Mining Manufacturing Databases to Discover the Effect of Operation Sequence and Operation Setting on the Product Quality

Lior Rokach[1], Roni Romano[2], Oded Maimon[2]

[1]Department of Information Systems Engineering, Ben Gurion University, Beer Sheva 84105, Israel

[2]Department of Industrial Engineering, Tel-Aviv University, Tel-Aviv, Israel

Abstract

Data mining techniques can be used for discovering interesting patterns in complicated manufacturing processes. These patterns are used to improve manufacturing quality. Classical representations of quality data mining problems usually refer to the operations settings and not to their sequence. This paper examines the effect of the operation sequence on the quality of the product using data mining techniques. For this purpose a novel decision tree framework for extracting sequence patterns is developed. The proposed method is capable to mine sequence patterns of any length with operations that are not necessarily immediate precedents. The core induction algorithmic framework consists of four main steps. In the first step, all manufacturing sequences are represented as string of tokens. In the second step a large set of regular expression-based patterns are induced by employing a sequence patterns. In the third step we use feature selection methods to filter out the initial set, and leave only the most useful patterns. In the last stage we transform the quality problem into a classification problem and employ a decision tree induction algorithm. A comparative study performed on benchmark databases illustrates the capabilities of the proposed framework.

1. Introduction

In many modern manufacturing plants, data that characterize the manufacturing process are electronically collected and stored in the organization's databases. Thus, data mining tools can be used for automatically discover interesting and useful patterns in the manufacturing processes. These patterns can be subsequently exploited to enhance the whole manufacturing process in such areas as defect prevention and detection, reducing flow-time, increasing safety, etc.

This paper focuses on mining quality-related data in manufacturing. Quality can be measured in many different ways. Usually the quality of batches of products is measured and not that of a single product. The quality measure can either have nominal values (such as "Passed"/"Not Passed") or continuously numeric values (Such as the number of good chips obtained from silicon wafer). Even if the measure is numeric, it can still be reduced to a sufficiently discrete set of interesting ranges. In the cases that we examined, the goal was to find the relation between the quality measure (target attribute) and the input attributes (the manufacturing process data).

Classification methods can be used to improve the learning curve both in the learning pace, as well as in the target measure that is reached at the mature stage. The idea is to find a classifier that is capable of predicting the measure value of a certain product or batch, based on its manufacturing parameters. Subsequently, the classifier can be used to set up the most appropriate parameters or to identify the reasons for bad measures values.

When data mining is directed towards improving manufacturing process, there are certain distinctions that should be noted compared to the classical methods employed in quality engineering, such as the experimental design. Data mining is considered as a "secondary data analysis of large databases" (Hand, 1998). The term “secondary” emphasizes the fact that the primary purpose of the database was not data analysis. That is to say, there is no control whatsoever on the data collected. Other classical methods, such as control charts, aim to monitor the process and not to infer the relationship between the target attribute and the input attributes.

Moreover the volume of the collected data makes it impractical to explore and detect intricate relations between the parameters of different processing steps using standard statistical procedures. Also, comprehensive manual analysis of the voluminous data becomes prohibitive, especially when on-line monitoring and control of the manufacturing process are considered (Braha and Shmilovici, 2003)

While there are many works that analyze the effect of the operation parameters setting (such as slicing speed) on the quality (see for instance Rokach and Maimon, 2006), da Cunha *et al.* (2006) have been the first to analyse the effect of the production sequence. More specifically data mining is used to determine the sequence of assemblies that minimizes the risk of producing faulty products. As motivated by da Cunha *et al.* (2006) the sequence analysis is particularly important for assemble-to-order production strategy. This strategy is useful when the basic components can be used as building blocks of a variety of end products.

In fact current manufactures are obligated to manage a growing product portfolio in order to meet customer needs. It is not feasible for the manufacturer to build all possible end products in advance, thus the assembly is performed only when the customer order is received (da Cunha *et al.*, 2006). Still in order to compete other manufactures, the assembly lead-time should be short as much as possible and any rework due to quality problems should be avoided. Because the quality tests of the components can be performed in advance, the quality of the final assembly operations should be the focus. da Cunha *et al.*, (2006) provide an industrial example of electrical wire harnesses in the automobile industry. There are millions of different wire harnesses for a unique car model, mainly because wire harnesses control many functions (such as the electrical windows) and because every function has different versions depending on other parameters such as engine type.

This set of wires and connectors transmits electricity and information between different devices all over the car (see Figure 1).The functions (airbag, electrical windows, headlights’ control, etc.) are performed by combination of different wires and connectors. To illustrate the diversity of this product, consider a standard wire harness in a middle range car. This wire harness performs 15 different functions. Depending on the silhouette and the motor, these functions appear in different

versions (up to 9). Potential diversity is then about 7 millions of different wire harnesses for a unique car model.

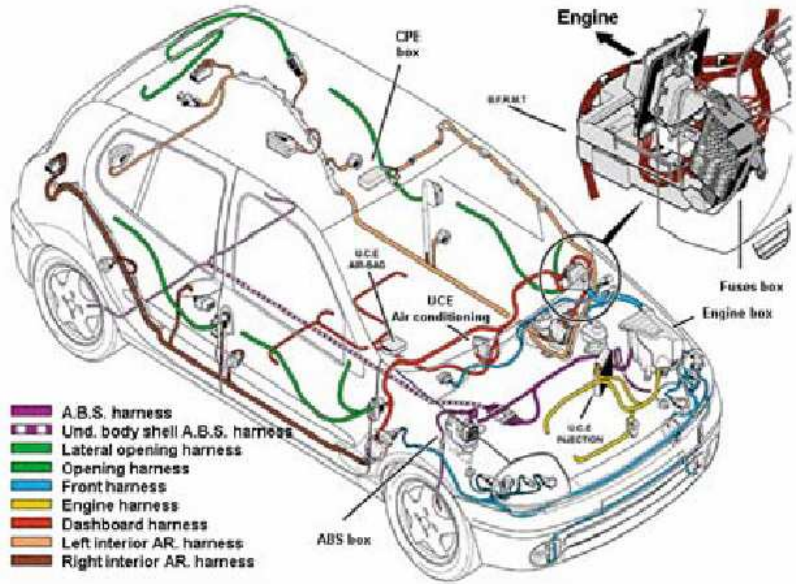

Figure 1:Wire harnesses in a car.

da Cunha *et al.*, (2006) have realized that the classical representations of quality data mining problems are not sufficient in this case. To solve this problem they suggest a novel encoding of the sequence problem. Each column in the table represents a different sequence pair. By that "the precedence information is contained in the data itself". For instance Figure 1 illustrates a product route which consists of 5 operations followed by an inspection test targeted to identify faults. Because a fault was identified, rework of operation 9 has to be done. The rework is followed by an additional inspection test that indicates that the product fits the product specification.

