J Intell Manuf
DOI 10.1007/s10845-011-0517-5

Automatic discovery of the root causes for quality drift in high dimensionality manufacturing processes

Lior Rokach · Dan Hutter

Received: 31 March 2010 / Accepted: 7 February 2011

Abstract A new technique for finding the root cause for problems in a manufacturing process is presented. The new technique is designated to continuously and automatically detect quality drifts on various manufacturing processes and then induce the common root cause. The proposed technique consists of a fast, incremental algorithm that can process extremely high dimensional data and handle more than one root-cause at the same time. Application of such a methodology consists of an on-line machine learning system that investigates and monitors the behavior of manufacturing product routes.

Keywords Automatic root cause discovery · Data mining · Failure analysis · Concept drift · Quality control · Fault detection · Yield improvement

Introduction

The semiconductor industry continually seeks to improve yields in order to meet growing needs. At the same time, wafer fabrication process becomes more complex as technology advances. As with many other manufacturing processes, wafer fabrication often faces fluctuations in product quality; random events and subtle environmental changes in might increase failures.

A major problem in virtually any manufacturing factory is minimizing the defects that may appear in the manufacturing process. In the semiconductor industry, the fabrication process consists of thousands of variables and interactions that create a chaotic, non-linear environment which is little understood and whose outcomes are very hard to predict. Among the variables are the nature and quality of the tools, environmental factors, such as humidity, temperature, and tool setup parameters. These are but a few of the many elements where any slight deviation of one of them, can cause defects leading to poor yield and increased manufacturing costs. Therefore, defect inspection is vital in the fabrication process.

Root-cause identification for quality-related problems is a key issue in quality and productivity improvement in manufacturing. Unfortunately, root-cause identification is also a very challenging engineering problem, particularly for a multi-stage manufacturing process.

Statistical Process Control (SPC) (Ben-Gal 2006; Durham et al. 1995) and Design of Experiments (DOE), (Jemmy et al. 2005) are very common statistical methods to detect and analyze variations in semiconductor manufacturing process. Unfortunately, their extensive use does not ensure high yields at the end of the process; while they may be very much necessary, they insufficient for reducing defects. In addition, although all the inline parameters may be in control according to SPC, one can observe significant yield losses. The inability of these traditional methods to sufficiently reduce defects necessitates the search for more efficient methods.

Due to the decreased costs of computing in recent years, data mining represents a new frontier in the evolution of yield management systems because of its capability to discover correlations between various types of input data and to link process parameters in order to determine which parametric issues are impacting yield (Kenneth et al. 1999). Kittler and Wang (1999); Choudhary et al. (2009) and Chang et al. (2009) describe possible applications of data mining in semiconductor manufacturing.

Analyzing wafer fabrication data poses many challenges since the fabrication process, as noted above, is complex and

L. Rokach (✉) · D. Hutter
Department of Information Systems Engineering, Ben-Gurion University of the Negev, 84105 Beersheba, Israel
e-mail: liorrk@bgu.ac.il

Published online: 01 March 2011

error-prone. Many factors contribute to fluctuations in the yield obtained. The entire manufacturing process typically takes weeks or even months to complete. On the other hand, the hundreds of operations which the wafers undergo generate a huge amount of data that is collected and monitored from various sensors each day. Thus, new tools for analyzing complicated manufacturing environments are required (Chen et al. 2009; Haapala et al. 2008; Duan et al. 2009).

There have been many important studies in semiconductor data mining in recent years. Gardner and Bieker (2000) have studied the possibility of combining a self-organizing map (SOM) with neural networks and rule induction. Goodwin et al. (2004) reviewed Intel's recent developments of using data mining to address different problems in semiconductor manufacturing and compared them to several algorithms. The solutions have been integrated into a software package, called Interactive Data Exploration and Learning (IDEAL). Rokach and Maimon (2006) applied decision trees to Work-in-Process (WIP) data. Jemmy et al. (2005) presented a framework to unify the various data-mining solutions for wafer manufacturing data using two-phase clustering algorithm to efficiently group the defect wafers. Hyeon et al. (2006) proposed an improvement to the ingot fabrication step using bootstrap method and multiple regression models for data generation and DPNN (Dynamic Polynomial Neural Network) and decision tree for extracting rules. Rokach et al. (2008) presented an effective composited classifier composed of decision tree induction. They reviewed their new approach on semiconductor data. Bergeret and Le Gall (2003) tested different statistical approaches, such as: moving average, Markov chain and Bayesian method. Hu and Shun-Feng (2004) have written about hierarchal clustering methods to improve manufacturing process scheduling.

All these data mining solutions rely on data mining on demand. That is to say, the algorithms are usually activated manually when trends in the manufacturing process or drifts in the yield are identified. Unfortunately the proposed solutions cannot identify drifts automatically, a factor that reduces the effectiveness of the data mining solutions. In addition, most of these data mining solutions exploit data which has been extracted from machine measurements in order to conclude on emerging drifts. Unfortunately, this data is usually characterized by extremely high dimensionality and missing entries, which automatically reduces the efficiency of the model. Our methodology offers a new method for identifying the root-cause for quality drifts by constantly monitoring and inspecting new quality variations in the input data which consists of manufacturing routes.

A few researchers have incorporated concept drift as part of the data mining solution for manufacturing solution. Rodrigues and Gama (2004) presented a method to on-line check for concept drift in continuous Glass Manufacturing Process. Though a lot of progress has been made since concept drift was first discovered one cannot avoid the overhead of integrating methods to detect concept drift. Furthermore, efficiency always decreases with time, even when using the most successful detection methods. One of the greatest advantages of our methodology is overcoming the concept drift by using sliding window that contains all the information required to analyze quality drifts.

```
Decision tree:
:
      Alpha = 4.6: 1 (21/6)
      Alpha = 9.0:
      :...F <= 5.0:
          :...N <= 1.2: 1 (8)
          :   N > 1.2:
          :   :...O3 <= 8: 1 (6/2)
          :       O3  > 8: 2 (16/6)
          F  > 5.0:
          :...NF3  > 13: 1 (6/1)
              NF3 <= 13:
              :...Delay = 0.00: 3 (16/6)
                  Delay = 0.14: 4 (2)
                  Delay = 1.05: 2 (1)
                  Delay = 1.17: 2 (8/4)
                  Delay = 0.15:
                  :...N <= 1.2: 3 (4)
                      N  > 1.2: 4 (7/1)
:

Error: 29.6%
```

Fig. 1 Usability demonstration of decision tree output example. Interpretation may require more efforts

Another issue that many of the techniques mentioned above fail to deal with relates to the usability of the data presented to the user. With traditional data mining solutions the output of the model tends to be hard to interpret due to the training data set that is usually composed of measurements produced by the machines. Further steps are usually required to fully interpret the model and to draw from it meaningful conclusions. Figure 1 presents an example of a typical output from a decision tree that was used in a traditional data mining approach. As can be seen, transforming this model into useful rules is not that simple.

In this paper we introduce the root-cause problem and define different types of failure. We present a new methodology for finding the root-cause for scraps in a manufacturing process. The advantages of the new methodology include:

- Automation—Using an incremental model instead of on-demand operation produces better understanding and sensitivity to the behavior of scraps. In addition, it naturally overcomes the concept drift.
- Overcoming the dataset problem—Unlike traditional data mining approaches, the proposed methodology can cope with difficulties arising from high dimensional data, small sample sizes and missing data.
- Usability—The model is immediately interpretable and does not require any additional analytical or computational effort.

We also propose a novel method for evaluating models characterized *by multi-classifications* and *classification with approximations*. We use the new method to evaluate our methodology and to optimize model parameters and thresholds.

The root-cause problem

We now define the root-cause problem. We assume that a processing batch consists of k identical products $\{P_1, P_2, \ldots, P_k\}$. Each product undergoes w steps $\{S_1, S_2, \ldots, S_w\}$ in sequence to be finished. We also assume that there are n manufacturing machines $\{M_1, M_2, \ldots, M_n\}$ in the factory. Note that some machines may be used more than once for producing the same product. The *manufacturing mapping relation* maps each step $S_i\{S_1, S_2, \ldots, S_w\}$ to a list of suitable machines. The *manufacturing process relation*, based on the relation schema $\{PID, S_1, S_2, \ldots, S_w, C\}$, can be used to record the processing information of each product P_i and its outcome. Among the attributes in the relation, PID is an identification attribute used to uniquely identify the products; S_i is a context attribute associated with a pair ⟨step, manufacturing machine⟩ and indicating that the manufacturing machine is used in a certain step; and C is a classification attribute that states whether a product is scrap or not.

