Data mining in an engineering design environment: OR applications from graph matching[1]

Carol J Romanowski, Rakesh Nagi[*]
Department of Industrial Engineering
University at Buffalo, SUNY
Buffalo, New York 14260

Moises Sudit
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York 14623

Abstract
Data mining has been making inroads into the engineering design environment – an area that generates large amounts of heterogeneous data for which suitable mining methods are not readily available. For instance, an unsupervised data mining task (clustering) requires an accurate measure of distance or similarity. This paper focuses on the development of an accurate similarity measure for bills of materials (BOM) that can be used to cluster BOMs into product families and subfamilies. The paper presents a new problem called Tree Bundle Matching (*TBM*) that is identified as a result of the research, gives a non-polynomial formulation, a proof that the problem is NP-Hard, and suggests possible heuristic approaches.

Scope and Purpose

In a typical life cycle of an engineering project or product, enormous amounts of diverse engineering data are generated. Some of these include Bills-Of-Materials (BOM), product design models in CAD, engineering drawings, manufacturing process plans, quality and test data, and warranty records. Such data contain information crucial for efficient and timely development of new products and variants; however, this information is often not available to designers. Our research employs data mining methods to extract this design information and improve its accessibility to design engineers. This paper focuses on one aspect of the overall research agenda, clustering BOMs into families and subfamilies. It extends previous work on a graph-based similarity measure for BOMs (a class of unordered trees) by presenting a new Tree Bundle Matching (*TBM*) problem, and proves the problem to be NP-hard. The overall contribution of this work is to demonstrate the OR applications from graph matching, stochastic methods, optimization, and others to data mining in the engineering design environment.

Keywords: Bills Of Material, unordered trees, similarity measure, clustering, weighted bipartite matching, matching problems.

[1] Accepted for publication in Computers & Operations Research, December 2004.
[*] To whom correspondence should be directed. Email: nagi@buffalo.edu, Fax: 716-645-3302.

1. Introduction

Engineering design is a multi-disciplinary, multi-dimensional, non-linear process. Once performed by a solitary engineer wielding drafting tools and vellums, today's design and development is likely to be a heavily computerized, team process – concurrent engineering – performed with CAD programs, computerized optimization and analysis algorithms, word processing and spreadsheet software, and e-mail. With the advent of cheap storage and fast computers, the amount of engineering data generated during product design and development accumulates beyond the ability of humans to process the data into useful knowledge.

Yet, accessing and distilling the valuable knowledge hidden in this vast amount of information is crucial. The demands of the current business climate require companies to be agile and proactive, moving products quickly from concept to market. A key element of time-to-market reduction is the ability to use existing knowledge and designs to generate new variations of existing products.

Data mining methodologies have been specifically developed for these types of situations, where the sheer volume of data is overwhelming for both human and existing methods. However, in the case of engineering design, the nature of the data requires modification of existing algorithms or creation of new methodologies. Characteristically, engineering design data appear in many forms: numerical, textual, graphical, imaged, and a mixture of two or more of these types. Therefore, while data mining algorithms have been specifically written to effectively analyze large datasets, the engineering data often cannot be simply "plugged in" to these programs.

Operations research formulations and methods, in conjunction with traditional data mining statistical and machine learning algorithms, offer new possibilities for characterizing, defining and solving the types of problems posed by design data. For instance, calculating the similarity between two designs for clustering purposes is a matching task; we prove, via OR methods, that the true problem is NP-hard. This application of operations research helps to define the structure and complexity of the problem, and to suggest possible heuristic approaches that might not otherwise be evident.

In this paper, we discuss clustering of engineering design data from a data mining and operations research perspective. Section 2 explains the similar design matching problem and an algorithm that calculates distances between BOMs, values that will be used in a clustering algorithm. Section 3 presents a new Tree Bundle Matching problem (*TBM*), gives a proof that *TBM* is NP-hard, and suggests future directions for solutions. Section 4 concludes the paper.