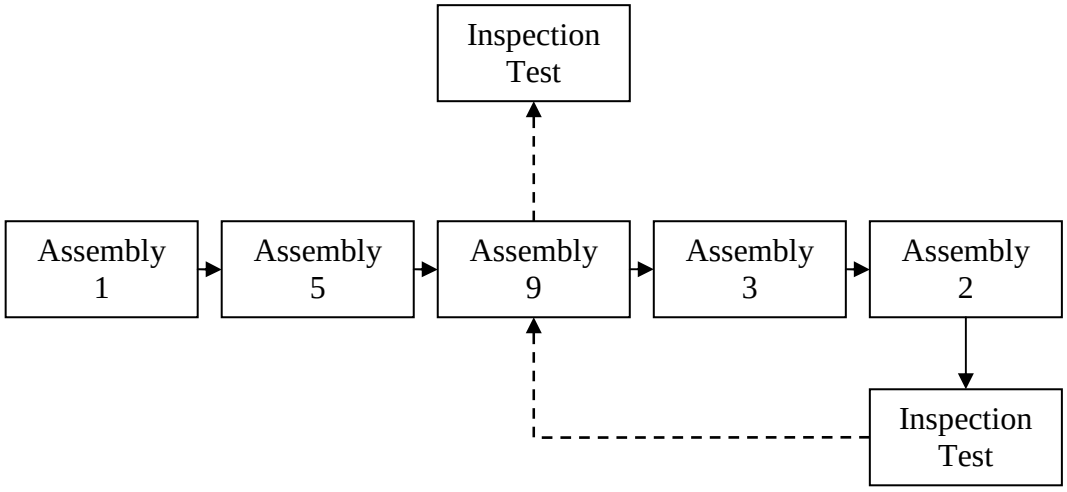

Figure 1: Illustration of a product route.

da Cunha *et al.*, (2006) suggest to represent the sequence 1-5-9-3-2 by the following set of pairs: {B_1, 1_5 , 5_9, 9_3, 3_2, 2_F}. Note that B_1 and 2_F represent the fact that the sequence began in operation 1 and that the final operation is 2. By proposing this encoding, it is now possible to analyze the effect of the production sequence on the quality of the product. For this purpose they have used association rules algorithm and have obtained very encouraging results.

Nevertheless the above representation is capable to handle only immediate precedents. da Cunha *et al.*, (2006) suggest to solve the problem by focusing on the most critical operations. However automatically identifying critical operations is not always simple.

Additionally while this representation is capable to handle sequences whose length is greater than two operations, it is not efficient for long sequences, because it requires much higher dimensionality. For instance in order to examine sequence of three operations, one is required to add columns for all triplets (B_1_5, 1_5_9, etc). Increasing the dimensionally may cause some problems to data mining algorithms – a phenomena known as "Curse of Dimensionality". Finally the above representation assume that a normal operation can be performed at most once per product (i.e sequence such as 1-3-2-1-4-3-1 is not allowed).

The goal of this work is to suggest a data mining process that is capable to mine the effect of the production sequence on the product quality. It extends the pioneer research presented by da Cunha et al. (2006) by searching for sequence patterns of any length and that may contains operations that are not necessarily immediate precedents. Moreover it can examine sequences in which the same operation is performed more than once for the same product. The proposed method can also consider the makespan factor (i.e. the time span of each operation, and the idle times).

The main contribution of this paper is developing a new data mining process that employs concepts that have been developed in various fields such as bioinformatics and artificial intelligence and use them in manufacturing applications.

2. Algorithmic Framework

2.1 Overview

In order to solve the problem presented above, we suggest using the framework presented in Figure 2. The process includes four phases:

1. **Sequence Representation:** A domain specific task designed to represents production sequences as string of tokens.
2. **Patterns discovery**: The automatic creation of a regular expression pattern from each pair of sequences via the Longest Common Subsequence (LCS) algorithm or the Teiresias algorithm.
3. **Patterns selection**: Applying heuristics and features selection techniques to select the best patterns for correct classification of the concept.
4. **Classifier training**: Training a C4.5 decision tree classifier to combine several patterns.

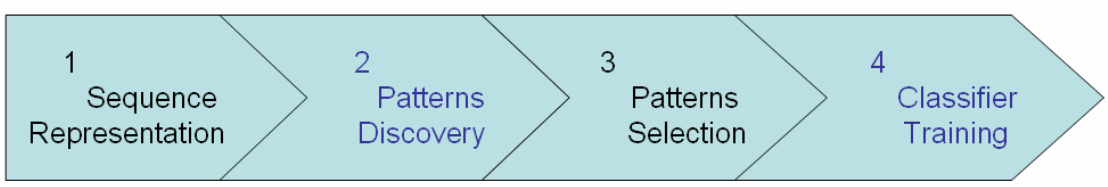

Figure 2: The process overview

The following subsections describe each of the above phases.

2.2 Representing the Production Sequence Using a String of Tokens

Each product manufacturing data is represented as a string of tokens in which each token represents a different operation activity. If the makespan factor is not important then the representation is straightforward. For instance the production sequence 1-5-9-3-2 is represented as the string "B 1 5 9 3 2 F".

If the makespan factor is important then a more complicated representation is required. For this purpose we first need to decide what the desirable time granularity is. Time granularity should be no more than the minimum operation duration in the database.

After deciding what the desirable time granularity is, we can represent the manufacturing process of each product instance as a string. Each time bucket is represented as a single letter. Following Rigoutsos and Floratos (1998), we denote by Σ the alphabet of the manufacturing. Each letter in Σ represents the identification of the operation performed in this time bucket. Idle time is also represented by a special letter (for instance "_"). The size of Σ depends on the number of operations needed to be encoded. For instance the string "B 1 1 _ _ _ _ 3 3 3 F" represents a manufacturing sequence with the operation "1" being performed during the first two time buckets. Then during the four subsequent time buckets no operation has performed followed by three time buckets in which operation "3" is performed.

2.3 Patterns Discovery

The term sequence pattern usually refers to a set of short sequences that is precisely specified by some formalism. Following many research in bioinformatics (Hatzigorgiou et al., 2001) we are also adopting regular expression in order to represent sequence patterns. A pattern is defined as any string consisting of letter of the alphabet Σ and the wild-card character '.'. The wild-card (also known as the don't care character) is used to denote a position that can be occupied by any letter of the alphabet Σ. For instance the pattern "B 1 1 _ . . _ 3 . 3" can be matched against the following production strings "B 1 1 _ _ _ _ 3 3 3", "B 1 1 _ 2 4 _ 3 1 3", "1 1 _ 7 _ _ 3 4 4", etc.

The pattern element *".*"* denotes an arbitrary string of symbols (possibly of length 0), *".{3,5}"* denote any string of between 3 and 5 characters.

In this paper, two alternatives for obtaining the regular expressions have been considered:

1. Longest common subsequence (Myers, 1986) that provides an efficient method of finding the longest common subsequence between two sequences.
2. TEIRESIAS (Rigoutsos and Floratos, 1998) - originally developed as a combinatorial pattern discovery algorithm in bioinformatics for analyzing DNA sequences. The algorithm identifies recurrent maximal patterns within sequences.

The following subsections describe each one of these methods.

2.3.1 Using the longest common subsequence algorithm

One approach for discovering the sequence patterns in the data is to compare any pair of production sequences with the same quality label (i.e. "Passed"/"Not Passed"). From each pair, we create the longest regular expression that fits the two sequences. For instance assuming we are given the following two sequences (note that the spacing between the operations have no meaning and are used for clarity)

```
B 1 8 4 2 3 4 F
B 9 1 4 2 7 F
```

Table 1 illustrates how the longest regular expression can be extracted from the two strings. Every line in the table refers to a different part in the production sequences. The first column enumerates the subsequence part. The following two columns present the subsequences. Note that by concatenating the subsequences, one can obtain the original complete sequence. The last column presents the generalized regular expression pattern that covers these subsequences. For instance in the first line, both subsequences contain the character "B" (Begin), thus the generalized pattern is also "B". In the second line, the first subsequence is empty (null) and the second subsequence is "9", thus the generalized subsequence is the regular expression ".{0,1}" meaning that "one or no operation" generalized these subsequences. Note that whenever there was empty subsequence we added a wild card expression of minimum length of 0 and maximum length of the compared subsequence. On the other hand whenever there were two unequal subsequences, we added a wild card expression with the minimum length of the shortest subsequence and maximum length of the largest sequence.

