A data mining and graph theoretic approach to building generic bills of materials

Carol J. Romanowski
Rakesh Nagi
Department of Industrial Engineering
University at Buffalo
Buffalo, NY 14260

Abstract

Generic bills of materials (GBOMs) are useful as configurators and aid in suggesting new product variants. However, current GBOM research does not address the many problems inherent in building GBOMs from legacy data. This research proposes a data mining and graph theoretic approach that partially automates the process. The resulting GBOMs are represented as constrained XML files that incorporate design and manufacturing information such as alternative parts and cost.

Keywords
Data mining, generic bills of material, graph theory, engineering design

1. Introduction
Mass customization results in many variations on the same product. In some cases, variants are targeted to specific customers, markets, or seasons. For instance, the candy industry often has special wrappers for major holidays such as Christmas or Easter. In other cases, companies provide the semblance of uniqueness by allowing customers to configure their products from an array of available choices. For example, one can specify body color, upholstery color and type, engine type, accessory packages, wheel covers, and other options on a new automobile – thus producing near-one-of-a-kind cars. Computer buyers also can configure "unique" systems by choosing different data storage, display, or multimedia options. The outcome of this customization is a proliferation of BOMs that have relatively minor differences in components or structure, but are in the same product family. Companies need ways to control this proliferation, easily configure product options for customers, and suggest new product variants.

In this research, we propose a methodology to address this problem by partially automatic generation of generic bills of material. In our methodology, we make use of graph theory, data mining clustering techniques, and constrained XML. We show how the generic bills are part of a component library that enables engineering designers to quickly find and reuse designs for similar items and parts, thus reducing the effort and time needed in creation of a new product. In the remainder of this paper, Section 2 discusses generic bills of material, graph theory as it relates to trees, clustering algorithms, and constrained XML. Section 3 proposes our methodology for generating GBOMs. In Section 4, we discuss implementation of the methodology and identify areas of future research.

2. Background and literature review
2.1 Generic bills of material (GBOMs)
Introduced by Hegge and Wortmann in 1991 [1], generic bills of material (GBOMs) are a method of encapsulating variant design options and alternate components in a single bill. Hegge's idea of a single specification for all variants of a product family was intended to avoid redundancy while maintaining structural information for assembly, design, and servicing needs.

Jiao et al [2] extend Hegge's GBOM, marrying the generic bill with routing information to form the generic bill-of-materials-and operations (BOMO). Building on Jiao and Tseng's [3] generic variety structure, the authors propose an object-oriented implementation of the BOMO. Their approach allows for alternate operations but not alternate parts. They assume a common structure among product family variants.

Previous approaches to generate GBOMs use group technology (GT) and manual clustering methods. Ramabhatta et al. [4] use an object-oriented, open GT-based product model. Chung and Fischer [5] also used object-oriented modeling for a bill of materials, but do not generalize to a generic BOM. Either approach supports parametric engineering design, giving designers the ability to search for parts or components that fit specified geometric constraints and relationships.

2.2 Graph-based theory of bills of materials (BOMs)

BOMs are essentially rooted, directed attributed graphs, or trees. The end item is the root of the tree, manufactured or assembled components are the nodes, and purchased parts are the leaves. Figure 1 shows Hegge's [1] office chair BOM structure as a tree graph.

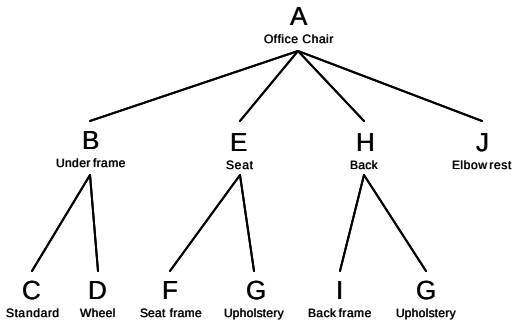

Figure 1: Office chair bill of materials

Assembly BOMs require that components be presented in a particular order when building the end product, and are represented as ordered trees. However, design and manufacturing BOMs are unordered - meaning that the order of nodes, or components, is not significant. For instance, it doesn't matter if we say a car has a body, wheels, and transmission or a car has a transmission, body, and wheels. Unordered trees are more difficult to match than ordered trees; but since assembly BOMs are not as prevalent in practice as design and manufacturing bills, we focus on unordered trees in this research.