2. Design similarity based on BOM tree matching

In previous work [1, 2] we introduced a design support system to facilitate the search for similar designs and thus reduce the interval between concept and product launch. The system is founded on a library of existing products; companies may, over the course of even a few years, generate thousands of designs, most of which are variants of a base product – in short, "variations on a theme." Our approach is to reduce the search space by using data mining methods such as clustering and association mining. We cluster these designs, represented as bills of materials (BOMs), into families and subfamilies; unify each cluster into a single entity (called a *generic bill of material* (GBOM) [3, 4, 5, 6]) that encapsulates the variations in content and structure within each family; and use rules induced from association mining to add design rules and constraints to the GBOM.

A BOM is the hierarchical, structured representation of a product, containing critical information such as components, raw materials, quantities, instructions for manufacture, and consumable items [7]. BOMs capture the make-up, content, and structure of complex products from these engineering domains. The major purpose for BOMs is to define the recursive parent-child relationships between the end item, its components or subassemblies, and the raw (or purchased) materials they contain.

BOMs can be depicted as rooted, unordered trees – meaning that the order of nodes, or components, is not significant. The end item, or finished product, is the root of the tree; manufactured or assembled components are the nodes; and purchased parts or raw materials are the leaves. Different engineers may build completely identical end items with very different BOM structures; since there is no common rule or template to follow, the engineer develops the BOM based on individual understanding of how the product is manufactured or assembled. Thus, trees representing otherwise identical end items can have very different topologies, from relatively flat trees (not much different than mere parts lists) to highly structured, multi-level trees. BOM trees may differ in three ways: (i) structural differences such as the number of intermediate parts, parts at different levels, and parts with different parents, (ii) differences in component labels, and (iii) differences in both components and structure.

For example, Figure 1 shows an office chair (*A*) and a variant (*A*'). Note the lumbar support (P); in Tree 1, on the left, it appears as the child of the end item, Office Chair (*A*). In Tree 2, on the right, the lumbar support is included as part of the subtree rooted at *M*, which represents the chair back subassembly (Note: all arcs in BOMs represent *AND* relationships). The change in the subassembly label from *I* to *M* reflects the new part number generated by adding the lumbar support to the original subassembly.

Therefore, similar BOMs may have the same components or parts, but have different structure, with some parts appearing at one level in one tree and at another level – or with a different parent – in a second tree. Additionally, BOMs may have similar structure but different components. These situations are common in practice.

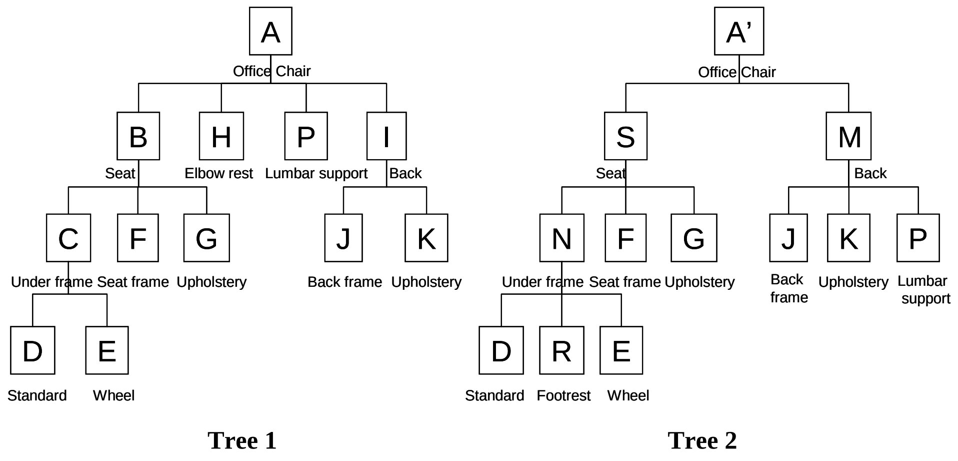


Figure 1: Variants of an office chair

The notion of similarity in BOMs is rooted more in content than in topology; the commonality of content lies in similarities between the component parts in the two BOMs. However, we do want to capture the differences in structure. Quantifying the similarities between BOMs is important in identifying similar end items whose designs can be re-used in new products. Research has been done on similarity measures for ordered trees [8, 9, 10, 11, 12], but these methods are not consistent for unordered trees. Unit cost editing operations are typically used to determine distance between unordered trees [13, 14, 15, 16], but we show in [17] that these operations do not give accurate distances for BOMs.