Table 1: Longest Common Subsequence Generation

#	Sequence 1	Sequence 2	Pattern
1	B	B	B
2		9	.{0,1}
3	1	1	1
4	8		.{0,1}
5	4 2	4 2	4 2
6	34	7	.{1,2}
7	F	F	F

By concatenating the expressions that appear in the pattern column, we can obtain the following regular expression pattern:

```
B .{0,1} 1 .{0,1} 4 2 . {1,2} F
```

In order to find the subsequences in Table 1, we use the longest common subsequence algorithm (Myers, 1986). One way to solve the problem is to covert it into the longest path problem based on the lattice graph presented in Figure 3. The nodes in the upper horizontal line are labeled with the operations of the first sequence and the nodes of the first vertical line are labeled with the operation of the second sequence. In this graph all horizontal and vertical edges are possible. Additionally diagonal edges in which the target node has the same horizontal and vertical label, are also available. If the horizontal and vertical edges have the length zero, and the diagonal edges have length one, then the longest common subsequence corresponds to the longest path from the top left corner to the bottom right corner. This graph is acyclic and is frequently solved using dynamic programming. The highlighted path presents one of the longest paths. Note that the destination node of the highlighted diagonal edges (B,1,4,2,F) are used in the regular expression, while the horizontal/vertical edges are converted to the wild-card expression.

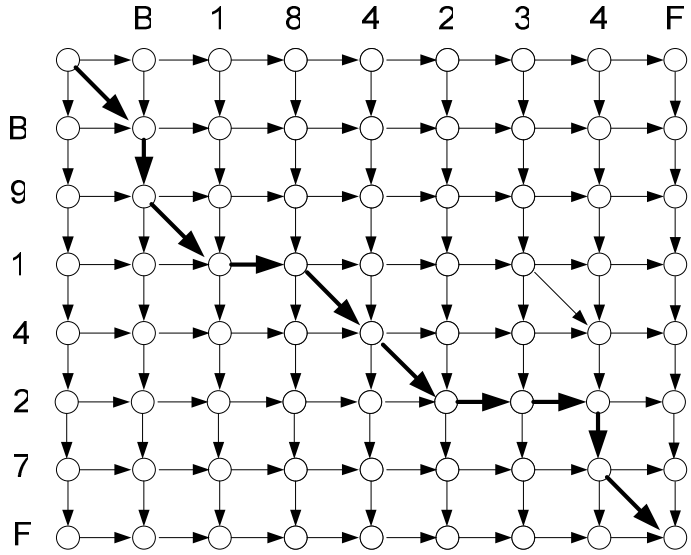

Figure 3: The longest path for solving the longest common subsequence

2.3.2 Using the Teiresias algorithm

The Teiresias algorithm was designed for the discovery of motifs in biological sequences, an important research problem (Rigoutsos I, Floratos A., 1998). The method is combinatorial in nature and able to produce all patterns that appear in at least a (user-defined) minimum number of sequences, yet it manages to be very efficient by avoiding the enumeration of the entire pattern space. The algorithm performs a well-organized exhaustive search. In the worst case the algorithm is exponential, but works very well for usual inputs. Furthermore, the reported patterns

are maximal: any reported pattern cannot be made more specific and still keep on appearing at the exact same positions within the input sequences. Teiresias searches for patterns which satisfy certain density constraint, limiting the number of wild-cards occurring in any stretch of pattern. More specifically Teiresias looks for maximal <L,W> patterns with support of at least K (i.e. in the corpus there are at least K distinct sequences of that match this pattern). A pattern P is called <L,W> pattern if every sub pattern of P with length of at least W operations (combination of specific operations and "." wild-card operations) contains at least L specific operations.

For example, given the following corpus of 6 negative production sequences (Note that the begin and finish indicators are removed):

```
1 3 2 5 4
1 7 8 2
1 9 10 11 2 6 4
1 12 2
13 14 7 1 12 2
16 17 1 3 18 8 2
```

The Teiresias program (L=K=2 and W=5) discovers 6 recurring patterns shown in the following table:

Table 2: Illustrative results of the Teiresias algorithm

#	Sequence 1
2	1 12 2
2	1 3
2	8 2
3	1 . 2
2	2 . 4
2	1 . . . 2

The first column represents the support of the pattern. The dot represents the wild-card. Next we transform the Teiresias patterns to regular expression patterns by replacing each dot with a regular expression such as *.{0,L}*, where L is the number of dots.

2.4 Patterns Reduction

Obviously there are many patterns that can be created via the LCS (each pair of sequences with "Not Passed" class). In fact, initially too many patterns are created and it is essential to keep with a manageable number of patterns. For example, a training set of 140 "Not Passed" sequences yielded 140*139/2=9730 patterns. Assuming the rework operation correlates with the faulty sequence pattern, we can reduce the number of learned patterns via splitting the training as per the rework operation, then learning the LCS/Teiresias patterns from each part, and finally merging the patterns. Another simple reduction technique is by ranking the patterns according to their descending support in the faulty corpus, keeping only the top patterns. In addition we suggest using a two phase pattern reduction as described in the following subsections.

In the first phase we introduce new patterns that are created by merging existing patterns. In the second phase we reduce these set of patterns using Correlation based feature selection.

Simple heuristic for patterns generalization

The proposed simple heuristic for patterns reduction is based on the idea of merging two or more patterns into a single pattern. The merging is performed based on the specific operations used in each pattern while ignoring the wild-cards. For instance the "specific operation" representation of "B 1 2 .{2,7} 3 F" is "B 1 2 3 F".

All patterns with the same "specific operation" representation are merged by the smoothing wild-card expressions. This is obtained by the taking the minimum and maximum For example, the patterns “B 1 2 .{2,7} 3 F” and “B 1 2 .{3,9} 3 F” and "B 1 .{1,3} 2 .{3,4} 3 F" which all have the same "specific operation" representation of "B 1 2 3 F" are generalized using the pattern “B 1 .{0,3} 2 .{2,9} 3 F”.

Any two or more patterns that their "specific operation" representations are different in one position (Hamming distance of one) are generalized by introducing a wild-card to that position.

Correlation based feature selection

Feature selection is the process of identifying relevant features in the dataset and discarding everything else as irrelevant and redundant. For this purpose each "regular expression" pattern represents a different feature. In this work we are use a non-ranker filter feature selection algorithm. Filtering means that the selection is performed independently of any learning algorithm. Non-ranker means that the algorithm does not score each pattern but provides only which pattern is relevant and which is not. The following Table 3 describes the training set matrix before features selection. The rows are training sequences (“Passed” and ”Not Passed”), the first K columns are the regular expression patterns; the last column is the target class (“Passed” / ”Not Passed”),. The cell value is 1 if the regular expression matches the sequence, 0 otherwise. The matrix described above is the input to the features selection algorithms (Chizi and Maimon, 2005).