BOM trees have some special characteristics that differ from other tree-structured domains. For example, because each end item is unique, the root node of a BOM is always unique; therefore, each and every BOM is unique. Similar BOMs may have common components or parts, but have different structures. As an example, Figure 2 shows two variants of the office chair BOM. Some internal nodes of the trees may be the same, and some may appear in one tree but not another. Additionally, because BOMs can be configured in many ways, parts may appear at different levels in the tree structure. Such structural BOM differences in otherwise similar end products are common in actual manufacturing practice, thus making BOM trees "unruly" as well as unordered. Note that BOMs cannot be represented as strings. The string sequence {ABCE} obtained from the left tree in Figure 2 makes no sense in the manufacturing context.

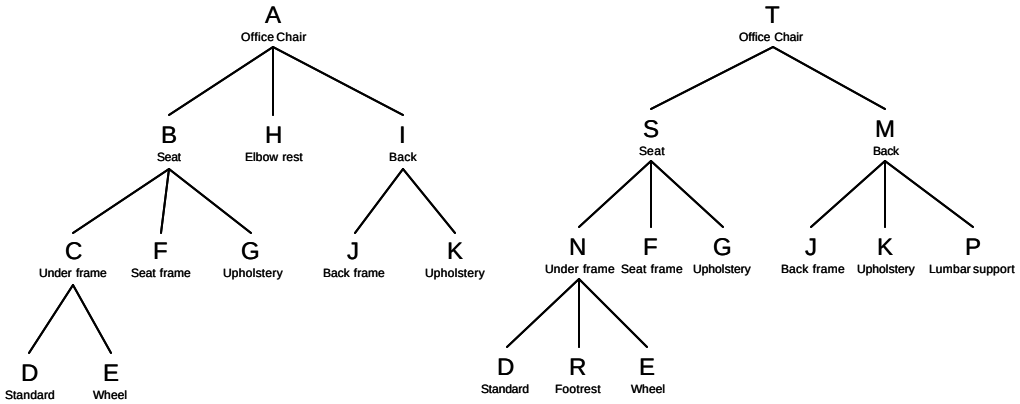

Figure 2: Variants of an office chair

Representing BOMs as trees allows us to use automated matching methods to find similarities and to build the generic bill. We can represent the GBOM as the minimum common supergraph (MCS) of a cluster of graphs; the MCS of two graphs g and g' is equivalent to g g' [6]. In building the tree, we assign a quantity attribute in each node or leaf label to keep the tree compact, and to avoid the difficulty of matching trees with identical repeated subtrees.

2.3 Data mining

Data mining, part of the process of Knowledge Discovery in Databases (KDD), is the search for meaningful and useful patterns and relationships in large databases. Data mining algorithms include machine learning and statistical techniques that are designed especially to work with large amounts of multidimensional data.

Clustering is a form of data mining called "unsupervised learning." The computer is given an unclassified dataset, and attempts to group records with similar attributes together. The clustering problem can be expressed as a linear program, where the objective function for one-dimensional Euclidean distance was first formulated by Fisher [7]and extended by Vinod [8] as

$$Min \sum_{i=1}^{N} \sum_{j=1}^{M} w_i x_{ij} \left(q_i - \bar{q}_j \right)^2 \qquad (1)$$

where

$\bar{q}_j$ = measure assigned to cluster center

q_i = measure assigned to entity i

w_i = weight assigned to entity i

N = number of patterns

M = number of clusters

subject to:

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad i = 1, 2, \ldots, N$$

$$x_{ij} = \begin{cases} 0 & if not assigned to cluster j \\ 1 & if assigned to cluster j \end{cases} \qquad \forall (i, j)$$

2.4 Constrained XML

Currently under development, the constraint-based extension of XML (CobWeb) combines the user-defined tags of XML with the ability to define constraints governing how the various elements of an XML document interact with each other [9]. The addition of constraints provides both an intuitive way of expressing relationships between the different components of an XML document and a method to check that constraints are satisfied within the document itself. For example, in a patient record DTD we might include the constraint that an adult must be at least 18 years of age. The CobWeb parser would check that any patient record containing the tag "adult" satisfies the constraint. This capability is especially important in engineering design, where constraints govern many aspects of the design process.

3. Data mining approach to building GBOMs

We propose a three-stage methodology for finding similar BOMs in a large design database. The first stage involves clustering the BOMs to form groups of products with similar content. Pre-processing for this stage is non-trivial, and required to generalize the content of the BOMs and allow more efficient clustering. For example, consider an automobile engine. Instead of referring to specific part numbers, we can generalize parts as pistons, valves, 4-cylinder block, etc.

The second stage unifies the generalized BOMs within clusters, forming one GBOM per cluster. In the third stage, we represent the GBOMs as constrained XML files, using the constraint capability to specify part numbers, processing information, and configurations, etc. – in effect, undoing the generalization performed in the first stage pre-processing. Figure 3 shows the three steps, culminating in a constrained XML file.

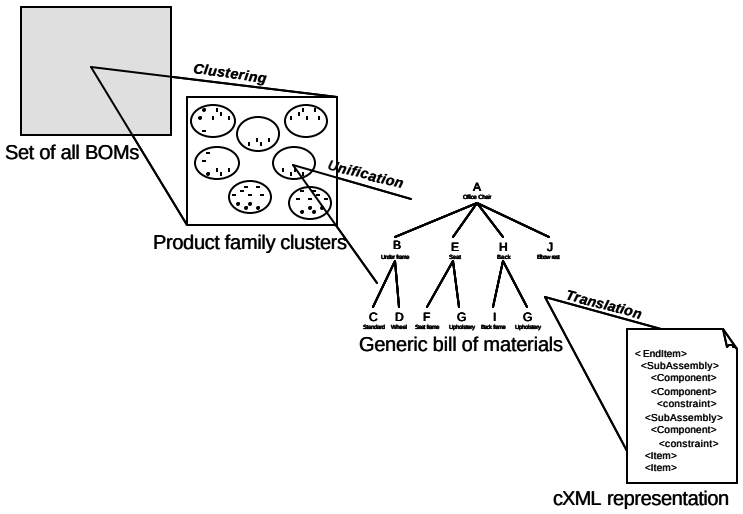

Figure 3: Methodology for generating generic bills of material (GBOMs)

3.1 Pre-processing of BOM trees

Because clustering is NP-complete, many heuristics have been proposed to solve the problem. In our approach, we are able to make use of known characteristics of bills of material, thus making the process more efficient. For instance, BOM trees often have a great degree of fanout in their structure – for example, packaging and final stage assembly hardware that is not part of a complex subassembly or expensive components. These non-critical items can be pruned from the trees before clustering takes place. The resulting tree content and structure therefore reflects the most important items and components in the particular product. We define a simple criticality measure, based on cost and complexity, as the pruning criterion. We define cost as either the item price (for purchased parts) at a leaf node or the sum of child node item prices for a parent node. Complexity refers to the number of child components (out-degree) below a parent node; we assume complex subassemblies are correspondingly critical. In the general case, any relevant weight measures can be substituted for these particular metrics.