Also in [17] we present a polynomial time decomposition-reduction algorithm (DeRe) for computing the weighted symmetric difference between two BOM trees. Essentially, the algorithm compares single level subtrees (SLTs), removes those that match exactly in topology and content, and determines the distances between the remaining SLTs using weighted bipartite matching. These distances are "rolled up" to the root node; their sum represents the total distance between the two BOM trees. In the following subsection, we give a simple example for illustrative purposes.

2.1 Example of the DeRe algorithm

Consider the trees in Figure 2:

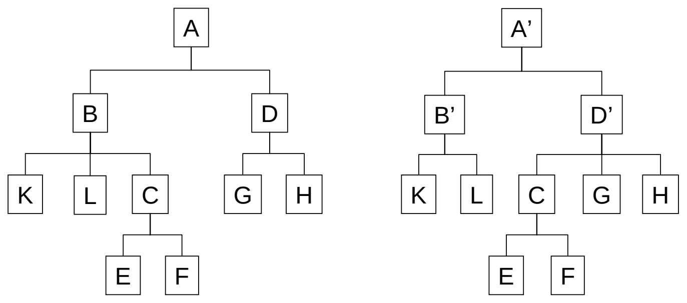


Figure 2: Two BOM trees

There are two terminal SLTs in Tree *A* (trees whose children are only leaf nodes): one rooted at *C* and one rooted at *D*. In Tree *A*' there are also two terminal SLTs: one rooted at *C* and one rooted at *B*'. The two SLTs rooted at *C* are exact matches. Therefore, we can remove their children *E* and *F*. Figure 3 shows the reduced tree.

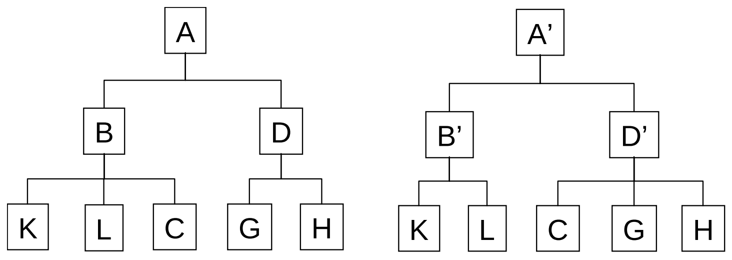


Figure 3: Trees after removing exact matches

We now look at the four SLTs rooted at *B*, *D*, *B*', and *D*'. We assume for this example that the distance between differently labeled nodes is 1. Consider the SLT rooted at *B*. The distance between *B* and *B*' is 1, since *K* and *L* match exactly and *C* has no match in *B*'. The distance between *B* and *D*' is 2, since the *C* nodes match exactly but the remaining nodes do not.

Consider the SLT rooted at *D*; the distance between *D* and *B*' is 2. The distance between *D* and *D*' is 1, which reflects the parent difference of *C*.

The last remaining SLTs are those rooted at *A* and *A*'. The best match for SLT *B* is *B*', at a distance of 1; the best match for *D* is *D*', also at a distance of 1. So, the distance between *A* and *A*' is the sum of the distances of the matched pairs (*B*,*B*') and (*D*,*D*') = 1 + 1 = 2.