Table 3: Training set matrix before features selection

Sequence #	Pattern 1	Pattern 2	Pattern 3	...	Pattern K	Quality
sequence 1	1	1	0	...	1	Passed
sequence 2	1	0	1	...	0	Passed
⋮	⋮	⋮	⋮	⋮	⋮	Passed
sequence N	1	0	1		1	Passed
sequence N+1	1	1	1		0	Not Passed
sequence N+2	0	1	1		0	Not Passed
⋮	⋮	⋮	⋮	⋮	⋮	Not Passed
sequence M	0	1	1		0	Not Passed

In this work we use the Correlation-based Feature Subset Selection (CFS) as a subset evaluator (Hall 1999). CFS Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low inter-correlation are preferred. This approach is suitable to this case, because in the first phase (the simple heuristic for patterns generalization), we create merged patterns that are correlated with the basic patterns. In this phase we select which of the patterns should remain.

The CFS algorithm was executed with Best First Forward Selection. This search strategy searches the space of attribute subsets by greedy hill-climbing augmented with a backtracking facility. It starts with the empty set of attributes and search forward.

3.6 Creating a Decision Tree Classifier

The filtered matrix, together with the manual classification of each concept, is fed into a C4.5 classification algorithm which creates a classification decision tree (Quinlan , 1993).

Using a decision tree as a classifier in this case has several advantages: (1). the production sequence is not classified based on a single pattern, but on set of patterns, i.e. this classifier can be used to indicate that a production sequence is classified to the label "Not passed" only if it matched two patterns and does not match to a third pattern. This is more expressive than the classical approach in which the classification is based on a single pattern. Moreover in this way instead of searching of complicated regular expressions, we can search for simple regular expressions and "rely" on the decision tree to combine them. In fact in some cases it is possible to express a tree

path as a single complicated regular expression; (2). The hierarchical structure of decision tree enforces an order (priority) in the usage of patterns, i.e. given a new production sequence, not all patterns should be matched in advance but one pattern at a time based on the specific branch traversing.; (3). As oppose to other classifiers (such as neural networks) the meaning of the classifier can be explained to the production engineer.

Experimental Study

1. Overview

The purpose of this study is to show that the proposed method matches the performance of the previous work in similar conditions and outperforms when the complexity of the problem domain increases, as in more realistic conditions. Therefore we compare the classification performance of the proposed method to an association rules classifier. In addition we compare to a Bi-gram based decision tree classifier, a near equivalent representation to association rules. We first explain the Bi-gram representation method. Then we overview the evaluation measures used in the experimental study.

2. Algorithms Examined

The Bi-gram representation is a common and proven method used for text classification problems. Texts cannot be directly interpreted by a classifier or by a classifier-building algorithm. Because of this, a transformation procedure that maps a text d_j into a compact representation of its content needs to be uniformly applied to the text corpuses (Sebastiani, 2002). In Text Categorization (TC) a text d_j is usually represented as a vector of term weights $d_j=(w_{1j},\ldots,w_{|\tau|j})$ where τ is the set of terms (sometimes called features) that occur at least once in at least one document of T_r, and $0 \leq w_{kj} \leq 1$ represents, loosely speaking, how much term t_k contributes to the semantics of document d_j . Differences among approaches are accounted for by (1) different ways to understand what a term is; (2) different ways to compute term weights. A typical choice for (1) is to identify terms with words. This is often called either the "*set of words*" or the "*bag of words*" approach to document representation, depending on whether weights are binary or not. Another popular choice for (1) is to identify terms with words sequences of length *n*. This n-gram vector text representation method is used to classify text documents (Damashek, 1995). Damashek selected the normalized frequency with which the n-gram occurs in the document as the choice of (2) term weight. Each vector identifies a point in a multidimensional space, and similar documents are expected to have points close to each other. Damashek (Damashek, 1995) used the dot product between two histogram vectors as a measure of their similarity, but he pointed out that other measures are possible. In this study we regard operations as words, binary weight and sequence length of 2.

3. Evaluation Measures

The evaluation measures that we use are: Precision, Recall, F-Measure, Accuracy and Complexity

The notion of "precision" (the proportion of correctly classified sequences to all the classified sequences) and "recall" (The proportion of correctly classified sequences, out of all sequences belonging to that class) are widely used in information retrieval (Van Rijsbergen, 1979) and data mining. Equation (1) the formal definition of precision and (2) of recall.

$$(1)\ \Pr ecision = \frac{Number\_of\_correctly\_classified\_sequences}{Number\_of\_classified\_sequences}$$

$$(2)\ \mathrm{Re}call = \frac{Number\_of\_correctly\_classified\_sequences}{Total\_number\_of\_sequences\_of\_classified\_class}$$

The weighted harmonic mean of precision and recall, the traditional F-measure is defined in the following Equation (3):

$$(3)\ F = \frac{2 \cdot P \cdot R}{P + R} = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

Another evaluation measure we use is the Accuracy. Accuracy is the number of correct classification / the number of classified sequences.

We use the common complexity measure for a decision tree classifier, the number of internal nodes. For a binary tree (matched/not matched) the number of leaves is always the number of internal nodes + 1.

The experiments involving decision trees are conducted using the Weka data mining framework. Performance testing is computed using the 10-fold-cross-validation setup.

4. Results

Two test sets of 250,000 products routes each were randomly generated with the following constraints:

Test set 1 (same experimental setup presented by Da Cunha *et al.* 2006):

The aim of this test set is to examine if the suggest method is capable to solve the original problem suggested by Da Cunha *et al.* 2006 and that it is does not harm.

- An operation can be performed as a normal task (i.e. no rework task) at most once per product.
- Random faults are generated at 5% of the products routes.
- Systematic fault sequences are introduced:
    - If operation 5 isn't the last task, this operation has to be redone.
    - If operation 2 precedes operation 4, operation 4 has to be redone.

The notation we use is as follows. Task b is the start operation. Task f is the last operation in case no work is redone. Task Rew followed by an additional (1-6) marks that the additional task was redone. For example, the following products routes:

1. b 2 5 4 f Rew4 => 2 precedes 4 so 4 has to be redone
2. b 3 6 5 2 f Rew5 => 5 isn't the last task so it has to be redone
3. b 2 6 3 f Rew2 => 5% random faults, 2 is randomly chosen for rework out of {2,6,3}
4. b 2 6 f => No work redone.
5. b 3 6 2 f => No work redone.

Following the same transformation presented by Da Cunha *et al.* 2006, an *arff* file was created, using only the faulty sequences and converted to *Tanagra* format using the *Datanamorf* conversion program (0 replaced with F; 1 replaced with T to avoid *Tanagra* issue). Association rules were then extracted from the data set, using supervised Assoc Rule algorithm with parameters: *support=0.10; confidence =0.75; and Max rule length = 2.*

The association rule extracted for rework operation 4 is:

Table 4: Association rules extracted for Test-Set 1 rework operation 4

N°	Antecedent	Length	Support	Confidence	Lift
1	a24=T	1	0.102 (0.00)	1.000 (0.00)	9.381 (0.00)

The following association rules were extracted for rework operation 5 after lowering the support to 0.07:

Table 5: Association rules extracted for Test-Set 1 rework operation 5

N°	Antecedent	Length	Support	Confidence	Lift
1	ab5=T	1	0.150 (0.00)	0.898 (0.00)	1.974 (0.00)
2	a56=T	1	0.090 (0.00)	0.883 (0.00)	1.941 (0.00)
3	a52=T	1	0.086 (0.00)	0.837 (0.00)	1.841 (0.00)
4	a54=T	1	0.099 (0.00)	1.000 (0.00)	2.199 (0.00)
5	a51=T	1	0.089 (0.00)	0.901 (0.00)	1.980 (0.00)
6	a53=T	1	0.088 (0.00)	0.901 (0.00)	1.982 (0.00)

An ideal decision tree classifier is presented in the following Figure 4.

