Robust fault detection and clustering in semiconductor manufacturing processes

Oct. 8, 2021

Woong-Kee Loh
School of Computing, Gachon University

Contents

- Introduction
- Related Work
- Fault Detection Algorithm
- Variable Selection Method
- Clustering Algorithm
- Evaluation
- Conclusions
- References

Introduction

- Semiconductor Manufacturing Processes
  - Semiconductor manufacturing consists of many processes; each process is also composed of many sequential steps
  - Some manufacturing steps are performed continuously without any intermission (e.g., etching and lithography)
  - A sequence of continuous steps is called a *run*, and is performed in a series of black box-like equipment

- *Fault Detection and Classification* (*FDC*) [7,13]
  - Even if a fault has occurred in any run step, it can only be detected when the entire run has been finished
  - Various sensors are attached to manufacturing equipment
  - Values read periodically from each sensor collectively constitute a (streaming) time-series data
  - Multiple time-series data are input to sophisticated algorithms based on statistics, expert systems, and data mining

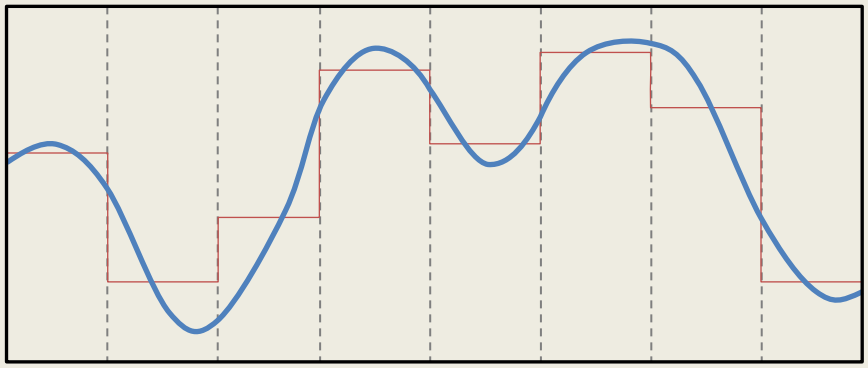
- Contributions of this study
  - We propose an algorithm for fault detection in semiconductor manufacturing processes
    - A modification of discord detection algorithm called *HOT SAX* [9]
  - We propose an algorithm for clustering runs using the result of our fault detection algorithm

- Evaluation of our algorithms
  - We used the experiment data obtained from real-world semiconductor etching processes
  - Our fault detection algorithm accurately distinguishes the normal and the perturbed (faulty) runs
    - Achieved 100% accuracy without false positive or false negative
  - Our clustering algorithm generated good clusters of runs having similar sources of faults

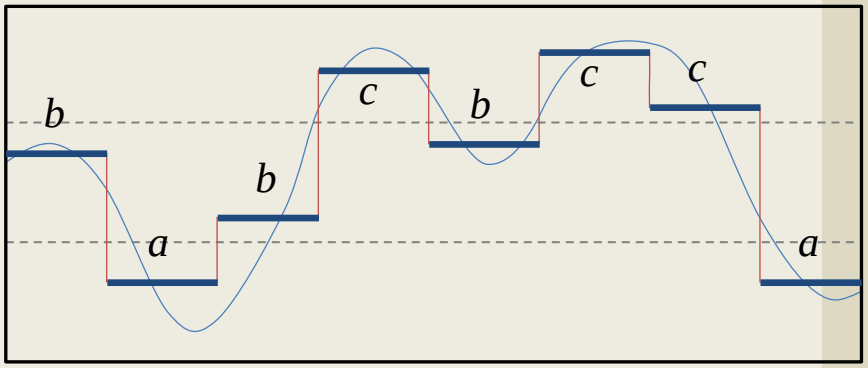
Related Work

- Representation of time-series
  - Compressed representation for efficient storage and computation of time-series
  - Discrete Fourier Transform (DFT), Piecewise Aggregate Approximation (PAA), etc.
  - Symbolic representation
    - Transforms continuous real values in time-series into a finite number of discrete symbols

- *Symbolic Aggregation approXimation* (*SAX*) [11,12]
  - A symbolic representation
  - Given two parameters *w* and *a*, a time-series *X* of length *n* is transformed into a sequence $\hat{X}$ of length *w*, where each symbol in $\hat{X}$ is obtained from a symbol set of size *a*