2.2. Limitation of the DeRe algorithm

As seen above, the weighted bipartite matching approach can overstate the total weighted distance between two BOM trees. For instance, a BOM tree with few intermediate subassemblies may have exactly the same raw material or purchased part content, but a flatter structure, than another BOM tree (see Figure 4, where Tree 1 is a less structured version of Tree 2). Using editing operations, the distance between Tree 1 and Tree 2 is 6 (removing two nodes from subtree *B*, inserting the subtree parent node and the two leaf nodes as siblings of *B*, and changing *B*'s label). Using the weighted symmetric difference measure, and matching subtree *B* with subtree *C*, the distance is 5 (two unmatched nodes from subtree *B*, the subtree rooted at *D* and its two children). But, the true distance between Tree 1 and Tree 2 is merely the presence of an additional subassembly root node; therefore the DeRe algorithm, while more accurate than editing operations, overstates this distance. A closer value would be obtained if we could match, or bundle, subtree *B* with more than one other subtree. However, this extended capability introduces more complexity to the overall matching problem, which is discussed in Section 3.

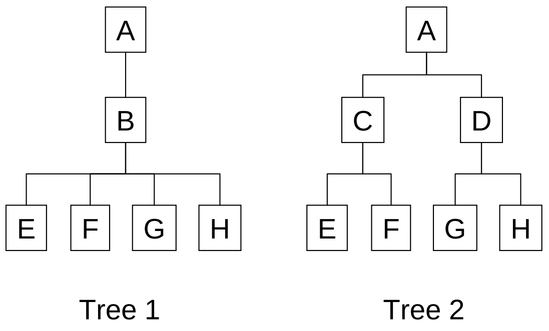


Figure 4: Two similar trees

3. Analysis of the TBM problem

In this section we present the mathematical formulation for the tree bundle matching problem discussed in the previous section, and prove that the problem is NP-hard.

Let T and R be two trees that are considered for a tree matching problem. Let B^T be the set of bundles created from tree T, where bundles are SLTs or groups of SLTs of T. Let $B_i^T \subseteq T$ be the individual bundles of B^T so that $B^T = \{B_1^T, B_2^T, ..., B_i^T, ..., B_m^T\}$. Similarly, let B^R be the set of bundles created from tree R and $B_j^R \subseteq R$ be the individual bundles of B^R, so that $B^R = \{B_1^R, B_2^R, ..., B_j^R, ..., B_n^R\}$). Let w_{ij} represent the "closeness," or similarity factor, between elements $i \in B^T$ and elements $j \in B^R$. We wish to find the matching of elements of B^T to elements of B^R such that the sum of the match elements is maximized (i.e., we maximize the similarity of the trees). We also define, for each element $t \in$T, the set $I_t = \{1 \le i \le m : t \in B_i^T\}$, which represents all bundles in B^T containing t. Similarly, for each element $r \in R$ we define the set $I_r = \{1 \le j \le n : r \in B_j^R\}$, representing all bundles in B^R containing r.

Mathematically, we can represent the Tree Bundle Matching (*TBM*) problem as follows:

$$x_{ij} = \begin{cases} 1 & \textit{if bundle } i \in B^T \textit{ is matched with bundle } j \in B^R \\ 0 & \textit{otherwise} \end{cases}$$

$$\begin{array}{llllr}
Max & \sum_{i \in B^T} \sum_{j \in B^R} w_{ij} x_{ij} & & & \\
s.t.: & \sum_{j \in B^R} x_{ij} & \le 1 & \forall i \in B^T & (1a) \\
 & \sum_{i \in B^T} x_{ij} & \le 1 & \forall j \in B^R & (1b) \\
 & \sum_{i \in I_t} \sum_{j \in B^R} x_{ij} & \le 1 & \forall t \in T & (2a) \\
 & \sum_{j \in I_r} \sum_{i \in B^T} x_{ij} & \le 1 & \forall r \in R & (2b) \\
 & x_{ij} & = 0 \ or \ 1 & \forall i \in B^T, \forall j \in B^R &
\end{array}$$

Constraints (1*a*) and (1*b*) represent the tree matching problem among predefined bundles, while constraints (2*a*) and (2*b*) allow any element of the two trees being considered to belong to at most one bundle. Clearly, if we consider constraints (1*a*) and (1*b*) only, the problem becomes a Weighted Bipartite Matching problem (*WBM*) which is polynomially-solvable. So, a natural research question is whether, with the addition of constraints (2*a*) and (2*b*), *TBM* remains a tractable problem. Additionally, if the problem is tractable, we would like to know if it can be viewed as a different matching structure.