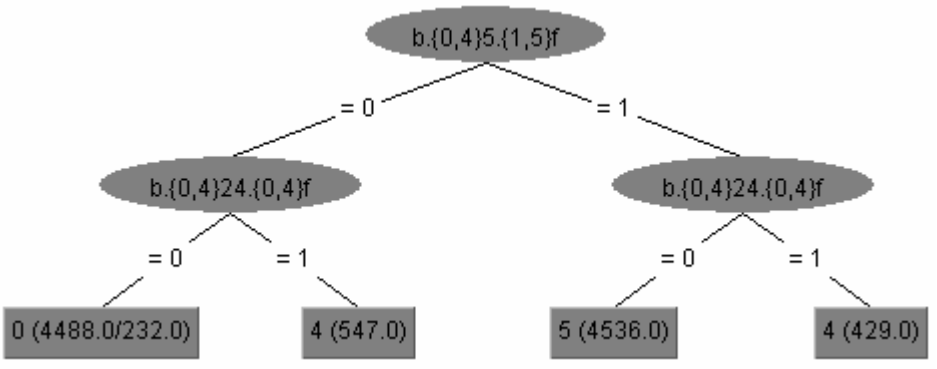

Figure 4: Ideal Decision tree classifier for Test set 1

A Bi-gram decision tree classifier is presented in the following Figure 5:

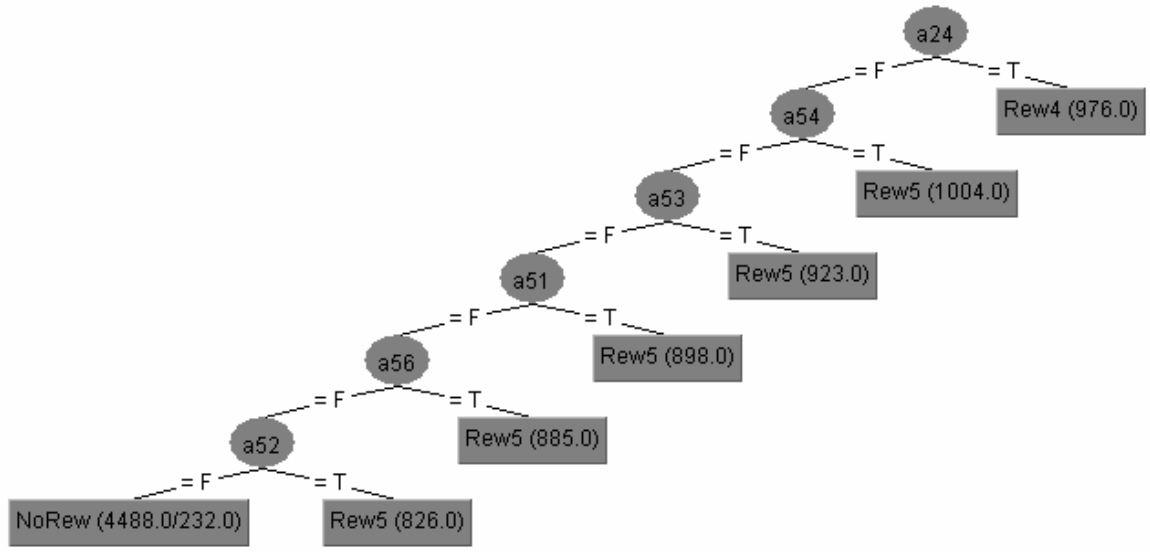

Figure 5: Bi-gram Decision tree classifier extracted from Test set 1

Next we apply the proposed method for automatic discovery of faulty sequences from Test set 1. A subset of 10,000 sequences from Test set 1 was randomly selected for the experimental study. LCS based patterns were generated using the sequences that required redone.

Since we assume that the faulty sequences are correlated with the required redone operation, we can split the training of a LCS based classifier into two phases. In the first phase we train a classifier per redone task (e.g. redone task 5), to distinguish sequences requiring redone of the specific task from sequences not requiring the redone of that task. In the second phase we combine the patterns obtained is phase 1, from the various rework operations, to train a combined classifier. The combined classifier is trained to classify sequences into the redone task (task 4, or 5, or other redone, or no redone).

For redone task 5 we first learn LCS patterns from 100 faulty sequences requiring redone 5. The resulting 4,950 patterns are then ranked according to their descending support in the faulty sequences corpus, first 100 are kept. Next we apply the generalization procedure described above. For example, the generated patterns: *b.{0,3}5.{0,2}4.{0,4}f; b.{0,3}5.{0,3}1.{0,3}f; b.{0,3}5.{0,3}2.{0,3}f; b.{0,3}5.{0,3}6.{0,4}f; b.{0,4}5.{0,3}3.{0,3}f;* and *b.{0,4}5.{1,4}f* are added following generalization of patterns having hamming distance 0. The generated patterns *b.{0,3}5.{1,4}f; b.{0,3}5.{1,5}f; and b.{0,4}5.{1,4}f* are added following generalization of patterns having hamming distance of 1. Next we remove duplicated patterns and obtain 109 patterns useful for detecting faulty rework 5 sequences.

Now we apply the same procedure to rework 4 sequences, yielding additional 175 patterns. Finally we combining both redone 4 and redone 5 pattern files (total 221 patterns, after removing duplicate patterns), apply a CFS filter and train a J48 decision

tree. The resulting J48 decision tree includes 4 leaves, achieves 97.6% accuracy (using 10 folds cross validation) in classifying the required rework sequences. This result is identical to the Bi-gram classifier performance and to the expected results from an ideal classifier.

The following Figure 6 demonstrates the resulting decision tree.

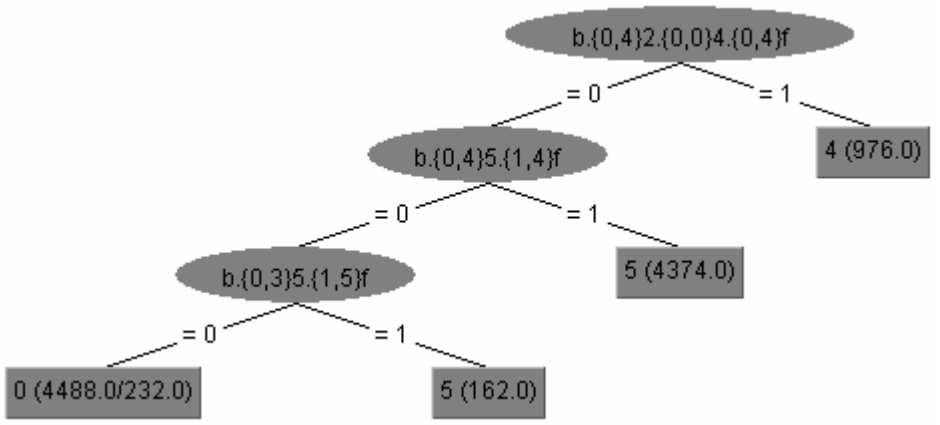

Figure 6: LCS based decision tree classifier extracted from Test set 1

The following Table 6 summarizes the results of Test set 1. It indicates that in the first test set all methods have obtained similar results from the predictive power.