X (*n* = 128)

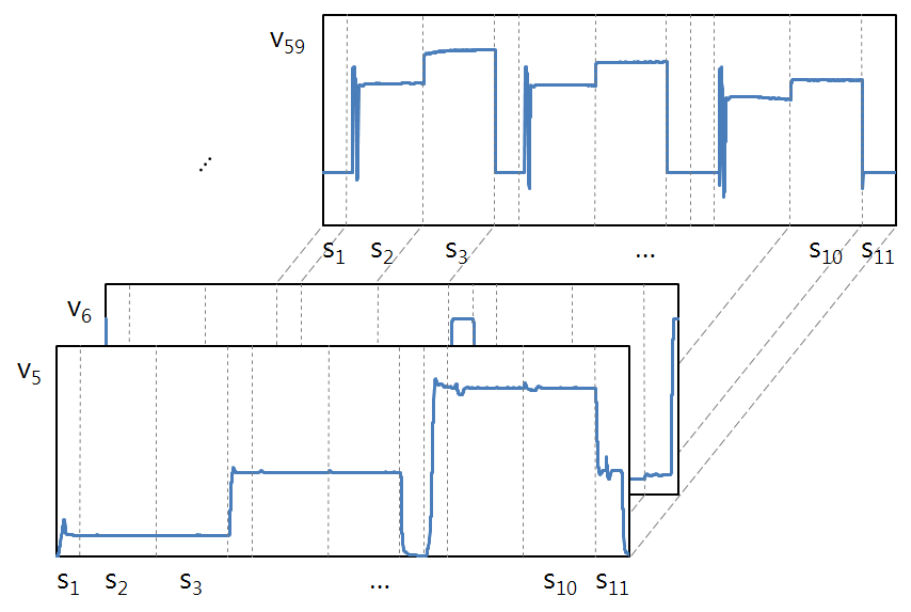


$\hat{X}$ = *babcbcca* (*w* = 8, *a* = 3)

- Applying SAX to time-series data mining
  - Discord detection using SAX [9, 17]
  - Finding motifs (the patterns appearing very frequently in a time-series) [3, 14]
  - Minimizing the number of parameters [10]
  - *iSAX*: efficient disk-based indexes for large-scale time-series databases [16]

Fault Detection Algorithm

- Stream sequences
  - While a run is being performed, the sequence of values from each sensor is assigned to a *variable*
  - Example: a run which consists of 11 steps (s_1 ~ s_{11}) and collects *stream sequences* for 55 variables (v_5 ~ v_{59})



- Our fault detection algorithm
  - Given a set of runs, it finds the runs that produced perturbed wafers
  - 2 run groups: model runs and experimental runs
    - All the model runs produced normal wafers, while a few of experimental runs produced perturbed ones
  - We employ the idea and terms introduced by the discord detection algorithm called *HOT SAX* [9]

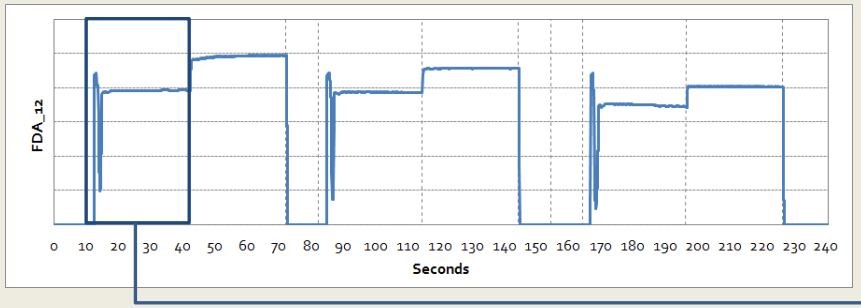
- Decision of fault
  - For every combination (variable *v*, step *s*), our algorithm checks:
    - $\min\{D(S'_{v,s}, M_i)\} > \max\{D(M_i, M_j)\}$
    - *Discord ratio* $R_D = \frac{D_{exp}}{D_{mod}} = \frac{\min\{D(S'_{v,s}, M_i)\}}{\max\{D(M_i, M_j)\}}$
    - M_i is a stream subsequence for (*v*, *s*) from a model run R_i
    - $S'_{v,s}$ is a stream subsequence from an experimental run to test
  - If $R_D > 1.0$, our algorithm takes it as an evidence of fault occurred in the corresponding step *s*.
  - If any combination in a certain experimental run reports a fault, the whole run is regarded as perturbed.