Some of the other matching problems that we can consider in understanding the solvability of *TBM* are:

- Weighted Matching (*WM*) problem (not necessarily bipartite), and
- Weighted Perfect Matching (*WPM*) problem (every vertex has to be incident to a matched edge).

Even though these problems are polynomially equivalent (they can be transformed to each other in polynomial number of steps and size), their structures might give us some insights on the behavior of *TBM*. Moreover, given that *WBM* is a relaxation of *TBM* and a special case of *WM* and *WPM*, it might allow us to obtain tighter bounds. Finally, both *WM* and *WPM* are two of the few structures having discrete phenomena that can be solved in polynomial time, and that do not have the Integrality Property (i.e., the convex hull of the linear programming relaxation has integer extreme points).

When determining the existence of an efficient (polynomial-time) algorithm for *TBM*, we need to first investigate the possibility that such an algorithm cannot exist. For this proof we will study the relation of *TBM* to another related problem, 3-Dimensional Matching (3*DM*), which is NP-Complete. In the 3-Dimensional Matching (3*DM*) problem, where the input is a graph, we are trying to find a set of disjoint triples covering all vertices. The problem can be written as an integer program:

$$
\begin{array}{llllr}
Max & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk} x_{ijk} & & & \\
s.t.: & \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} & \leq 1 & i=1,\ldots,n & (3) \\
 & \sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} & \leq 1 & j=1,\ldots,n & (4) \\
 & \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} & \leq 1 & k=1,\ldots,n & (5) \\
 & x_{ijk} & = 0 \ or \ 1 & \forall i, \forall j, \forall k &
\end{array}
$$

To understand the relationship between 3*DM* and *TBM*, we introduce a special case of the problem in which the two elements of the two trees to be considered can be defined as $T = \bigcup_{i=1}^{n} i \ and \ R = \bigcup_{j=1}^{n} j \bigcup_{k=1}^{n} k$. The bundles to be formed for each tree can be represented as:

$B^T = \{1,\ldots,i,\ldots,n\}$ $(with \ B_i^T = \{i\})$ and $B^R = \{\{1,1\},\ldots,\{j,k\},\ldots,\{n,n\}\}$ $(with \ B_{jk}^R = \{j,k\})$

Intuitively, B^T represents bundles of singletons of one of the sets to be matched, while B^R represents bundles of all possible pairs of the other two sets and the problem, referred to hereafter as *TBM*3, can be written as:

$$
\begin{array}{llllr}
Max & \sum_{i \in B^T} \sum_{\{j,k\} \in B^R} w_{ijk} x_{ijk} & & & \\
s.t.: & \sum_{\{j,k\} \in B^R} x_{ijk} & \leq 1 & \forall i \in B^T & (6a) \\
 & \sum_{i \in B^T} x_{ijk} & \leq 1 & \forall \{j,k\} \in B^R & (6b) \\
 & \sum_{i \in I_t} \sum_{\{j,k\} \in B^R} x_{ijk} & \leq 1 & \forall t \in T & (7a) \\
 & \sum_{\{j,k\} \in I_r} \sum_{i \in B^T} x_{ij} & \leq 1 & \forall r \in R & (7b) \\
 & x_{ijk} & = 0 \ or \ 1 & \forall i \in B^T, \forall \{j,k\} \in B^R &
\end{array}
$$

Theorem 1: **3*DM* can be polynomially reduced to *TBM*3**

Proof: The size of *TBM*3 is polynomial in the size of 3*DM*, since the sizes of the trees (*n and 2n*) as well as the bundles (*n and n*^{2}) are polynomial on the input of the 3*DM* problem. Given the definition of the bundles, the objective function of *TBM*3 can be shown to be identical to the 3*DM* problem:

$$\sum_{i\in B^T}\sum_{\{j,k\}\in B^R} w_{ijk}x_{ijk} = \sum_{i\in\{1,\ldots,n\}}\sum_{\{j,k\}\in\{\{1.1\},\ldots,\{n,n\}\}} w_{ijk}x_{ijk} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} w_{ijk}x_{ijk} \xleftrightarrow{c_{ijk}=w_{ijk}} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}x_{ijk}$$