Table 6: Summary of Test set 1 result

	Precision			Recall			F measure			Accu.
	Rew4	**Rew5**	**NoRew**	**Rew4**	**Rew5**	**NoRew**	**Rew4**	**Rew5**	**NoRew**	
Ideal classifier	100%	100%	94.6%	95.6%	99.3%	100%	97.7%	99.6%	97.3%	97.6%
Association rules	100%	100%	94.7%	95.6%	99.3%	100%	97.8%	99.7%	97.3%	97.6%
Bi-gram	100%	100%	94.6%	95.6%	99.3%	100%	97.7%	99.6%	97.3%	97.6%
LCS	100%	100%	94.6%	95.6%	99.3%	100%	97.7%	99.6%	97.3%	97.6%

(* Number of association rules)

	Complexity
Ideal classifier	4
Association rules	7*
Bi-gram	7
LCS	4

The next test set is designed to demonstrate the capability of the proposed method to handle some of the limitations described by Da Cunha *et al.* 2006. The primary limitations are: (1) Up to one normal task per product; and (2) Representing only preceding tasks.

Test set 2 (including some of the challenges described by Da Cunha *et al.* 2006):

The aim of this test set is to examine if the proposed method is capable to handle non-immediate precedents.

- An operation can be performed as a normal task (i.e. no rework task) **any number of times**.
- Random faults are generated at 5% of the products routes.
- Systematic fault sequences are introduced:
    - In sequences including operation 5, if operation 2 is performed, **followed by any other operation**, then operation 4, operation 4 has to be redone.

We follow the same notation as in Test set 1 above. For example, the following 5 products routes:

1. b 2 6 5 7 4 f Rew4 => 5, 2 followed by some operations, then 4 so 4 has to be redone
2. b 2 6 3 f Rew2 => 5% random faults, 2 is randomly chosen for rework out of {2,6,3}
3. b 2 3 2 3 f => Operation 2 performed as normal task more than once. No work redone.
4. b 2 6 f => No work redone.
5. b 3 5 6 2 f => No work redone.

An ideal classifier for test set 2 is presented in the following Figure 7:

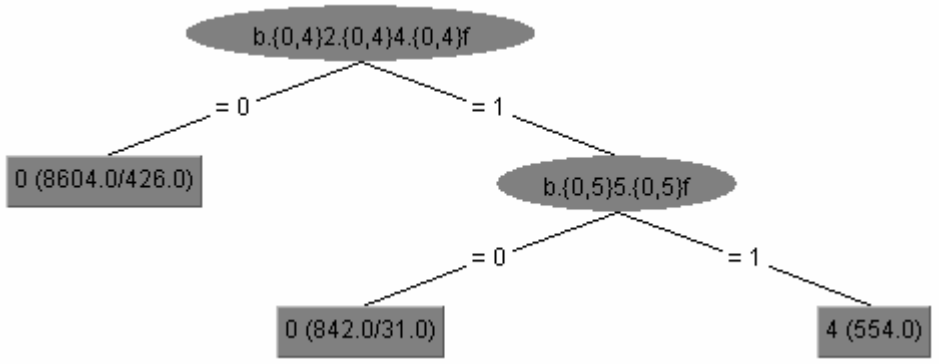

Figure 7: Ideal Decision tree classifier for Test set 2

The association rules for test set 2, rework operation 4 are:

Table 7: Association rules extracted for Test-Set 2 rework operation 4

No	Antecedent	Length	Support	Confidence	Lift
1	ab2=T - a54=T	2	0.008 (0.00)	1.000 (0.00)	15.773 (0.00)
2	ab2=T - a45=T	2	0.009 (0.00)	1.000 (0.00)	15.773 (0.00)
3	af5=T - a24=T	2	0.010 (0.00)	1.000 (0.00)	15.773 (0.00)
4	af4=T - a25=T	2	0.009 (0.00)	1.000 (0.00)	15.773 (0.00)
5	af4=T - a52=T	2	0.008 (0.00)	1.000 (0.00)	15.773 (0.00)
6	ab5=T - a24=T	2	0.009 (0.00)	1.000 (0.00)	15.773 (0.00)
7	a25=T - a24=T	2	0.002 (0.00)	1.000 (0.00)	15.773 (0.00)
8	a25=T - a54=T	2	0.010 (0.00)	0.897 (0.00)	14.141 (0.00)
9	a25=T - a34=T	2	0.002 (0.00)	0.750 (0.00)	11.830 (0.00)
10	a65=T - a24=T	2	0.002 (0.00)	1.000 (0.00)	15.773 (0.00)
11	a51=T - a24=T	2	0.002 (0.00)	1.000 (0.00)	15.773 (0.00)
12	a24=T - a53=T	2	0.003 (0.00)	1.000 (0.00)	15.773 (0.00)
13	a24=T - a35=T	2	0.003 (0.00)	1.000 (0.00)	15.773 (0.00)
14	a24=T - a52=T	2	0.010 (0.00)	1.000 (0.00)	15.773 (0.00)
15	a24=T - a56=T	2	0.003 (0.00)	1.000 (0.00)	15.773 (0.00)
16	a24=T - a54=T	2	0.003 (0.00)	1.000 (0.00)	15.773 (0.00)
17	a24=T - a45=T	2	0.011 (0.00)	1.000 (0.00)	15.773 (0.00)
18	a24=T - a15=T	2	0.003 (0.00)	1.000 (0.00)	15.773 (0.00)
19	a24=T - a55=T	2	0.003 (0.00)	1.000 (0.00)	15.773 (0.00)
20	a12=T - a54=T	2	0.002 (0.00)	0.767 (0.00)	12.093 (0.00)

Next we apply the proposed method as described for Test set 1 above. The following Figure 8 demonstrates the resulting decision tree. It is clearly shown that the important sequence patterns are included in the decision tree.

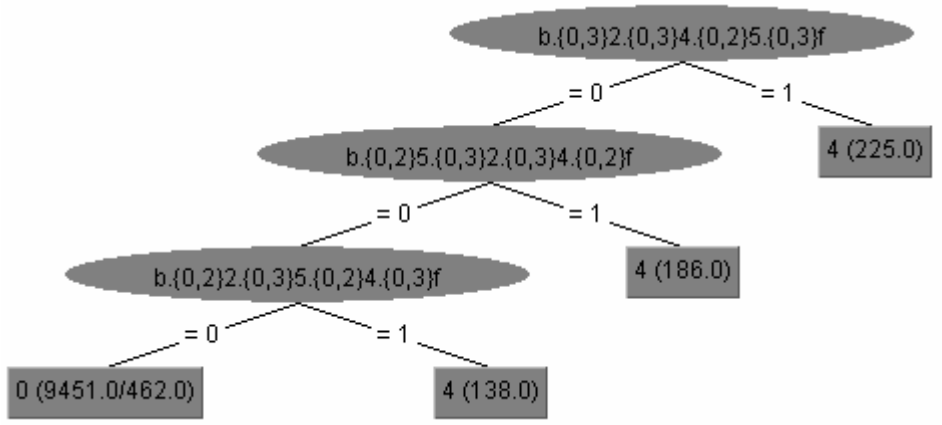

Figure 8: LCS based decision tree classifier extracted from Test set 2

The following Table 8 summarizes the results of Test set 2. It is shown that the LCS classifier performs better than the Association rules and Bi-gram classifiers with much smaller complexity. This observation is especially important in realistic conditions, when the domain complexity increases, as classifier complexity becomes a limiting issue. In addition it is shown that the new method performs very near to the ideal classifier.