Figure 2 illustrates a manufacturing mapping relation with four steps ($w = 4$) and ten machines ($n = 10$). Figure 3 shows a manufacturing process relation that uses the manufacturing mapping relation presented in Fig. 2. That is to say, the relation is used to record five steps ($w = 5$) and processing information from ten machines ($n = 10$) for a batch of five products ($k = 5$). The first tuple shows that product P_1 passed through stage 1 on M_1, stage 2 on M_2, stage 3 on M_9, stage 4 on M_3, and that its classification result shows a defect ($C = 1$). Other tuples have similar meanings.

The goal is to identify the root-cause for a given manufacturing process relation tuple. In general, we can have more than one root-cause for a certain scrap, and each root-cause can contain more then one ⟨step, machine⟩ pair. From these notations we can derive the model's input and output. Given a product's manufacturing route and scrap classification, the model's output (root-cause), will be a set of clusters thus there are chains of ⟨step, machine⟩ pairs from the product manufacturing route where each ⟨step, machine⟩ can appear only once in the clusters. These notations are best described using a commonality graph.

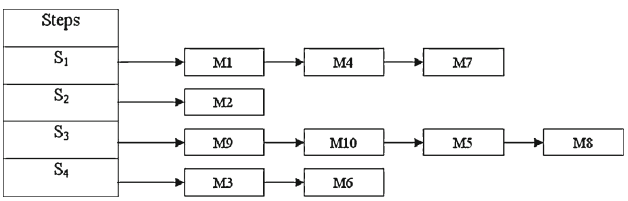

Fig. 2 A manufacturing mapping relation with five steps ($w = 4$) and ten machines ($n = 10$). Each tuple represents a step and a list of machines that can process the step

Product ID	S_1	S_2	S_3	S_4	C
1	M_1	M_2	M_9	M_3	1
2	M_1	M_2	M_8	M_6	0
3	M_7	M_2	M_5	M_3	1
4	M_7	M_2	M_{10}	M_3	1
5	M_4	M_2	M_5	M_3	0

Fig. 3 A manufacturing process relation that makes use of the manufacturing mapping relation presented in Fig. 2

The commonality graph spreads the manufacturing routes to sub-components connected by a commonality feature. The commonality graph is composed of commonality nodes. Each commonality node contains a chain of ⟨step, machine⟩ pairs which represent a portion of the chain from the lower commonality node connected to it. Taking two or more commonality nodes from the same depth can reveal which commonality node is directly related to those that have been selected. From a quick glance, the commonality graph looks very similar to a lattice network or a combinatorial network in the process of decomposing into all possible combinatorial options.

The commonality graph is composed of two main layers. The upper layer describes all possible root causes for all possible manufacturing routes. The lower layer describes all possible manufacturing routes. Crossing from the upper layer to the lower one will increase the manufacturing route dimensionality in the vertexes while from the inverse way, will decrease it. The commonality graph best describes the commonality relationships between the different components in the system. Figure 4 presents the commonality graph of a certain beverage manufacturing process that is composed of three steps and two possible values for each step.

Failures in the semiconductor manufacturing may arise from different reasons and contexts. In order to better distinguish between the different failures, we will review them and examine how they affect the input data. For any given time, one or more of the following failures may appear in the manufacturing process.

- *Single machine with a specific failure*—We place a failure in this category if the failure was due to a specific problem in one of the machines on the manufacturing floor. This failure may cause only small to medium drift in the manufacturing yield depending on the excessive use of the problematic machine. Yield management may interpret this phenomenon as random failure. This phenomenon has a high probability of occurring and it can

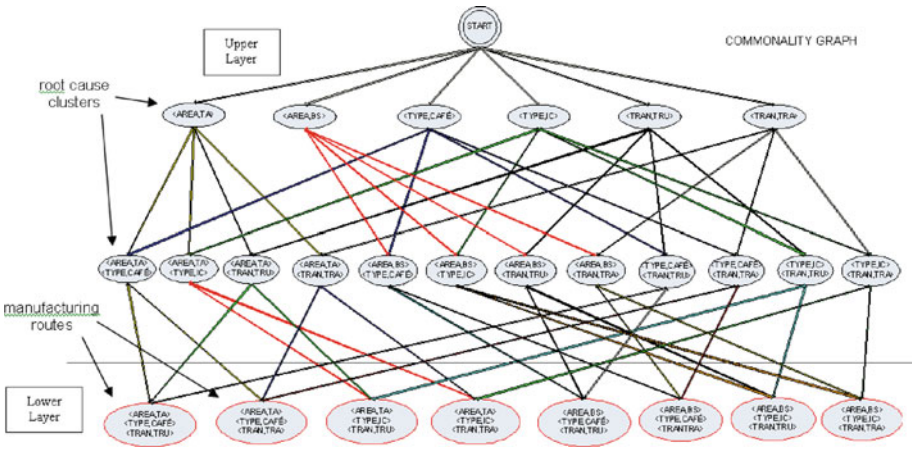

Fig. 4 A commonality graph of a certain beverage manufacturing process composed of three steps with two possible values for each step

be tied to human error or environmental changes or other such similar events. Whenever a problematic machine is used in many manufacturing steps, scraps will appear after engaging $\langle \text{step}_i, \text{problematic machine} \rangle$ where step_i is a group of steps that the problematic machine can process.

- *Many machines, each with a specific failure*—We place a failure in this category if it was due to a specific problem in one or more of the machines on the manufacturing floor. This failure will probably cause a high drift in the manufacturing yield depending on the amount of problematic machines. Yield management may interpret as spatial event which may cause a manufacturing stoppage. This phenomenon has only small probability of occurring and yet it can be tied to a natural event or sudden problem with the clean room or any similar spatial events. In cases where the problematic machines are used in many manufacturing steps, scraps will appear after engaging $\langle \text{step}_i, \text{problematic-machine}_i \rangle$ where step_i is a group of steps that the problematic machine can process and $\text{problematic-machine}_i$ is a group of problematic machines.
- *Many machines, process combination failure*—We place a failure in this category if the failure was due to a specific process combination problem arising from a specific sequence of machines on the manufacturing floor. This failure may cause a small drift in the manufacturing yield due to the small probability of a certain product to be processed using the problematic combination of machines. Yield management may interpret this quality drift as a random event and therefore it will not attract further attention. This phenomenon has only a small probability of occurring. In cases where the problematic sequence of machines is used, scraps will appear after engaging $\langle \text{step}_1, \text{machine}_1 \rangle$, $\langle \text{step}_2, \text{machine}_2 \rangle \ldots \langle \text{step}_i, \text{machine}_i \rangle$, hence the problematic sequence.

Methodology

In this section we describe in detail the new approach.

Data structure

The methodology's data structure saves the minimum relevant data for finding the root-cause. The data structure consists of the raw data that corresponds to the lower layer of the commonality graph. The model constructs an object from each ⟨step, machine⟩. For each constructed object, the model maintains two queues with a predefined size. One queue, the Non-Scrap PID (NSP) queue saves the latest product IDs of the products that were classified as non-scrap and were also engaged through the ⟨step, machine⟩ pair. The other queue, Scrap PID (SP) queue is similar to the first queue, but for those which are classified as scrap.

Updating the model is done with each new transaction, regardless of whether it was classified as scrap or not. For each new transaction, we examine its ⟨step, machine⟩ manufacturing chain. For each ⟨step, machine⟩ pair, we update the relevant object in the data structure. If the transaction was classified as scrap, we update the object's SP queue with the new product ID; otherwise we update the NSP queue. Using the queues with a predefined size denotes the important feature of this data structure. At any given moment the data structure includes a view of each ⟨step, machine⟩ and its latest information. The data structure is easy to construct and extremely compressed. It enables the methodology to handle extreme highly dimensional data very fast.