So, to finalize this transformation, it suffices to show that the feasible regions are also identical. Consider constraint (6*a*) of *TBM*3, which can be shown to be equivalent to constraint (3) of 3*DM*:

$$\sum_{\{j,k\}\in B^R} x_{ijk} \le 1,\ \forall i\in B^T \Leftrightarrow \sum_{\{j,k\}\in\{\{1,1\},\ldots,\{n,n\}\}} x_{ijk} \le 1,\ \forall i\in\{1,\ldots,n\} \Leftrightarrow \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} \le 1,\ i=1,\ldots,n$$

Any feasible solution satisfying constraints (3) and (4) in 3*DM* will also satisfy constraint (6b) in *TBM*3. Consider summing constraint (6*b*) over *k*; then, the resulting set of constraints are

$\sum_{i\in B^T}\sum_{\{j,k\}\in\{\{1.k\},\ldots,\{n,k\}\}} x_{ijk} \le n, \forall k=1,\ldots,n \Leftrightarrow \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} \le n, \forall k=1,\ldots,n$. Similarly, summing constraint (6*b*) over *j* gives a resulting set of constraints

$$\sum_{i\in B^T}\sum_{\{j,k\}\in\{\{j.1\},\ldots,\{j,n\}\}} x_{ijk} \le n, \forall j=1,\ldots,n \Leftrightarrow \sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} \le n, \forall j=1,\ldots,n\ .$$

Constraint (7*a*) is redundant to constraint (6*a*) for this special case:

$$\sum_{i\in I_t}\sum_{\{j,k\}\in B^R} x_{ijk} \le 1, \forall t\in T \xleftrightarrow{I_i=\{i\}} \sum_{i\in\{i\}}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk}, \forall i\in T \Leftrightarrow \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk}, \forall i=\{1,\ldots,n\}$$

Finally, we need to consider constraint (7*b*). Let *R* be divided in two disjoint sets $R^J=\{1,\ldots,j,\ldots,n\}$ and $R^K=\{1,\ldots,k,\ldots,n\}$, such that $R=R^J\cup R^K$. Then, we can divide the set of constraints into two sets:

$$\sum_{\{j,k\}\in I_r}\sum_{i\in B^T} x_{ij} \le 1, \forall r \in R^J \xleftrightarrow{I_r=\{\{j,k\}:j=r \text{ and } k=1,\ldots,n\}} \sum_{k=1}^{n}\sum_{i=1}^{n} x_{irk} \le 1, \forall r = 1,\ldots,n \qquad (7bJ)$$

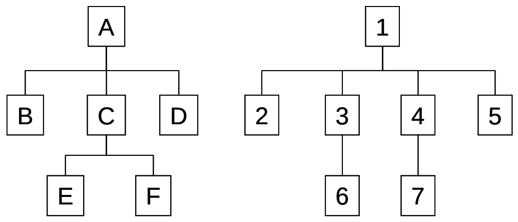
$$\sum_{\{j,k\}\in I_r}\sum_{i\in B^T} x_{ij} \le 1, \forall r \in R^K \xleftrightarrow{I_r=\{\{j,k\}:k=r \text{ and } j=1,\ldots,n\}} \sum_{j=1}^{n}\sum_{i=1}^{n} x_{irj} \le 1, \forall r = 1,\ldots,n \qquad (7bK)$$

If we compare (7*bJ*) and (7*bK*) with (4) and (5) respectively, we see that the constraints are identical, showing that 3*DM* polynomially reduces to *TBM*3.

A direct result from Theorem 1 is that *TBM* is NP-Hard. Moreover, given that the special structure of *TBM*3 only includes bundles with singletons or pairs, there will be no simpler version of the problem for which we can hope to find a polynomial-time algorithm. Therefore, our research will concentrate in finding fast heuristic optimization procedures, such as inexact graph matching approaches, that perform efficiently.