Decision of fault *cont'd*

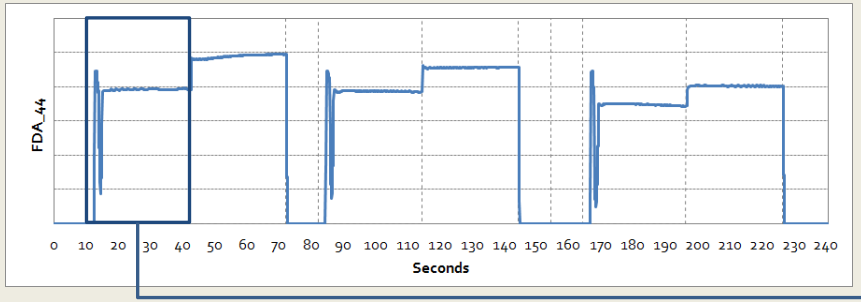
- Example: $v = 51$, $s = 2$

$$R_D = \frac{D_{exp}}{D_{mod}} = \frac{\min\{D(S'_{v,s}, M_i)\}}{\max\{D(M_i, M_j)\}}$$

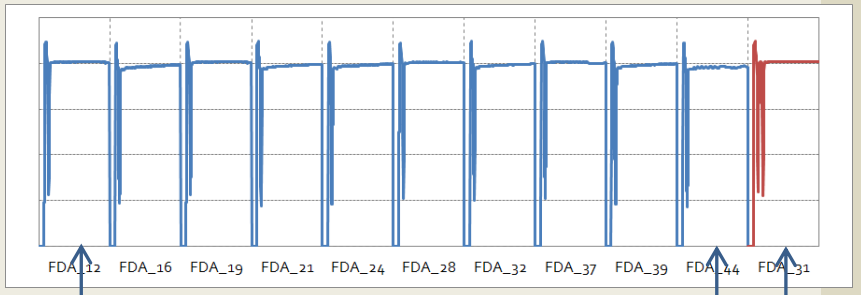
Model runs



⋮

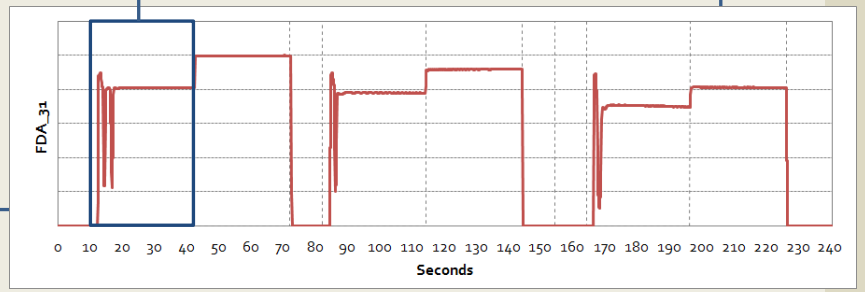


Subsequence concatenation



…

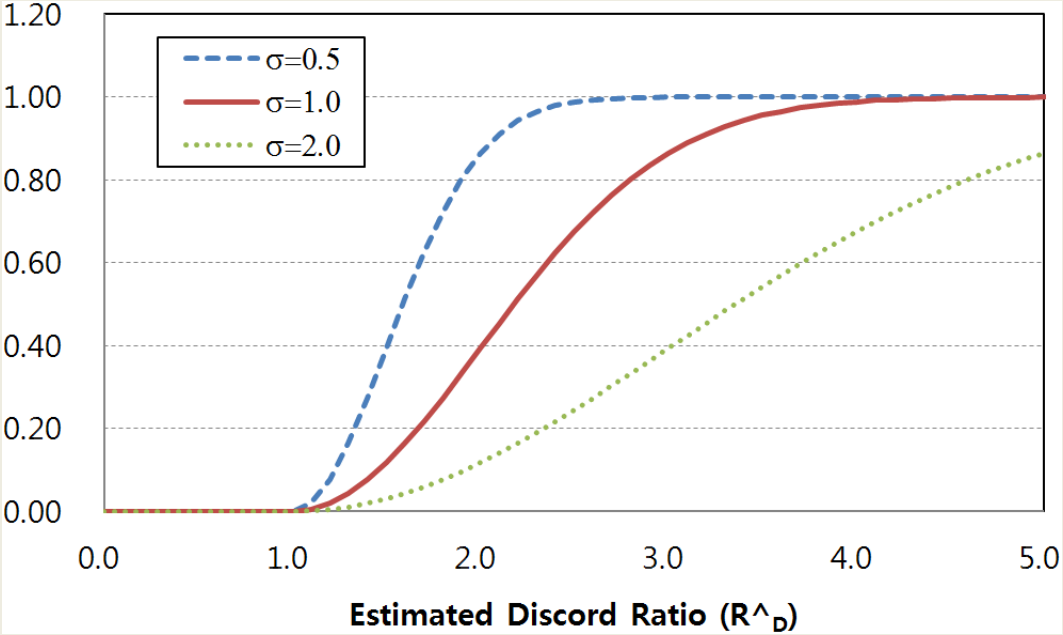
Experimental run



- Adopting SAX transformation
  - Actually, we use an estimate $\hat{R}_D$ (≥ 0) instead of R_D
  - $\hat{R}_D = \frac{\hat{D}_{exp}}{\hat{D}_{mod}} = \frac{\min\{MINDIST(\hat{E},\hat{M}_i)\}}{\max\{MINDIST(\hat{M}_i,\hat{M}_j)\}}$
  - $\hat{E}$, $\hat{M}_i$, and $\hat{M}_j$ are SAX-transformed subsequences of $S'_{v,s}$, M_i, and M_j, respectively

- Adopting SAX transformation *cont'd*
  - $\hat{R}_D$ > 1.0 does not necessarily imply R_D > 1.0
  - We define *fault probability function* $F(\hat{R}_D)$
  - $$F(\hat{R}_D) = \begin{cases} 0 & \text{if } \hat{R}_D \leq 1 \\ 1 - \exp\left\{-\frac{(\hat{R}_D - 1)^2}{2\sigma^2}\right\} & \text{otherwise} \end{cases}$$
  - If $F(\hat{R}_D)$ is close to 1.0, it is regarded that a fault has occurred in the corresponding step

- Adopting SAX transformation *cont'd*
  - $F(\hat{R}_D)$ graphs for a few σ values



- Differences of our algorithm from HOT SAX
  - While HOT SAX requires the length *l* of discord subsequence as an input, our algorithm derives the length from a run step
  - Our algorithm checks whether a stream subsequence $S'_{v,s}$ is the discord subsequence or not, while HOT SAX finds a discord subsequence that may be located at any position