The root-cause algorithm: clustering common failures

The root-cause algorithm consists of two steps: clustering common failures and determining the root cause. Since the data structure saves data about each possible ⟨step, machine⟩ in the manufacturing process, the first two types of failures,

Single machine with a specific failure and *Many machines, each with a specific failure* can be easily handled, mainly because these failures consist of a specific problem in each of the machines involved . On the other hand, finding connections between failures is much more complicated since for each combination described in the upper layer of the commonality graph, historical data needs to be saved. Using a *Squeezer* algorithm (Zengyou et al. 2002), we successfully overcome the greatest disadvantage of the brute force strategy of saving data on different combinations, and therefore we are able to handle the third failure type, *Many machines, process combination failure*.

On the assumption that ⟨step, machine⟩ pairs belonging to the category *Many machines, process combination failure* will have approximately similar values in their SP queue, we can use clustering methods to find commonalities between failures. Failures within the category *Many machines, process combination failure* may occur when using a specific problematic combination of two or more ⟨step, machine⟩ pairs in the product route. When this happens, this type of failure will appear in most of the manufacturing scraps and therefore will eventually end up creating the same SP queue for each of the objects that are responsible for the common sequence failure in the data structure.

Since the *Squeezer* algorithm is incremental, we can use the data structure to go over each ⟨step, machine⟩ pair in the scrap manufacturing route while activating the algorithm on the objects belonging to the SP queue. Each created cluster is characterized by a common SP and NSP queues. Since the clustering algorithm is only done on the SP queue, the NSP queue will be composed of the intersection of each NSP queue belonging to the cluster elements.

At the end of the clustering phase, we produce all the clusters with a common SP queue from the manufacturing scrap route. This group of clusters is eventually a subset of the root-cause clusters belonging to the manufacturing scrap route in the commonality graph. This group of clusters contains without exceptions, all ⟨step, machine⟩ pairs in the manufacturing route, whether they are clustered into groups or not. A single ⟨step, machine⟩ pair can only be in one cluster in the group. In the next phase we will examine each cluster in order to determine if it can be classified as a root-cause or not.

The root-cause algorithm: determining the root-cause

In the second phase of the root-cause algorithm we develop a novel method to determine if a single cluster can be classified as root cause. We exploit both the SP and NSP queues of the cluster to calculate and distribute weights to each element in the NSP queue. We then accumulate the weights and compare them against a pre-defined threshold to determine if the cluster can be classified as root-cause. By merging both the SP and NSP queues into a single queue with respect to the latest order, we can observe recent changes in the cluster productivity behavior.

We now describe two interesting merging scenarios. In the first extreme scenario, all product IDs belonging to SP queue are larger than the largest product ID found in the NSP queue. This means that all product IDs in the SP queue are closer to the current point of view than the product IDs in the NSP queue. In this extreme scenario, we are very convinced that the cluster is indeed a root-cause of the problem. Looking at the extreme scenario from the opposite point of view, in which all the product IDs belonging to the SP queue are smaller than the smallest product ID found in the NSP queue, will lead to the opposing conclusion that this cluster cannot be classified as the root-cause.

These two unique merging scenarios can define the scope of the weighting calculation method, mainly because they best define if a cluster can be classified as root-cause or not. We can also refer to these merging scenarios as the edges in our weighting function, giving them the highest and the lowest scores.

We now fully describe our weighting method:b

For each cluster we calculate the following weights according to the following method:

1. We first merge the cluster's SP and NSP queues and sort them from the latest to the older.
2. We then distribute weights to each PID in the NSP queue according to its place in the merged queue using Eqs. 1, 2 and 3.

$$W(s_i) = (1 - PLACE(s_i) \cdot k) \cdot k \tag{1}$$

$$PLACE\,(x) = \sum_{\forall f_i > x} f_i \tag{2}$$

$$k = \frac{1}{q} \tag{3}$$

3. We then accumulate the weights and compare this value to a predefined threshold; if the value is smaller than the predefined threshold we classify the cluster as root-cause.

In Eq. 3, q refers to the predefined size of the SP and NSP queues. In Eqs. 1 and 2, s_i refers to examined product ID in the NSP queue, while f_i refers to certain product ID in the SP queue. Equation 2 calculates the number of product IDs in the SP queue which is bigger than s_i, i.e., the younger product IDs. Equation 1 denotes a value ranging from 0 to $1/q$, where the value of $1/q$ is given to a non-scrap product ID which is younger than any product ID in the SP queue. On the other hand, the value 0 is given to a non-scrap product ID which is older than any product ID in the SP queue. If the queues sizes are even and a non-scrap product ID is

younger than half of the product IDs in the SP queue, then, it is assigned a weight equal to half of $1/q$.

We can now check our extreme merging scenarios. For the first one, which represents perfect root-cause classification, every product ID in the NSP queue is assigned a 0 value weight. Since the number of the product IDs is exactly the size of the queue, accumulating the values will denote 0. Since the predefined threshold ranges from 0 to 1, we determine that the first extreme scenario will be classified as root-cause. On the other hand, in the second scenario that represents rejection of the root cause classification, every product ID in the NSP queue is assigned $1/q$ value weight. Since the number of the product IDs is exactly the size of the queue, accumulating the values will denote exactly 1. We will therefore determine that the second extreme scenario will not be classified as root-cause.

The predefined threshold represents the sensitivity of the root-cause algorithm to the root-cause classification. Any change will affect the performance of the methodology. Of course, each application requires its own level of sensitivity that is directly tied with the yield manufacturing performance.

In the next section, Evaluation, we examine different threshold values in order to find the optimal value. Interestingly enough, we demonstrate that other model parameters can also greatly affect the performance of the model.

Evaluation

Traditionally the evaluation of data mining algorithm has focused on evaluating two-class classification using ROC curves (Rokach 2010). However, in the root-cause detection problem, the classification consists of a set of labels which represents the set of root causes (also known as multi-label classification). Moreover, the root causes can be inter-connected via hierarchical relations as indicated in the commonality graph.

Classifications which consist of a set of labels can expand or not the real classification by adding irrelevant information. In other words, both groups of real and predicted classification may contain similar labels, hence the term, classification with approximations.

Although the approximations that we derive from this form of classification do not provide us with the exact answer, nevertheless, regardless of the irrelevant information that has been added to the classification, our current state of knowledge regarding the root-cause is significantly improved. Furthermore, in real-life, defects can arise from different sources at a time, *multi- classifications*, creating more than one problematic root-cause.

Evaluating classifiers characterized with approximations using the traditional evaluation methods may not take into account their proximity to the real classification. For example if the real classification was: "angry", "sad" and "embittered" and the classifier indicates "angry" and "sad", traditional methods will indicate that the classifier was wrong. However, one cannot ignore that the classifier was 66.6% right, and therefore our knowledge was significantly improved. A simple approach to evaluate such classifiers with the traditional methods of confusion matrixes and ROC curves is to transfer the problem into traditional two-class, binary classification by deciding that a certain classifier is either right or wrong by using a 50% rule. If the classification is correct in more than 50% of the labels, we refer it as correct classification. Otherwise, if less than 50%, it will be considered as misclassification. This kind of approach can precisely evaluate classifiers characterized with approximations when handling problems which contain small amounts of category labels. However, when classifications become more complex, thus include more category labels, this kind of approach fails to correctly evaluate them.