Table 8: Summary of Test set 2 results

	Precision		Recall		F-measure		Accuracy	Complexity
	Rew4	NoRew	Rew4	NoRew	Rew4	NoRew		
Ideal classifier	100%	94.4%	85.4%	100%	93.3%	97.5%	95.43%	3
Association rules	95.2%	94.5%	77.4%	99.7%	85.4%	97.1%	94.6%	20**
Bi-gram *	97.6%	94.6%	77.9%	99.9%	86.7%	97.1%	94.7%	42
LCS *	100%	95.1%	86.6%	100%	92.8%	97.5%	95.38%	4

(* without feature selection
** Number of association rules)

Test set 3 is designed to demonstrate how the proposed method can detect faulty sequences with length greater than two. Test set 3 is like test set 2 with the following addition: In sequences including the subsequence "2 3 4" (as-is) then operation 4 is rework, no other faulty sequences may result rework 4. In addition, in order to enable enough faulty sequences, the sequence length is extended up to eight normal operations.

An ideal classifier for test set 3 is presented in the following Figure 9:

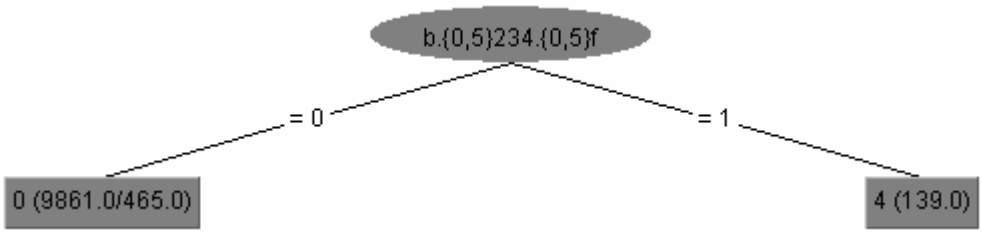

Figure 9: Ideal Decision tree classifier for Test set 3

The following association rule was extracted for rework operation 4 after lowering the support to 0.01 and confidence to 0.7:

Table 9: Association rules extracted for Test-Set 3 rework operation 4

N°	Antecedent	Length	Support	Confidence	Lift
1	a34=T - a23=T	2	0.014 (0.00)	0.720 (0.00)	31.178 (0.00)

The Bi-Gram tree for Test set 3 is presented in the following Figure 10:

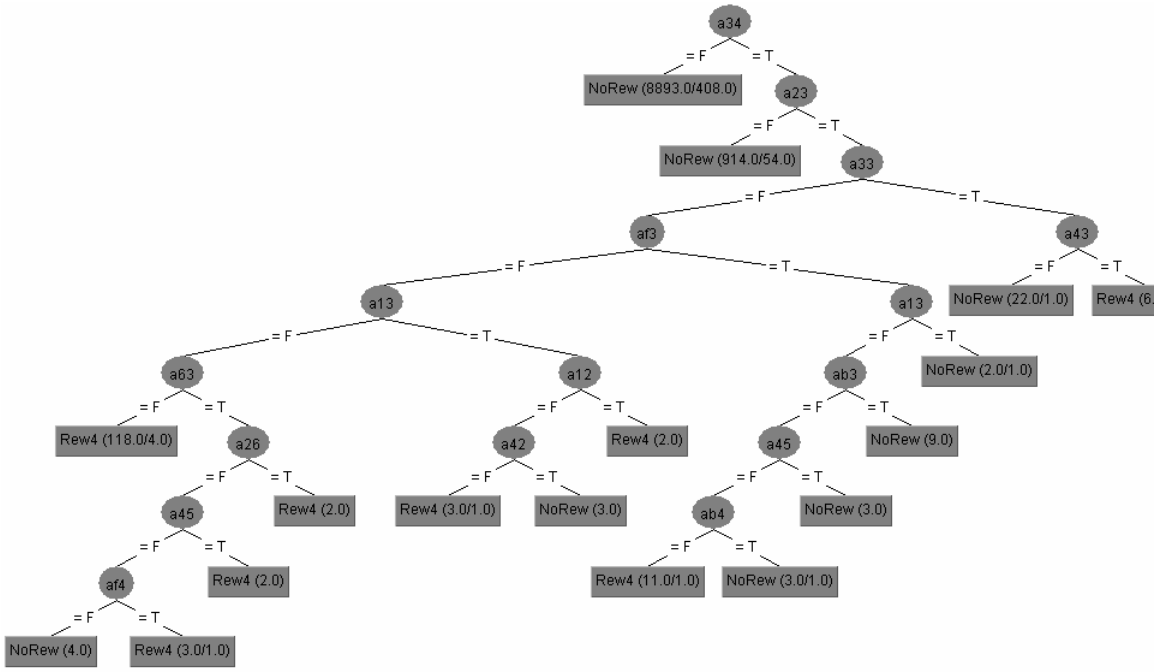

Figure 10: Bi-Gram Decision tree classifier for Test set 3

Next we apply the proposed method as described for Test sets 1 and 2 above. The following Figure 11 demonstrates the resulting decision tree. The new method automatically learned an equivalent of the “ideal pattern“.

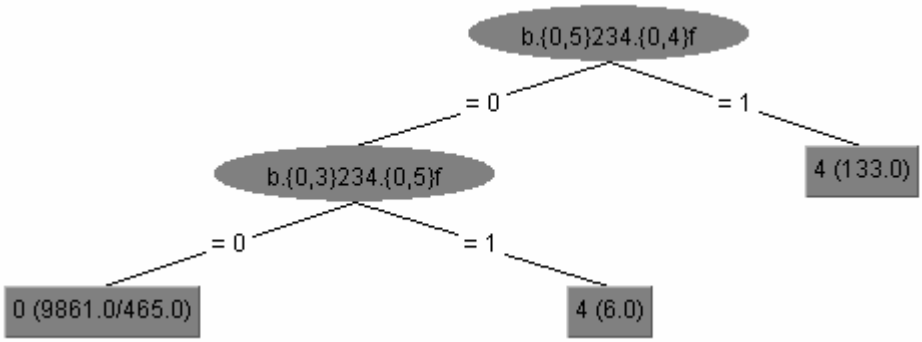

Figure 11: LCS based decision tree classifier extracted from Test set 3

The following Table 10 summarizes the results of Test set 3. It is shown that the LCS classifier performs much better than the Association rules and Bi-gram classifiers with much smaller complexity.