- Rationale of adopting SAX transformation
  - SAX reduces the size of stream data dramatically
    - Given a parameter *w*, the SAX-transformed sequence has *w*/(*n**8) (<< 1.0) times the size of original data, where *n* is the length of original data
  - SAX helps improve the performance of our algorithm
    - For computing *MINDIST*() between two SAX-transformed sequences of length *w*, we need only *w* (< *n*) arithmetic operations

Variable Selection Method

- Variable selection
  - Selecting the minimal number of variables that assure accurate results of our fault detection algorithm
  - By using smaller number of variables, we can achieve higher performance of our algorithm
  - Our variable selection method is based on *Dempster-Shafer Theory (DST)*, which is a mathematical theory of probability
  - DST has been used for various applications of real-time malfunction diagnosis

- DST compared with traditional probability theory
  - DST calculates probabilities based on 'evidences'
    - E.g., when a coin is tossed, the probability (support) of having a head up is 0, if there is no evidence
  - The probability of a proposition A in DST is represented with two measures *support* $s(A)$ and *plausibility* $pl(A)$
    - $0.0 \leq s(A) \leq pl(A) \leq 1.0$
    - $pl(A) = 1 - s(A')$
  - DST provides a *rule of combination* for combining probability measures (evidences) from multiple 'independent' sources
    - E.g., a semiconductor manufacturing process where two sensors generate fault alert independently with their own probabilities