Figure 5 illustrates the complexity of evaluating classifiers that classify with approximation. The figure describes the real classification {3-1}; thus, the root-cause uses machine number 3 to process step 1. For the example, we will assume that product X with manufacturing route {1-1, 2-2, 3-1} has been classified as scrap. By looking at the commonality graph we also notice other clusters which may have been misclassified by the classifier. Luckily some of these clusters may significantly improve our knowledge. For example, if the classifier indicated that the root-cause is cluster {2-2, 3-1}, then a

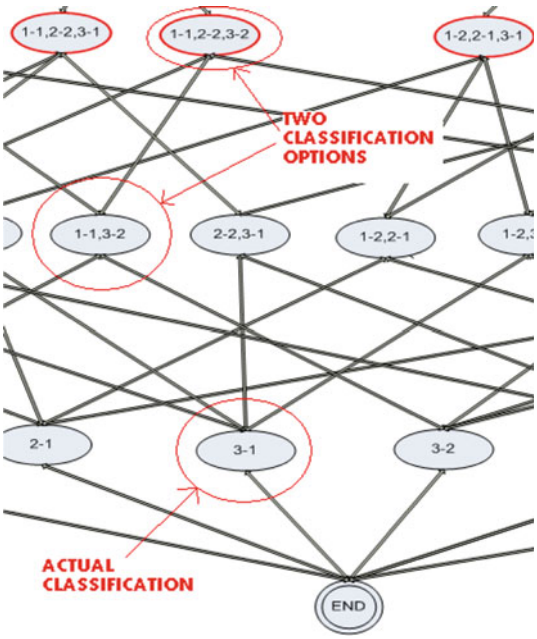

Fig. 5 Evaluating models with classifications that are close to the actual classification. The figure shows a commonality graph in *which* two close classifications reside next to the actual classification

portion of the answer still stands—machine 3 in step 1 indeed caused the failures. Although the classifier added "irrelevant data", (2-2), we still saved a third of the energy we would have invested without using this information since tracking the root-cause of the full scrap manufacturing route, {1-1, 2-2, 3-1} was the only piece of information we have.

Increasing the manufacturing route dimensionality even further will increase the complexity of finding the root-cause. Hence, enhance our appreciation to any knowledge improvement arise from using classifiers characterized with approximations. The potential of knowledge improvement is the key advantage of classifiers characterized with approximations comparing to traditional classifiers when tackling problems similar to root-cause identification. Therefore, we cannot ignore the need for strong methods to evaluate these unusual but potentially useful classifiers.

Another way to evaluate a model's correctness is by developing a method that grades classifiers according to the distance from the predicted root-cause to the real root-cause cluster on the commonality graph. Naturally, among all possible clusters, the clusters that improve our knowledge are those that possess a dimensionality that is equal to or higher than the real-root cause cluster and which are also connected in a direct or indirect commonality link to the real root-cause cluster. Since those clusters with the closest dimensionality are the ones that best improve our knowledge, they will be compensated with a greater score than those that do not. Following this intuition, we can define distances between different root-cause clusters using Eq. 4.

Given a root-cause cluster A with n dimensions and root-cause cluster B with m dimensions, the distance from A to B is defined as:

$$DISTANCE(A,B) = \begin{cases} n-c & n > m \text{and} \quad c > 0 \\ m-c & m \geq n \text{and} \quad c > 0 \\ t & c = 0 \end{cases} \quad (4)$$

where c is the commonality dimension between A and B and t is the manufacturing route dimensionality ($c < m, n$). In other words, the similarity distance between A and B is simply the maximum number of unagreed dimensionalities. When the common dimensionality is equal to 0, the distance is the number of dimensions of the manufacturing route. If $c = m = n$, then $DISTANCE(A,B) = 0$. If root-cause clusters A and B have $c = 0$ then $DISTANCE(A,B) = t$. So far we described the distance between two clusters, but in order to address multi-classification cases, we have to also define the distance between more than two root cause clusters.

Given root-cause clusters, A, B, C and D, and their distances from each other, we would like to know the distance between A and C to B and D, meaning, $DISTANCE(A \cap C,\ B \cap D)$. In order to find these distances, we use the following similarity distance matrix:

	Reported problematic	
	B	D
Real problematic		
A	$DISTANCE(A, B)$	$DISTANCE(A, D)$
C	$DISTANCE(C, B)$	$DISTANCE(C, D)$

Now we can define the full evaluation method as follows: given n—real problematic root-cause clusters, and m—reported root-cause clusters and a cost matrix $n \times m$. The model's error is simply the minimum cost between all possible assignments divided by the maximum possible cost. In order to solve the minimum cost problem, we exploit the Hungarian algorithm to solve the assignment problem in computational complexity of $O(n^3)$.

Recall the case presented in Fig. 5. The manufacturing route of product X which has been classified as scrap is {1-1, 2-2, 3-1}. Assuming that the real root-causes are {3-1} and the reported root-causes are {2-2, 3-1}, {2-1}, we can now build the similarity distance matrix for the example:

	Reported problematic	
	{2-2, 3-1}	{2-1}
Real problematic		
{3-1}	d_1	d_2
{virtual cluster*}	d_3	d_4

where:

d_1 *DISTANCE* ({3-1}, {2-2, 3-1}) = 1

d_2 *DISTANCE* ({3-1}, {2-1}) = 3

d_3 *DISTANCE* ({virtual cluster*}, {2-2, 3-1}) = 3

d_4 *DISTANCE* ({virtual cluster*}, {2-1}) = 3

In order to be able to connect all reported classifications, we introduce virtual clusters. Without using the virtual cluster, the minimum cost of assignment is 1. But this situation ignores the extra cluster, {2-1}. Adding a virtual cluster, where the distance to it always equals the manufacturing route dimensions, will achieve the required balance by increasing the minimal cost to 4 instead of 1. Finally, the model error is equal to 4 out of 6 (minimum assignment cost divided by the maximum possible costs), a 66.666% error. This number also makes sense when looking at the spread elements in the reported clusters—66.666% of the reported elements are missed classifications.

Usually, confusion matrixes are used to evaluate two-class classifiers. However, we can utilize *multi-similarity distance* to generalize the confusion matrix for multi-label classifications with approximations. Since the multi-similarity distance approach outputs the accuracy in the range of [0,1], we can easily use these values for creating a confusion matrix. For example, let's consider the case that the similarity distance assigns a 65% accuracy to a certain classifier. Then, if it was a "positive" instance, meaning that the classifier was suppose to be 100% accurate, the score will denote 0.65 in the confusion matrix's positive-positive quarter, and will decrease the positive-negative quarter by 0.35.

Experimental setup

Simulation tool

To evaluate our new methodology, we developed a tool to simulate a production line with a variable number of manufacturing steps where each step can be processed on different machines and each machine can process more than a single manufacturing step. The simulation itself consists of two major phases. In the first phase the simulation fully maps the relationship between the manufacturing process and the various machines necessary to implement it by randomizing the machine list to each manufacturing step from a fixed number of available machines.

The second phase consists of preparing the dataset. The dataset itself is also divided into two parts. The first part consists of normal manufacturing production. In this part we produce a small portion of data, which is characterized by a constant scrap rate that has been created from abnormal behavior which we cannot really relate to any specific root-cause. The second part of the data is reserved for simulating poor quality drift, in which our random root-cause produces more scrap than usual. Basically, we randomize a manufacturing route. If the manufacturing route contains our root-cause cluster, the final classification will have greater chance of being scrap. A steady period of production followed by a poor quality drift period completes the full picture of the simulation. The complete simulation presents a very close approximation to a real scenario. The simulation assumes a normal distribution in all coincidental aspects. The simulation produces rows of data ordered by time, meaning that each following row is newer than the previous one and therefore the larger the product ID, the newer the lot is. The simulation tool was developed in C# using Virtual Studio 2005. The output file that the simulation produces is compatible for Weka software package (Frank et al. 2005).

Datasets

In our experimental study we utilized the simulation tool to produce datasets that fully simulate a semiconductor fabrication process line. The datasets simulate three different types of failures: (1) *Single machine with a specific failure*; (2) *Many machines, each with a specific failure;* and (3) *Many machines, process combination failure.* Table 1 summarizes the three different configurations.

As one can notice, the third configuration is slightly different than the first and second configurations since we substantially increased the size of the problematic training size and decreased the number of machines for each operation. This was done due to the low probability of randomizing an entry when both of the ⟨step, machine⟩ pairs are presented in the manufacturing route (The probability is exactly the multiplication of both probabilities when using a certain ⟨step, machine⟩ pair in the manufacturing route). By increasing the size, the model can "feel" the drift caused by this combined failure. Another difference appears in the third configuration where the probability for scrap in a stable manufacturing period decreases from 0.1 to 0.05, while the probability for scrap in an unstable manufacturing period in the problematic root-cause cluster has been increased from 0.85 to 0.9. The reason for this, as explained below (see next section) is related to the model's ability to locate third configuration failures.

