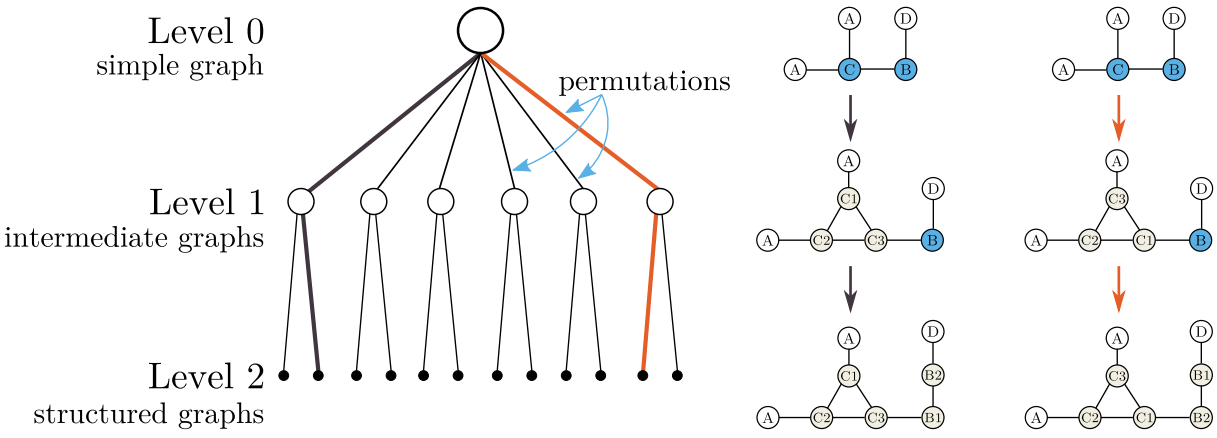
Generating all possible structured graphs from a single simple graph requires the generation of the appropriate permutations of the connection labels for each structured component. For example, the expansion of the planetary gear component in the hybrid powertrain is shown in Fig. 1, generating six structured graphs. Consider the structured components with p_i total number of ports, r_i is the number of replicates, and ℓ_i internal loops where i is the i-th structured component type. Then the total number of graphs N produced by expanding all structured component types is bounded by:

$$N \leq \prod_{i=1}^{\Sigma X} \underbrace{\frac{p_i!}{\ell_i!(p_i-\ell_i)!}\,(2\ell_i-1)!!}_{\text{internal loops}} \times \underbrace{(p_i-2\ell_i)!}_{\text{external}} \times r_i \tag{2}$$

where ! is the factorial function and !! is the double factorial function. This is an upper bound since some graphs may not be unique (discussed in the next section).

2.2 Tree Graph Representation of the Expansion Process

The expansion process of all the structured components can be seen as visiting all nodes in a tree graph. This is visualized for a specific architecture problem in Fig. 2c. Each node in the tree represents a graph produced after a single structured component is expanded. At each level, one structured component is expanded and the multiple branches represent the different permutations defined in Eqn. (2).



(a) Tree representation. (b) Purple path. (c) Orange path.

Figure 2: Tree-based representation of expansion process.

3 Generating the Set of Unique Graphs More Efficiently

During the expansion process, isomorphic (or similar) graphs can be produced. An example of this is shown in Fig. 3, where both structured graphs represent the same architecture. In general, we only want the set of nonisomorphic (or unique) architectures. We could check the entire N graphs after all structured components are expanded, but $(N-1)(N-2)/2$ graph comparisons would be needed. In this section, we

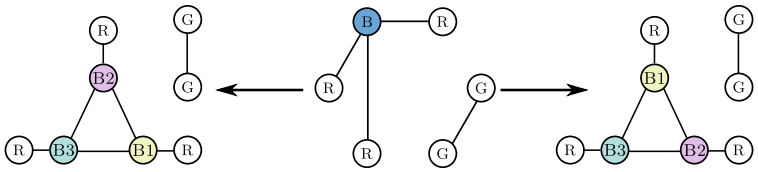
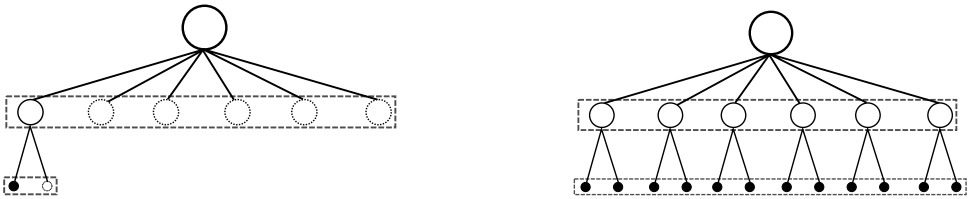


Figure 3: Two isomorphic structured graphs.

explore ways to reduce the total number of graph comparisons needed.