Given a tree of order (N+1), with nodes 0,…,N the criticality measure for node *i* is

$$Cr_i = \max\{Cx_i, Co_i\} \tag{2}$$

where

$$Cx_i = \frac{\text{out-degree}_i}{\max_n(\text{out-degree}_n)} \qquad n = \text{nodes } 1,\ldots,N$$

$$Co_i = \frac{\text{cost}_i}{\max_n(\text{cost}_n)} \qquad n = \text{nodes } 1,\ldots,N$$

$$Cr = \{0,1\}$$

By choosing the maximum of normalized cost or complexity values, we avoid inadvertent pruning of high cost purchased items (which would have no children) or complex subassemblies that may be low in cost. The user sets the criticality parameter threshold.

3.2 Similarity measure

To determine the similarity between two trees, we must evaluate two things: node labels and tree/subtree structure. We generalize BOM nodes based on the part description to reduce dimensionality and make clustering more efficient. For space reasons, in this paper we omit the discussion of preprocessing and generalizing trees.

Tree matching algorithms use the concept of edit distance, which assigns a cost to node operations such as insertion, deletion, and relabeling [10]. These node operations transform the trees into isomorphisms, or identical mappings. Similarity is then determined by the total cost of the edit operations (the edit distance). These methods are primarily for classification or determining cluster membership. In clustering, we are indeed interested in grouping like trees together; however, transforming one tree into another is not our goal. We are interested in obtaining a general

description of cluster members that will aid us in grouping new patterns with the proper cluster. In the BOM domain, this general description is the GBOM.

Therefore, instead of edit distance, we propose a measure that calculates the cost of unifying two trees *a* and *b*, where *a* is the template pattern and *b* is to be unified with *a*. The operations available are node insertions, label insertions, and transitive closure; we assume a uniform cost for each. In the case of common nodes, the label from node *i* in tree *b* is added to the label of the corresponding node in tree *a* at no cost.

$$\mathrm{Uni}_{ab} = \sum node\_ins_{ba} + \sum tr_{ba} \tag{3}$$

where

node_ins = insertion of node from tree *b* to tree *a*
tr = transitive closures from tree *b* to tree *a*

3.4 Clustering the trees

Expressed as a linear program, the tree clustering problem then becomes

$$Min \sum_{i=1}^{N} \sum_{k=1}^{M} w_i x_{ik} (Uni_{ik}) \tag{4}$$

where

k = cluster center tree pattern template
Uni_{ik} = cost of unifying tree *i* with template *k*
w_i = weight assigned to entity *i*
N = number of tree patterns
M = number of clusters

subject to:

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad i = 1, 2, \ldots, N$$

$$x_{ik} = \begin{cases} 0 & if not assigned to cluster k \\ 1 & if assigned to cluster k \end{cases} \qquad \forall (i,k)$$

The discussion of clustering algorithm and initial template choice is omitted in this paper because of space limitations.

3.5 Unification of clusters into GBOM

The GBOM for each cluster is formed from the union of BOM trees within the clusters. We choose the tree with the greatest amount of structure as the starting point, and unify over all other cluster members. Where a tree has less structure than the unified tree, we again make use of transitive closure to represent the difference in structure. Node labels of the GBOM carry end item BOM and part numbers. Figure 4 shows a unified GBOM of two office chair bills of material. Note the transitive closure of the lumbar support, P, which appears in one BOM at level 1, and in the second BOM at level 2. Footrest R is an inserted node that appears in one BOM but not the other. This information is passed to the cXML representation of the GBOM as a constraint on Footrest R.