Metrics

In the experiments that we carried out, we evaluated the methodology by focusing on two dimensions, fast detection of the root-cause, and second, following detection, preserving accuracy by correctly classifying the incoming scraps that followed. We will therefore define and describe the metrics involved:

- Number of scraps until discovery—This parameter counts each new scrap that is directly related to the simulated root-cause until the root-cause has been successfully discovered. If more than one root cause exists, an averaging technique will be used. Of course, the lower the value of this parameter, the lower the scrap costs.

Confusion matrix—Classification systems are usually evaluated using a confusion matrix which contains information about actual and predicted classifications performed by a classification system. The performance of such systems is commonly evaluated using the data in the matrix. As noted earlier, using our multi-similarity distance approach makes it possible to generalize the confusion matrix to multi-classifications with approximations.

Experimental results

In this section we present nine different experiments. In the First experimental series, we get a first glance into the model's performance while discussing the adjustments of the predefined parameters and thresholds. Each experiment in the First experimental series is different in respect to two dimensions, the simulation data set and the model's parameters. In this first trial, we also outline the model's advantages and limitations. In the Second experimental series we present a configuration approach which overcomes these limitations, and a general integrated approach that positions this methodology as a leading approach. In the Third experimental series we present more experiments and compare our new methodology to the well known decision tree induction algorithm

Table 1 Three simulation configurations for three datasets used in the experimental section

Simulation parameters	First configuration	Second configuration	Third configuration
Failure type	*Single machine with a specific failure*	*many machines, each with a specific failure*	*many machines, process combination failure*
Training size for a stable manufacturing period	2,000 entries	2,000 entries	2,000 entries
The probability for scrap in a stable manufacturing period	10%	10%	5%
Training size for an unstable manufacturing period	1,500 entries	1,500 entries	5,000 entries
The probability for scrap in a unstable manufacturing period traversing the problematic root cause cluster	85%	85%	90%
The number of manufacturing steps	100	100	100
The number of machines for a manufacturing step	15	15	7
The number of available machines in the factory	250	250	250
The problematic clusters	{⟨step45, machine184⟩}	{⟨step17,machine18⟩}, {⟨step4, machine162⟩}, {⟨step58, machine41⟩}	{⟨step85,machine49⟩ ⟨step66,machine207⟩}

Each simulates a different type of failure

called C4.5. We evaluate the performance of the algorithms using our new evaluation method, multi similarity distance.

First experimental series

In this experimental series we focus on introducing the performance of our new methodology while discussing the model's parameters and thresholds. We briefly present below the parameters and thresholds that must be properly adjusted before using this methodology:

- Queue-size parameter—This parameter controls the maximum size of the latest data that is saved for each possibly ⟨step, machine⟩ pair, in order to find failure commonalities between them. It defines the size of the queues, NSP and SP that belong to each possible ⟨step, machine⟩ pair, according to the methodology's data structure. Increasing this value upward will automatically harden the clustering formalization since clusters will now have to agree on more entries between their SP queues. This will also have an impact on root-cause acceptance, making it more difficult to accept an increased number of entries due to the fact that the root-cause determination will now "dig deeper" into the past.
- *Squeezer* clustering threshold—Although *Squeezer* algorithm does not require the number of clusters as an input parameter, it still requires a special threshold to determine the acceptance of a new instance to an existing cluster. Increasing this threshold and new instance who wishes to enter an existing cluster will now have to agree on more entries between their SP queues, which will automatically harden the clustering formalization, leaving us with more widespread clusters. On the other hand, carelessly decreasing this threshold will cause much fewer, denser clusters, with little chance of being classified as a root-cause.
- Root-cause acceptance threshold—This threshold is being used during the root-cause determination phase. After accumulating the non-scrap PIDs weight, we compare it to a predefined threshold, the root-cause acceptance threshold. Increasing this threshold will cause more root-cause clusters to be accepted which, in turn, will lower the model's precision. On the other hand, it will also raise the model's sensitivity to new root-cause clusters.

Tables 2, 3 and 4 present the results of the first experimental series, each table presenting the results of one of the three configurations described above. Each table describes nine different experiments, in which the three parameters/thresholds of the model are configured differently. In addition to the accuracy (AC) we also report the following measures:

1. The true positive rate (TP) is defined as the proportion of positive cases that were correctly identified.

Table 2 *First experimental series—First configuration results*—Experimental results of running the model with different parameters

	Queue size	Squeezer acceptance	Root cause acceptance	Normal records	Scraps	Scraps related	Scraps until discovery	AC	TP	FP	TN	FN	P	g-mean1	g-mean2
1	5	4	0.2	1, 500	228	91	2	0.254	0.478	0.870	0.130	0.522	0.234	0.334	0.249
2	5	5	0.2	1, 500	228	91	2	0.619	0.696	0.429	0.571	0.304	0.499	0.589	0.630
3	5	5	0.1	1, 500	228	91	3	0.777	0.655	0.146	0.854	0.345	0.740	0.696	0.748
4	7	7	0.25	1, 500	228	91	4	0.896	0.908	0.111	0.889	0.092	0.840	0.873	0.898
5	7	6	0.143	1, 500	228	91	5	0.871	0.750	0.052	0.948	0.250	0.902	0.822	0.843
6	7	7	0.143	1, 500	228	91	5	0.884	0.750	0.030	0.970	0.250	0.942	0.841	0.853
7	10	8	0.1	1, 500	228	91	7	0.833	0.595	0.015	0.985	0.405	0.962	0.757	0.766
8	10	9	0.15	1, 500	228	91	7	0.907	0.762	0	1	0.238	1	0.873	0.873
9	10	10	0.2	1, 500	228	91	6	0.967	0.918	0	1	0.082	1	0.958	0.958

Table 3 First Experimental series—Second configuration results—Experimental results of running the model with different parameters

	Queue size	Squeezer acceptance	Root cause acceptance	Normal records	Scraps	Scraps related	Scraps until discovery	AC	TP	FP	TN	FN	P	g-mean1	g-mean2
1	5	4	0.2	1, 500	363	263	3	0.162	0.237	0.976	0.024	0.763	0.308	0.270	0.075
2	5	5	0.2	1, 500	363	263	3	0.250	0.320	0.890	0.110	0.680	0.415	0.364	0.188
3	5	5	0.1	1, 500	363	263	3	0.436	0.516	0.706	0.294	0.484	0.564	0.539	0.389
4	7	7	0.25	1, 500	363	263	4	0.470	0.541	0.707	0.293	0.459	0.657	0.596	0.398
5	7	6	0.143	1, 500	363	263	4	0.682	0.695	0.346	0.654	0.305	0.824	0.757	0.674
6	7	7	0.143	1, 500	363	263	4	0.682	0.694	0.346	0.654	0.306	0.823	0.756	0.674
7	10	8	0.1	1, 500	363	263	8	0.755	0.677	0.043	0.957	0.323	0.976	0.813	0.805
8	10	9	0.15	1, 500	363	263	6	0.824	0.785	0.074	0.926	0.215	0.965	0.870	0.853
9	10	10	0.2	1, 500	363	263	6	0.848	0.839	0.128	0.872	0.161	0.945	0.890	0.855

Table 4 First Experimental series—Third configuration results—Experimental results of running the model with different parameters

	Queue size	Squeezer acceptance	Root cause acceptance	Normal records	Scraps	Scraps related	Scraps until discovery	AC	TP	FP	TN	FN	P	g-mean1	g-mean2
1	5	4	0.2	5, 000	344	101	8	0.014	0.090	1	0	0.910	0.016	0.038	0
2	5	5	0.2	5, 000	344	101	14	0.603	0.266	0.283	0.717	0.734	0.242	0.254	0.437
3	5	5	0.1	5, 000	344	101	14	0.619	0.262	0.258	0.742	0.738	0.259	0.260	0.441
4	7	7	0.25	5, 000	344	101	73	0.756	0.464	0.103	0.897	0.536	0.684	0.563	0.645
5	7	6	0.143	5, 000	344	101	8	0.595	0.329	0.314	0.686	0.671	0.264	0.295	0.475
6	7	7	0.143	5, 000	344	101	73	0.765	0.464	0.088	0.912	0.536	0.722	0.579	0.651
7	10	8	0.1	5, 000	344	101	8	0.721	0.417	0.158	0.842	0.583	0.512	0.462	0.593
8	10	9	0.15	5, 000	344	101	71	0.776	0.733	0.206	0.794	0.267	0.611	0.669	0.763
9	10	10	0.2	5, 000	344	101	76	0.732	0.280	0.022	0.978	0.720	0.875	0.495	0.523

2. The false positive rate (FP) is defined as the proportion of negatives cases that were incorrectly classified as positive.
3. The true negative rate (TN) is defined as the proportion of negatives cases that were classified correctly.
4. The false negative rate (FN) is defined as the proportion of positives cases that were incorrectly classified as negative
5. The precision (P) is defined as the proportion of the predicted positive cases that were correct
6. Geometric mean (g-mean) of TP and P (g-mean1) and TP and TN (g-mean2)

In order to properly compare and arrive at a conclusion regarding the optimal parameter/threshold configuration, the same configurations were used for each table. For example, in the first experiment, the first entry in each table consists of the same parameters: *queue- size* equals 5; *Squeezer acceptance threshold equals* 4; and the *root-cause acceptance threshold* equal to 0.2. The structure of the tables is the same and they contain the same columns and measurements.