3.1 Checking Isomorphic Graphs at Each Level

An alternative to checking for colored graph isomorphisms at the end of the process (the original method) would be to check for isomorphisms at each level during a breadth-first search of the tree [4], termed level-order isomorphism (LOI) checking. Recall the levels in the tree represent the expansion of a single structured component. The LOI checking method can potentially remove isomorphic intermediate graphs, reducing the size of the tree and potentially reducing the number of graph comparisons needed. The LOI method can perform either better or worse depending on the extent of the isomorphic nature of the graph set. The minimum number of graph comparisons needed occurs when all graphs are isomorphic ($N = 1$) as illustrated in Fig. 4a, since we eliminate most graphs at each level and LOI method greatly outperforms the original method. On the other hand, the LOI method performs poorly when all graphs are unique as illustrated in Fig. 4b because the comparisons at each level do not eliminate any graphs. Typical architecture problems of interest involve many isomorphic graphs, so the LOI method would be selected.



(a) All graphs isomorphic.

(b) All graphs unique.

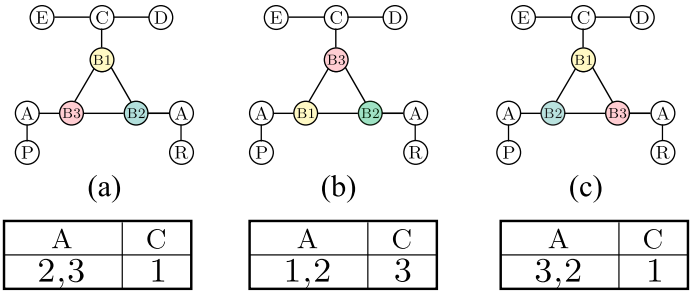
Figure 4: Two extreme cases using the LOI method.

3.2 Ordering of Components

We explored the effect of different component orderings (six different methods) on the efficiency of LOI method. Different orderings change the structure of the tree because different structured components are expanded in different orders. In general, the most efficient method was to sort the number of ports in an increasing order.

3.3 Checking of Connection Sets Between Two Structured Graphs

Child graphs from the same parent graph have the same ordering of their structured components. As a consequence, if we can determine the mismatch between any set of connections in the structured components between two graphs, we can conclude that two graphs are unique (not isomorphic) without complete isomorphism checking. For example consider the graphs in Fig. 5a and Fig. 5b. Since the connections related to C and A are different between the two graphs (i.e., $\{2,3\} \cup \{1,2\} = \{1,2,3\} \neq \{2,3\}$), we can conclude that these graphs are different. This simple check is much faster than a complete isomorphism check. However, by comparing connection orders of the Fig. 5a and Fig. 5c graph, we cannot decide whether the two graphs are



A	C
2,3	1

A	C
1,2	3

A	C
3,2	1

Figure 5: Comparing different connection sets to determine if two graphs are potentially isomorphic.

unique so the complete isomorphism checking would be needed. In general, this check reduces the number of complete isomorphism checks needed, reducing computational expense.

4 Example from Ref. [3]

From the work in this project, we were able to replicate the hybrid powertrain architectures found in Ref. [3] using their assumptions. For one planetary gear there are 120 architectures, and for two planetary gears there are 3618 architectures.

5 Acknowledgements

I am especially grateful of Prof. James T. Allison and Ph.D. Candidate Daniel R. Herber for bringing me to team of ESDL. They have been very supportive to my project and invested a great amount of time to actively providing me with professional advice and assistance. I would also like to express gratitude towards Department of Industrial and Enterprise Systems Engineering for their support and recognition of my research.

References

[1] D. R. Herber, T. Guo, and J. T. Allison, "Enumeration of architectures with perfect matchings," *Journal of Mechanical Design*, vol. 139, p. 051403, May 2017. doi: 10.1115/1.4036132.

[2] D. R. Herber, T. Guo, and J. T. Allison, "PM architectures project." https://github.com/danielrherber/pm-architectures-project.

[3] A. Bayrak, Y. Ren, and P. Papalambros, "Topology generation for hybrid electric vehicle architecture design," *Journal of Mechanical Design*, vol. 138, p. 081401, June 2016. doi: 10.1115/1.4033656.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009.