3.1 A numerical example

Prior to developing an efficient heuristic our interest is to determine the solvability of the TBM using a standard mixed integer solver. We present the details of a small numerical example in Figure 5 (showing the trees and the distance matrix). The TBM was formulated and solved using CPLEX 7.1 on an SGI Platform running IRIX 6.3. This problem was able to solve in a fraction of a second. The optimal match resulted in a match of *b* with 2, *c* with bundle {3,4}, *d* with 5, *e* with 6, and *f* with 7, providing an objective value of 4.75. In the SLT matching approach of Section 2, the best one could do is match c with either 3 or 4, providing an objective value of 4.5. This simple example illustrates the efficacy of the *TBM* to more accurately represent the similarity between two BOMs. It also indicates that small size problems can be solved using a standard solver. It is noted that for this example, constraints (1*a*) and (1*b*) are redundant.



	2	3	4	5	6	7	23	24	25	26	27	34	35	37	45	46	56	57
Equivalent	B	C		D	E	F	BC2	BC2	BD	BE	BF	C	C2D	C2F	C2D	C2E	DE	DF
B	1	0	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0	0	0	0	0	0	0
C	0	0.5	0.5	0	0	0	0.25	0.25	0	0	0	0.75	0	0	0.25	0	0	0
D	0	0	0	1	0	0	0	0	0.5	0	0	0		0		0	0.5	0.5
E	0	0	0	0	1	0	0	0	0	0.5	0	0	0	0	0	0.5	0.5	0
F	0	0	0	0	0	1	0	0	0	0	0.5	0	0	0.5	0	0	0	0.5
BC	0.5	0.25	0.25	0	0	0	0.5	0.5	0.5	0	0	0.5	0.5	0.25	0.25	0.25	0.25	0.25
BD	0.5	0	0	0.5	0	0	0.5	0.5	0.75	0.5	0.5	0	0.5	0	0.5	0	0.5	0.5
BE	0.5	0	0	0	0.5	0	0.5	0.5	0.5	0.75	0.5	0.25	0	0	0	0.5	0.5	0
BF	0.5	0	0	0	0	0.5	0.5	0.5	0.5	0.5	0.75	0.25	0	0.5	0	0	0	0.5
CD	0	0.25	0.25	0.5	0	0	0.25	0.25	0.5	0	0	0.5	0.5	0.25	0.5	0.25	0.5	0.5
DE	0	0	0	0.5	0.5	0	0	0	0.5	0.5	0	0.25	0.5	0	0.5	0.5	0.75	0.5
DF	0	0	0	0.5	0	0.5	0	0	0.5	0	0.5	0.25	0.5	0.5	0.5	0	0.5	0.75

Figure 5: A numerical example of TBM

3.2 Potential heuristic approaches

While small-dimensioned problems can be solved to optimality using standard solvers, for complex BOM trees it is desirable to seek heuristic approaches. The objective of this brief paper is to introduce some ideas along which such heuristics can be developed. Specific algorithm development and performance evaluation are recommended for further study.

We first note that the complication of the TBM comes from two sources: (i) combinatorial number of bundles of single level trees (SLTs) to be considered in the matching, and (ii) constraints (2a) and (2b) that require an elemental SLT to be matched in at most one bundle. The former increases the dimension of the problem, while the latter results in NP-hardness.

Therefore, our first idea is to reduce the state space of the bundles to be matched. This draws from the observation that in the domain of BOMs, a SLT in one BOM is likely to match (with strictly positive value) very few SLTs in the other BOM because matching requires similar child components. Further, there is no advantage to bundling one or more non-matching SLTs with partially matching SLT(s) because they only “dilute” the weight associated with the resulting match. Therefore, we can reduce the problem dimension by creating only bundles of strictly positively matched element SLTs for a target SLT or bundle. An appropriate threshold match value can also be selected for further reduction in number of bundles created.