The necessity of adjusting these three parameters is particularly important. As can been seen from Table 2, experiment 9 presents better results than all the other experiments in the table. By adjusting the sliding window (*queue size*) to 10 transactions, the same as the *Squeezer acceptance threshold* and by using a *root-cause acceptance threshold* equal to 0.2, we can achieve a very high accuracy, 96.7% and an overall measure of 95.8% (g-mean1 and g-mean2). On the other hand, observing the worst performance, as presented to us by experiment 1, with an accuracy of 25.4% and an overall measure of 29.15% (by averaging g-mean1 and g-mean2) we can understand the need to properly adjust the model parameters and thresholds.

In order to properly understand why experiment 9 presented such a unique performance, we need to look at the following perception. In Table 2, the experiments which adjusted their *Squeezer acceptance threshold* to match their *queue size* achieved better performance. By matching the *Squeezer acceptance threshold* to the *queue size* we decrease the probability of clusters to merge. On the other hand, hardening the ability of clusters to merge is perfectly suitable to scenarios with failures from *Single machine with a specific failure*, category, because the simulate root-cause consists of single element cluster.

Another observation is the tradeoff resulting from increasing the sliding window, hence the *queue size.* By increasing the sliding window we indeed directly improve the general performance of model. On the other hand, we increase the number of scraps until the model correctly spots the root-cause for the first time. In some applications this tradeoff may be appealing from both points of view.

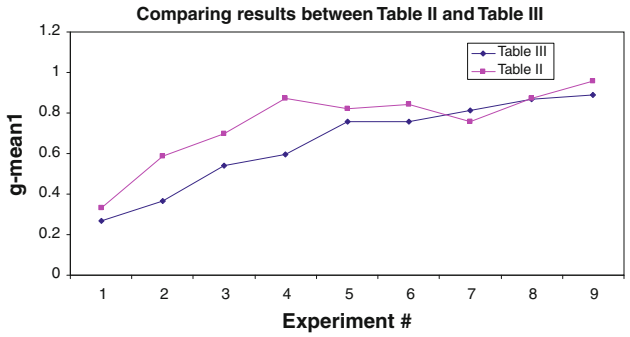

Fig. 6 Comparing Tables 2 and 3 using g-mean$_1$ metric. Table 2 presents a better performance since the Table 3 dataset simulates *many machines, each with a specific failure.* This type of *scenario* creates higher scrap rates that lead to more "noise" in the dataset, making it more difficult to achieve greater classification accuracy

The results from Table 3 are quiet similar to Table II. Only a small drift in the performance can be observed. The results in Table 3 also confirm that experiment 9 excels all other experiments in the table with an accuracy of 84.8% and an overall measure of 87.25% (by averaging g-mean1 and g-mean2). The worst performance presented also belongs to the first experiment with an accuracy of 16.2% and an overall measure of 9.75% (by averaging g-mean1 and g-mean2).

Since the scenario in the Table 3 dataset is meant to simulate *many machines, each with a specific failure, we* can expect the same results as with Table 2 since the model still does not need to formulate clusters with sizes of more than one. However, the model is now facing an increased scrap rate in the system due to the increasing number of manufacturing routes that can lead to scrap. The high scrap rate creates more "noise" in the system, which decreases the variations that the model is looking for. Finally it automatically leads to more mistakes in the model's classifications. This phenomenon can be easily seen by comparing each experiment from Table 2 to the equivalent experiment from Table 3 with the same model parameters/thresholds. Figure 6 presents a graph that presents this comparison using a g-mean$_1$ metric. Putting aside experiment 7, Table 2 presents a better performance which strongly emphasize our findings. Nevertheless, the performance of the model as presented in Table 3 is considered more than sufficient and the experiments in Table 3 proved that more than one failure can be spotted and handled accordingly.

The dataset that was used for the experiments in Table 4 is very unique and different from the basic notations that characterized Tables 2 and 3 datasets. The Table 4 dataset consists of a failure that can be only related to a specific combination of two processes. The dataset used in the Table 4 experiments consist of a 5% probability for scrap and 90% probability for scrap when the product traversed the two ⟨step, machine⟩ pairs. Even with a more relaxed scenario, the model presented a moderate performance compared to the ones presented in Tables 2 and 3. The best performance with an accuracy of 77.6% and an overall measure of 71.6% (by averaging g-mean1 and g-mean2) was produced from the Table 4 set of experiments using the configuration of experiment number 8 The worst performance, with an accuracy of 1.4% and an overall measure of 1.9% (by averaging g-mean1 and g-mean2), also belonged to the first experiment.

Another important metric that must be taken in account is the number of scraps that have been discovered until the root-cause is detected. In addition to the general decrease in performance relative to the performance presented in Tables 2 and 3, the *number of scraps until discovery* measurements also presented a decrease in performance, by climbing from ∼6, 7 scraps to ∼70 scraps compared to other failure types, when choosing the most accurate configuration. Since the best configuration configured for maximum accuracy, the model solved the root-cause mystery, but with more scraps to be "paid".

Table 4 results strongly emphasizes model limitations in discovering root-causes that contain clusters with more than one element. However, the impact of this limitation is quite small considering the fact that this kind of failure has a small chance of occurring.

Second experimental series

In the previous experimental series we focused on introducing the performance of our new methodology while optimizing the model's parameters and thresholds. We observed the model performance and accuracy, observing very good performances when tackling *Single machine with a specific failure* and *many machines, each with a specific failure* and moderate performance when trying to solve *many machines, process combination failure* type. However, these performances can be further improved at the expense of increasing the *number of scraps until discovery* measurement. In this experimental series we investigated even more combinations of the model's parameters and thresholds, with the main aim of drastically improving the performance of the model when tackling the category *many machines, process combination failure*.

Table 5 presents the results of the second experimental series. Experiments 1–6 present the results obtained from running the model on a dataset produced from the simulation tool using the third configuration as described in the experimental setup. Experiment 7 presents the results obtained from running the model on a dataset produced from the simulation tool using the first configuration and Experiment 8 presents the results obtained from running the model on a dataset produced from the simulation tool using the second configuration.

Table 5 Second Experimental series—Experimental results of running the model with different parameters

	Queue size	Squeezer acceptance	Root cause acceptance	Normal records	Scraps	Scraps related	Scraps until discovery	AC	TP	FP	TN	FN	P	g-mean1	g-mean2
1	22	17	0.2	5, 000	344	101	56	0.888	0.778	0.056	0.944	0.222	0.875	0.825	0.857
2	25	18	0.2	5, 000	344	101	30	0.840	0.683	0.101	0.899	0.317	0.719	0.701	0.784
3	25	19	0.2	5, 000	344	101	58	0.944	0.884	0.024	0.976	0.116	0.950	0.916	0.929
4	30	22	0.2	5, 000	344	101	60	0.952	1	0.071	0.929	0	0.872	0.934	0.964
5	33	24	0.2	5, 000	344	101	62	0.991	1	0.014	0.986	0	0.975	0.987	0.993
6	33	23	0.3	5, 000	344	101	37	0.848	0.727	0.103	0.897	0.273	0.744	0.735	0.808
7	33	24	0.2	1, 500	228	91	27	1	1	0	1	0	1	1	1
8	33	24	0.2	1, 500	363	263	24	0.987	0.982	0	1	0.018	1	0.991	0.991

Experiments 1–6 describe six different experiments in which the three parameters/thresholds of the model are configured differently. In Experiments 7 and 8 we demonstrate performance when using our optimal configuration obtained from Experiments 1–6 on datasets that represent the rest of the failure types. The optimal configuration is obtained following adjustment so that queue*size* = 33, *squeezer acceptance threshold*= 24, and *root cause acceptance threshold* = 0.2. The structure of the tables is the same and they contain the same columns and measurements.