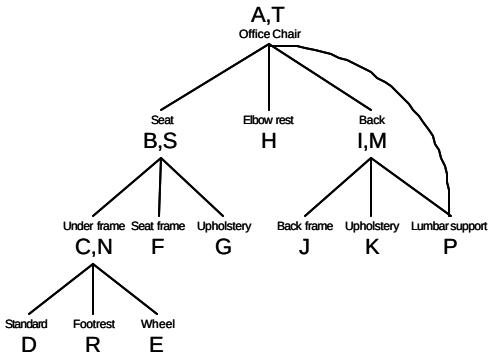

Figure 4: Generic bill of materials for an office chair and its variant

3.5 cXML representation of GBOM

Once the GBOMs are formed, we translate the trees to a constrained XML document. Node labels, attribute values, and constraints are contained in this file. Figure 5 shows a portion of the cXML file for the GBOM in Figure 4; note the constraint on Footrest F, and the transitive closure constraint on Lumbar Support P. Designers can search the cXML files for similar parts or subassemblies to aid in developing new variant products.

```
<EndItem>Chair</EndItem>
        <EndItemNo> A</EndItemNo>
        <EndItemNo> T</EndItemNo>
        <Subassembly>Seat</Subassembly>
                <PartNo>B</PartNo>
                        <EndItemNo> A</EndItemNo>
                <PartNo>S</PartNo>
                        <EndItemNo>T</EndItemNo>
                <Subassembly>Underframe </Subassembly>
                        <PartNo>C</PartNo>
                        <PartNo>N</PartNo>
                        <Item>Standard</Item>
                                <ItemNo>D</ItemNo>
                        <Item>Footrest</Item>
                                <ItemNo>F</ItemNo>
                                <Constraint>[EndItemNo]=T</Constraint>
                        <Item>Wheel</Item>
                                <ItemNo>E</ItemNo>
        <Subassembly>Back</Subassembly>
                <PartNo>I</PartNo>
                <PartNo>M</PartNo>
                <Item>Lumbar Support</Item>
                        <ItemNo>P</ItemNo>
                                <Constraint>TranClos=Yes</Constraint>
                                        <ParentNo>A</ParentNo>
```

Figure 5: Portions of the cXML file for the office chair GBOM

4 Further Work

As a work in progress, the next step in this research is to implement the methodology and test for efficiency and accuracy. Further work includes a proof of the GBOM as minimum common supergraph, determination of a measure to assess cluster quality, a similarity measure for mixed data types, and an accurate search algorithm for cXML files.

Acknowledgements

Carol Romanowski acknowledges the support of the Engineering Research Program of the Office of Basic Energy Sciences at the Department of Energy. Rakesh Nagi acknowledges the support of the National Science Foundation under career grant DMI-9624309.

References

1. Hegge, H. M. H., and Wortmann, J.C., 1991, “Generic bill-of-material: a new product model,” *International Journal of Production Economics*, 23, 117-128.
2. Jiao, J., Tseng, M.M, Ma, Q., and Zou, Y., 2000, “Generic Bill-of-Materials-and-Operations for high-variety production management,” *Concurrent Engineering-Research & Applications*, 8(4), 297-321.
3. Jiao, J. and Tseng, M.M., 1999a, “Methodology of developing product family architecture for mass customization,” *Journal of Intelligent Manufacturing*, 10(1), 3-20.
4. Ramabhatta, V., Lin, L., and Nagi, R., 1997, “Object Hierarchies to aid Representation and Variant Design of Complex Assemblies in an Agile Environment,” *International Journal of Agile Manufacturing*, 1(1), 77-90.
5. Chung, Y. and Fischer, G. W., 1994, “A conceptual structure and issues for object-oriented bill of materials (BOM) data model,” *Computers and Industrial Engineering*, 26(2), 321-339.
6. Bunke, H., 2000, “Recent developments in graph matching," *Proceedings of the International Conference on Pattern Recognition (ICPR'00)*, September 3-8, Barcelona, Spain.
7. Fisher, W. D., 1958, “On grouping for maximum homogeneity,” *Journal of the American Statistical Association*, 53(284), 789-798.
8. Vinod, H., 1969, “Integer programming and the theory of grouping,” *Journal of the American Statistical Association*, 64(326), 506-519.
9. McKernon, T. J. and Jayaraman, B., 2000, "CobWeb: A Constraint-based XML for the Web," Technical Report, Department of Computer Science, University at Buffalo.
10. Shasha, D., Wang, J., Zhang, K., and Shih, F., 1994, “Exact and Approximate Algorithms for Unordered Tree Matching,” *IEEE Transactions on Systems, Man and Cybernetics*, 24(4), 668-678.