The second idea is to solve the TBM without constraints (2) using a weighted bipartite matching algorithm. If an elemental SLT is matched in more than one bundle, constraint resolution can be performed along standard optimization techniques. Alternately, we introduce a dummy node with perfect 1-match to the pair of bundles that violate constraints (2), retaining the bipartite structure. A high arc weight ensures that the perfect-1 match is achieved and the WBM algorithm is rerun. This process can be repeated until constraints (2) are respected. Further effort needs to be invested in algorithm development and rigorous performance studies.

4. Conclusion

In this paper, we discussed the search for similar designs and identified a new OR problem, the tree bundle matching problem, and provided a proof that the problem is NP hard. We presented a numerical example to show the problem is solvable for small-sized problems using a standard solver. Ideas for heuristic development are also presented.

The synergy between OR and data mining is clearly evident, as shown by the contribution OR methods make to the development of the distance measure used in this research. The outcome of the overall data mining task is critically dependent on the accuracy of the distances that are inputs to the clustering algorithm.

The field of data mining, as it matures, must draw not only from its statistics and machine learning roots, but also incorporate knowledge and approaches from related fields. The resulting merger makes it possible to solve old problems better and new ones for the first time.

References

1. Romanowski C. J., Nagi R. A Data Mining-Based Engineering Design Support System: A Research Agenda. In: Braha, D (Ed.) *Data Mining for Design and Manufacturing: Methods and Applications*. Dordrecht: Kluwer Academic Publishers, 2001. p. 235-254.
2. Romanowski C. J., Nagi R. A Data Mining Approach to Integrating Product Life Cycle Information in Engineering Design. *Proceedings of the 10th Industrial Engineering Research Conference*, 2001.
3. Romanowski C. J., Nagi R. The search for similar parts in variant design. *Proceedings of the 12th Industrial Engineering Research Conference*, 2003.
4. 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.
5. Jiao, J. and M.M. Tseng, M.M., 1999a, Methodology of developing product family architecture for mass customization. *Journal of Intelligent Manufacturing*, 10(1): 3-20.
6. Jiao, J., M.M. Tseng, Q. Ma, and Y. Zou, 2000, Generic Bill-of-Materials-and-Operations for high-variety production management. *Concurrent Engineering-Research & Applications*, 8(4): 297-321.
7. Orlicky, J. *Material Requirements Planning*. New York: McGraw-Hill Book Company, 1975.
8. Kilpelainen P., Mannila H. Ordered and Unordered Tree Inclusion. *SIAM Journal on Computing* 1995; 24: 340-356.
9. Shasha D., Zhang K. Tree Pattern Matching. In: Apostolico A. (Ed.). *Pattern Matching Algorithms*. London: Oxford University Press, 1998. p. 341-369.
10. Tai KC. The tree-to-tree correction problem. *Journal of the ACM* 1979; 26(3): 422-433.
11. Hoffmann CM, O'Donnell MJ. Pattern Matching in Trees. *Journal of the ACM* 1982; 29(1): 68-95.
12. Amoth TR, Cull P, Tadepalli P. Exact Learning of Tree Patterns from Queries and Counterexamples. In: *Proc. 12th Annual ACM Conference on Computational Learning Theory (COLT)*, 1999. p. 323-332.
13. Shasha D, Wang J. T.-L., Zhang K, and Shih F. Exact and Approximate Algorithms for Unordered Tree Matching. *IEEE Transactions on Systems, Man and Cybernetics* 1994; 24: 668-678.
14. Jiang T, Wang L, Zhang K. Alignment of trees - an alternative to tree edit. *Theoretical Computer Science* 1995; 143: 137-148.
15. Chawathe S, Garcia-Molina H. Meaningful Change Detection in Structured Data. In: *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1997. p. 26-37.
16. Zhang K, Jiang T. Some MAX SNP-hard results concerning unordered labeled trees. *Information Processing Letters* 1994; 49:249-254.
17. Romanowski C. J., Nagi R. On clustering bills of materials: A similarity/distance measure for unordered trees. Accepted to *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 2004.