As can be seen from experiments 1–6, further improvement is achieved using different parameter/threshold configurations. Experiment 5 excels compared to experiments 1–6, and we can now achieve better accuracy, 99.1% and an overall measure of 99.0% (g-mean1 and g-mean2). Comparing these results to the previous experiments, the accuracy increased from 77.6 to 99.1% and the overall measure from 71.6 to 99.0%.

In order to achieve these impressive results we adjust the parameters/thresholds by implementing an approach where we increase the *queue size* and *Squeezer acceptance threshold* while preserving the special ratio between them. This approach necessitates that formulated clusters must accept more entries in the SP queue. However, at the same time, the formulated clusters may also disagree on more entries. Using this approach also reduces the effectiveness of *root-cause acceptance threshold* and the model is now far less sensitive for any slight variation when adjusting the *root-cause acceptance threshold*. The reason for this lies with the increase of the *queue size* parameter which causes the model to further investigate the "past", reducing misclassification and increasing the accuracy. However, this approach to improve accuracy is usually on account of the*number of scraps until discovery* measurement. Raising the *Squeezer acceptance threshold* to 24 automatically hardens the clusters from being formulated by randomly causes. By raising the *queue size* to 33 we also increase the disagreed entries to 9, i.e., clusters can now be formulated with much more accuracy. This configuration approach even reduced the *number of scraps until discovery* measurement from ∼70 to ∼60 compared to Table 4. However, dropping to ∼30 scraps will be at the expense of accuracy and performance, as can be seen when comparing experiments 5 and 6.

Experiments 7 and 8 in Table 5 present the effectiveness of using the optimized approach on other failure types, *Single machine with a specific failure* and *many machines, each with a specific failure*. As can be seen, the performances are even further improved. In the case of the first configuration dataset, accuracy improved from 96.7 to 100% and the overall measure improved from 95.8 to 100% (g-mean1 and g-mean2). In the second configuration dataset, accuracy improved from 84.8 to 98.7% and the overall measure improved from 87.25 to 99.1% (g-mean1 and g-mean2). However, as mentioned

before, this high level performance usually comes at the expense of increasing the *number of scraps until discovery* measurement. In the case of the first configuration dataset, the *number of scraps until discovery* increased from 6 to 27 and in the second configuration dataset, the *number of scraps until discovery* increased from 6 to 24.

We now present below some guidelines for choosing the optimal model configuration. Using historical data, one can estimate the probabilities for each failure type and derive from them the desired configuration approach. If the possibility for *Single machine with a specific failure* and *many machines, each with a specific failure* is relatively high compared to *Many machines, process combination failure,* then the model's parameters/thresholds should be adjusted as follows: the *queue size* is set to around 10; the *squeezer acceptance threshold* is the same as *queue size;* and the *root cause acceptance threshold* is set to 0.2. This approach will guarantee a fast reaction and great sensitivity to any new root-cause. On the other hand, if the possibility for *Single machine with a specific failure* and *many machines, each with a specific failure* is relatively low, compared to *Many machines, process combination failure*, then the model's parameters/thresholds should be adjusted as follows: the *queue size* is set to 33; the *squeezer acceptance threshold* to 24; and the *root cause acceptance threshold* to 0.2.

This approach will guarantee high level performance on account of the fast reaction to any new root-cause. However, these probability assumptions may lead to poor performance since, as mentioned above, the wafer fabrication process is chaotic and unpredictable and therefore probabilities and assumptions are not recommended. In addition, tackling a situation without prior knowledge about the probabilities for each failure type will require a general integrated approach. Therefore, our approach for adjusting the model parameters and thresholds is simply to configure both at the same time. This involves running both methodologies on the same dataset where each model adjusted for different purpose. One, called *sensitivity component*, will be responsible for sensitivity and fast reaction to any new root-cause, while the other, called *accuracy component*, will be responsible mainly for discovering *Many machines, process combination failure* types and on enhancing the performance. In addition, the *sensitivity component* classifications only count when the *accuracy component* has nothing to indicate. Since both configurations are independent. This means that the *sensitivity component*, that consists of setting the *queue size* to around 10, and the *Squeezer acceptance threshold* to the same setting as the *queue size* will harden the threshold and make it almost impassible to any new cluster to formulate and will guarantee that there is no false clustering. The *accuracy component* consists of setting the *queue size* to 34, and the *squeezer acceptance threshold* to 24. Based on the experiments, this will guarantee almost 100% accuracy without creating any false alarms. These rules are based on empirical results and therefore will not always guarantee an optimal configuration. However, following these guidelines will guarantee good results.

Comparative study

In this experimental series, we compare our methodology with the general integrated approach to the well known C4.5 algorithm. Comparing an incremental algorithm to algorithm that optimizes its current state is not trivial. While the algorithm at the core of our methodology changes incrementally the C4.5 algorithm optimize its current dataset to formulate a decision tree that does not change over time. We will therefore exploit the C4.5 algorithm using the following method. We first precisely measure the number of scraps until discovery of the root-cause. This provides us with the first metric that we need for the comparison between our algorithm and C4.5. Afterwards, in order to estimate the consistency of the two algorithms, we build a new decision tree with each five related scrap intervals. For each new measurement, we truncate the data set accordingly and then run both algorithms in order to compare them.

Figure 7 presents the results of the third experimental series. Figure 7a, d, g presents the results obtained from running C4.5 (a); proposed methodology—*sensitivity component* (d); proposed methodology—*accuracy component* (g) on a dataset produced from the simulation tool using the first configuration as described in the experimental setup. Figure 7b, e, h presents the results obtained from running C4.5 (b); proposed methodology—*sensitivity component* (e); proposed methodology—*accuracy component* (h) on a dataset produced from the simulation tool using the second configuration. Figure 7c, f, i presents the results obtained from running C4.5 (c); proposed methodology—*sensitivity component* (f); proposed methodology—*accuracy component* (i) on a dataset produced from the simulation tool using the third configuration. As mentioned above, we compare performances using five related scrap intervals. The axes in the figure are exactly the same. Axis *X* represents the related scrap timeline, while axis *Y* represents the accuracy level which is calculated using the *multi-similarity distance* approach.

In each graph we can visually compare the discovery phase, hence the *number of scraps till discovery,* and approximately the extent to which classification accuracy has been preserved. As can be seen from Fig. 7a, d, g and Fig. 7b, e, h, the discovery rate of the proposed methodology was seven to nine times greater than that of the C4.5 algorithm. However, in Fig. 7c, f, i, the tables are turned when C4.5 overcomes the proposed methodology by presenting a discovery rate that was twice as effective.h

Figure 7c, f, i results together with those from Table 4 in the first experimental series demonstrate one of the model's