Table 10: Summary of Test set 3 results

Classifier	Precision		Recall		F-measure		Accuracy	Complexity
	Rew4	NoRew	Rew4	NoRew	Rew4	NoRew		
Ideal classifier	100%	95.3%	60.2%	100%	75.1%	97.6%	95.35%	3
Association rules	72.0%	95.3%	60.2%	99.5%	65.6%	97.3%	94.8%	1*
Bi-gram	89.1%	95.1%	52.8%	99.9%	66.3%	97.4%	95.04%	17
LCS	100%	95.3%	60.2%	100%	75.1%	97.6%	95.35%	3

(* Number of association rules)

Conclusion and Future Research

It is shown that the proposed method matches the performance of the previous work in similar conditions and outperforms when the complexity of the problem domain increases, as in more realistic conditions. The proposed method is more powerful due to two main features. Regular expressions are more generic representation than bi-grams, and decision trees are more expressive compared to support vectors. The bi-gram representation is merely a specific type of regular expression. The regular expressions are not limiting in any way the re-occurrences of tasks within a product sequence. As shown, the LCS algorithm also captures more complex sequences which are very likely in realistic manufacturing environment.

The proposed method is evaluated using a synthetic test set. In the next phase we plan to apply the method to actual data from a manufacturing environment, e.g. a semiconductor fabrication plant. Additional work can yield improved patterns so that the resulting decision trees will be clear and readable to engineering staff in a manufacturing environment.

One important extension to this work will be combining the operation sequence with the operation setting in the same decision tree classifier. As oppose to classic classification problem, not all input attributes in this problem are given in advance.

This is true because the value of an input attribute that is associated with an operation setting is available only if the operation was actually used in this product instance. One way to solve this problem is to create a sparse training set in which most attributes instances are set to a dummy value (for instance zero) that represents that this attribute is not relevant to this sequence. Sparse training set can prolong the training time. Thus it is desirable to modify the existing decision tree induction algorithm by adding a new step in which the training set is extended based on the sequence matching. The operation setting attributes become available only after a certain operation is matched as part of the operation sequence. Thus the decision tree always begins with a sequence pattern that might be follows with either an operation setting parameters or another operation sequence. Thus whenever a new sequence pattern node is added to the tree, we vertically extend the training set with the relevant operation setting attributes.

Figure 12 illustrates the idea by presenting part of the decision tree. For the sake of comprehensibility we differentiate between three type of nodes: (1) full ellipse which represents a sequence pattern (2) dashed ellipse which represents operation setting. (3) rectangle which represents the terminal node with the predicted outcome.

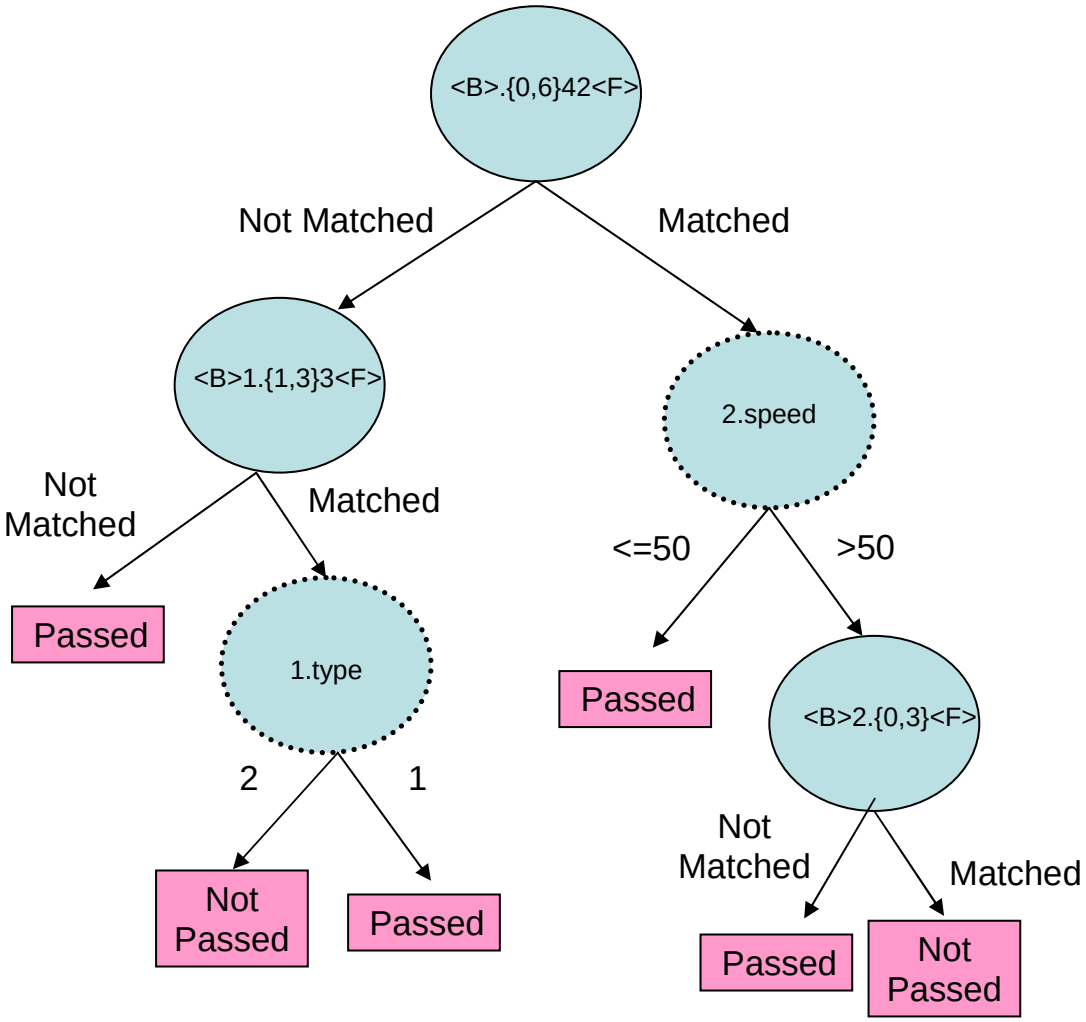

Figure 12: Illustration of a decision tree that combines sequential patterns and operations settings

References

Braha, D., Shmilovici, A., “On the use of Decision Tree Induction for Discovery of Interactions in a Photolithographic Process,” IEEE Transactions on Semiconductor Manufacturing, vol. 16 (4), pp. 644-652, 2003.

Chizi B., Maimon O., Dimension Reduction and Feature Selection, The Data Mining and Knowledge Discovery Handbook, 93-111, 2005

da Cunha C. , Agard B., & A. Kusiak, Data mining for improvement of product quality, International Journal of Production Research, Volume 44, Numbers 18-19, -19/15 September-1 October 2006, pp. 4027-4041.

Hall, M. Correlation- based Feature Selection for Machine Learning, Phd Thesis, University of Waikato, 1999

Hand, D. (1998), “Data Mining – reaching beyond statistics”, Research in Official Stat. 1(2): pp. 5-17.

E. W. Myers An O(ND) Difference Algorithm and Its Variations, Algorithmica, Vol. 1, No. 1. , pp. 251-266, 1986

Quinlan, J. R. (1993), "C4.5: Programs for Machine Learning", Morgan Kaufmann

Rigoutsos, I. and Floratos, A. (1998). Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. Bioinformatics, 14(1):55-67.

L. Rokach and O. Maimon, "Data mining for improving the quality of manufacturing: a feature set decomposition approach", Journal of Intelligent Manufacturing, 17(3):285–299, 2006.