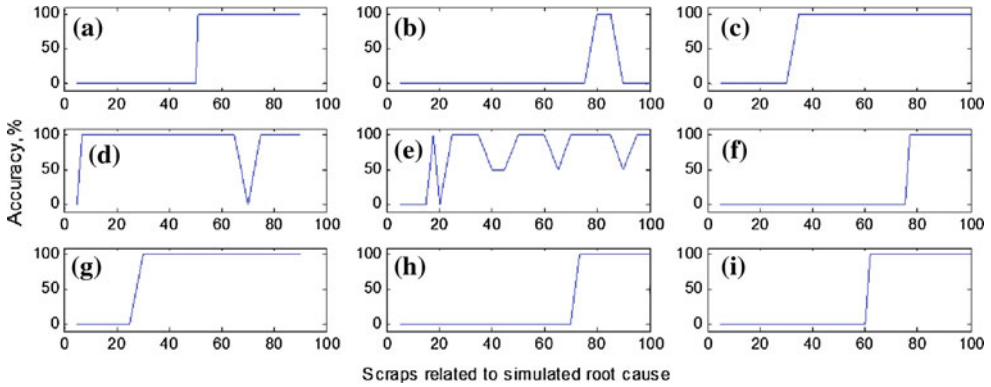

Fig. 7 Results of comparing C4.5 algorithm to the proposed methodology using the first configuration data set (**a, d, g**); second configuration set (**b, e, h**); and the third configuration set (**c, f, i**). C4.5 algorithms results (**a, b, c**); proposed methodology—sensitivity component results (**d, e, f**); and proposed methodology—accuracy component results (**g, h, i**)

weaknesses in quickly discovering root-causes that contain clusters with more than one element. The second experimental series demonstrates the tradeoff between preserving accuracy and quickly discovering root-causes. To cope with this tradeoff and to ensure or preserve the possibility of quick root-cause discovery, experiment number 2 in Table 5 from the second experimental series can serve as an alternative to the C4.5 algorithm. However, as noted above, this comes at the cost of preserving accuracy, The experiment mentioned above, presented 30 related scraps until discovery and with an overall accuracy of 75% (by averaging g-mean1 and g-mean2). The performance is quite poor compared to the 100% accuracy achieved by the C4.5 algorithm, Nevertheless, considering the fact that Many machines, process combination failure has a small chance of occurring. And when it does occur its impact on the manufacturing yield is negligible since the probability of a certain product traversing the problematic combination is very small. Furthermore, the weakness is quiet small compared to the results achieved in Fig. 7a, d, g and Fig. 7b, e, h.

Conclusions

Identification of the root-cause to quality drifts in manufacturing cannot only reduce manufacturing costs, but also improve manufacturing performance. However, conventional methodologies for identifying root-causes are restricted and dependent on experience and expertise. In this paper we have introduced the root-cause problem and defined different failure types. We presented a new methodology for finding the root-cause for scraps in a manufacturing process. The methodology consists of fast, incremental algorithm that can process extremely high dimensional data and handle more than one root-cause cluster at the same time. The methodology consists of three main stages: first, updating the model with each new instance and its classification; second, if the instance classified as scrap, common failures are then clustered using the *Squeezer* categorical clustering algorithm; and finally, determining for each cluster whether it can be a root-cause or not. Since the methodology classifications characterize with *multi- classification* and *classification with approximations*, we introduced a novel approach, *multi-similarity distance* for evaluating classifiers whose classification consists of a vector of categorical values, hence the term *classification with approximations*. And since there are times more than one classification vector, the term *multi- classification*. The new evaluation method can be easily adjusted for use with the known measurements, confusion matrix and ROC to compare and analyze.

The experimental results show that root-causes belonging to *Single machine with a specific failure* or *many machines, each with a specific failure* can be spotted with great accuracy and in a very short time compared to the C4.5 algorithm even with a fixed scrap rate and when the probability for scrap, when traversing through the root-cause, is less than 100% (experimental tests were measured at 85%). However, the performance slightly decreases in *many machines, each with a specific failure* with the number of root-cause clusters in the system due to the high scrap rate which causes lack of variation, which in turn weakens the ability of the model to spot the root-cause without projecting false alarms.

In the category *many machines, process combination failure* performance was moderate compared to the C4.5 algorithm. However, this limitation was totally removed in the *second experimental* series by increasing the *queue size* parameter, which caused the model to further investigate the "past". This feature was achieved at the cost of finding the root cause in a longer period of time. Tracking a root-cause to failure belonging to *many machines, process combination failure* category is indeed extremely hard, and will require high variation in the dataset. However, considering the fact

that this kind of failure has only a small chance of occurring and when it does occur its impact on the manufacturing yield is negligible, this limitation is relatively unimportant. These failures are usually swept away under the table. Thus, using the proposed methodology; this kind of failure can be discovered, even if it takes more scraps compared to other types of failures. Future research may propose and empirically compare other categorical clustering algorithms, propose alternatives to the root-cause determination algorithm, and evaluate the methodology on a real dataset.

References

Ben-Gal, I. (2006). Outlier detection. In O. Maimon & L. Rokach (Eds.), *Data mining and knowledge discovery handbook: A complete guide for practitioners and researchers* (pp. 131–146). New York: Springer, US.

Bergeret, F., & Le Gall, C. (2003). Yield improvement using statistical analysis of process dates. *IEEE Transactions on Semiconductor Manufacturing, 16*, 535–542.

Chang, P. C., Fan, C. Y., & Wang, Y. W. (2009). Evolving CBR and data segmentation by SOM for flow time prediction in semiconductor manufacturing factory. *Journal of Intelligent Manufacturing, 20*(4), 421–429.

Chen, T., Wang, Y. C., & Wu, H. C. (2009). A fuzzy-neural approach for remaining cycle time estimation in a semiconductor manufacturing factory—A simulation study. *International Journal of Innovative Computing, Information and Control, 5*(8), 2125–2140.

Choudhary, A. K., Harding, J. A., & Tiwari, M. K. (2009). Data mining in manufacturing: A review based on the kind of knowledge. *Journal of Intelligent Manufacturing, 20*(5), 501–521.

Duan, G., Chen, Y. W., & Sukekawa, T. (2009). Automatic optical inspection of micro drill bit in printed circuit board manufacturing using support vector machines. *International Journal of Innovative Computing, Information and Control, 5*(11(B)), 4347–4356.

Durham, J., Marcos, Von J., Vincent, T., Martinez, J., Shelton, S., Fortner, G., Clayton, M. & Felker, S. (1995). Automation and statistical process control of a single wafer etcher in a manufacturing environment. *Advanced Semiconductor Manufacturing Conference and Workshop IEEE/SEMI*, pp. 213–215.

Frank, E., Hall, M. A., Holmes, G., Kirkby, R., Pfahringer, B., & Witten, I. H. (2005). Weka:Data A machine learning workbench for data mining. In O. Maimon & L. Rokach (Eds.), *mining and knowledge discovery handbook: A complete guide for practitioners and researchers* (pp. 1305–1314). Springer.

Gardner, M., & Bieker, J. (2000). Data mining solves tough semiconductor manufacturing problems, *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 376–383.

Goodwin, R., Miller, R., Tuv, E., Borisov, A., Janakiram, M., & Louchheim, S. (2004). Advancements and applications of statistical learning/data mining in semiconductor manufacturing. *Intel Technology Journal, 8*, 325–336.

Haapala, K. R., Rivera, J. L., & Sutherland, J. W. (2008). Application of life cycle assessment tools to sustainable product design and manufacturing. *International Journal of Innovative Computing, Information and Control, 4*(3), 577–592.

Hu, H. C. H., Shun-Feng, S. (2004). Hierarchical clustering methods for semiconductor manufacturing data, *Proceeding of the 2004 IEEE International Conference on Networking, Sensing and Control, vol. 2*, pp. 1063–1068.

Hyeon, B., Sungshin, K., Kwang-Bang, W., Gary, S., & Duk-Kwon, L. (2006). Fault detection, diagnosis, and optimization of wafer manufacturing processes utilizing knowledge creation. *International Journal of Control, Automation, and Systems, 4*, 372–381.

Jemmy, S., Wynne, H., Mong, L. L. & Tachyang, L. (2005). Mining wafer fabrication: Framework and challenges, next generation of data-mining applications.

Kenneth, W., Thomas, P., & Shaun, S. (1999). Using historical wafer data for automated yield analysis. *Journal of Vacuum Science Technology A, 17*, 1369–1376.

Kittler, R., & Wang, W. (1999). The emerging role for data mining. *Solid State Technology, 42*, 45–58.

Rodrigues, P., & Gama, J. (2004). Prediction of product quality in continuous glass manufacturing process, *4th European Symposium on Intel Tech and Smart Adaptive Systems*, pp. 488–496.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review, 33*(1), 1–39.

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

Rokach, L., Romano, R., & Maimon, O. (2008). Mining manufacturing databases to discover the effect of operation sequence on the product quality. *Journal of Intelligent Manufacturing, 19*(3), 313–325.

Zengyou, H., Xiaofei, X., & Shengchun, D. (2002). Squeezer: An efficient algorithm for clustering categorical data. *Journal of Computer Science and Technology, 17*, 611–624.