- Outline of our variable selection method
  - Computes a goodness measure for each variable in an experimental run
    - Probability (support) that the variable correctly contributes for detecting faults in a certain experimental run
    - Calculated for each of experimental runs independently
  - Joint goodness measure for each variable is calculated using DST's rule of combination
  - Variables with the highest joint goodness measures are selected for our fault detection algorithm

- Goodness measure $g(v_i)$ for a variable v_i
  - $$g(v_i) = \begin{cases} 1 - \max\{F(\hat{R}_D)\} & \text{if } R \text{ is a normal run} \\ \max\{F(\hat{R}_D)\} & \text{if } R \text{ is a perturbed run} \end{cases}$$
  - $\max\{F(\hat{R}_D)\}$ is the maximum $F(\hat{R}_D)$ across all the steps
- Support $s(v_i)$
  - $s\langle v_1, \dots, v_N, \theta \rangle = \left\langle \frac{1}{N} g(v_1), \dots, \frac{1}{N} g(v_N), 1 - \frac{1}{N} \sum g(v_i) \right\rangle$
  - N is the number of variables, θ indicates 'any' variable
  - Calculated for each of experimental runs

- Combination of support values
  - Using DST's rule of combination

$$s_{joint}(v_i) = \frac{1}{k} \sum_{v_A \cap v_B = v_i} s_1(v_A) s_2(v_B)$$

$$k = \sum_{v_A \cap v_B = \varnothing} s_1(v_A) s_2(v_B)$$

  - s_1 and s_2 are support values calculated in any two different experimental runs
  - DST's rule of combination is commutative and associative; hence the joint goodness value can be calculated in any order of runs

Clustering Algorithm

- Our clustering algorithm
  - It forms clusters of experimental runs using the result of our fault detection algorithm
    - It uses the fault steps of experimental runs, i.e., the experimental runs with the same fault steps are gathered
  - Even in case we do not know the source of faults in a certain experimental run, we can estimate it by investigating the experimental runs in the same cluster

- Representation of runs
  - A bitmap $B = b_1 b_2 \ldots b_S$ is used to represent the fault steps for each experimental run (S = the number of steps)
  - A bit b_i is set to 1, if a fault has occurred in the corresponding step; the bit is reset to 0, otherwise.

- Clustering procedure
  - Initially, for each experimental run R_i, a cluster C_i containing the R_i only is created
  - Our algorithm merges the clusters containing the two experimental runs R_i and R_j ($i \neq j$) , if it holds:
    - $Onebit(B_i \oplus B_j) \leq \varepsilon$
  - *Onebit*() function returns the number of 1 bits in a bitmap, the sign $\oplus$ represents XOR operator, and ε is a pre-specified parameter

Evaluation – settings

- Experiment data
  - Real-world semiconductor etching process data
  - 2 run groups
    - model run group: 10 normal runs
    - experimental run group: 3 normal and 7 perturbed runs
  - Each run consists of 11 steps, and real-time stream data of 55 variables were collected at 10Hz

Experiment data *cont'd*

Baseline runs	Experiment runs	
Run#	Run#	Description
FDA_12	FDA_14	Unperturbed control run
FDA_16	FDA_15	–0.5mT change to base pressure
FDA_19	FDA_17	+0.5mT change to base pressure
FDA_21	FDA_20	–1% MFC conversion shift
FDA_24	FDA_23	+1% MFC conversion shift
FDA_28	FDA_25	Source RF cable: loss simulation
FDA_32	FDA_31	Unperturbed control run
FDA_37	FDA_34	Bias RF cable: power delivered
FDA_39	FDA_38	Unperturbed control run
FDA_44	FDA_43	Added chamber leak rate by 1.3mT/min

Evaluation – result

- First experiment
  - We used the 11 variables selected by principal component analysis (PCA) in [7]
  - Our algorithm caused false positive on FDA_20 and FDA_23 and false negative on FDA_31

Table 3 $\max\{F(\hat{R}_D)\}$ values obtained in the first experiment.

	1	2	3	4	5	6	7	8	9	10	11	Decision
FDA_14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	normal
FDA_15	0.000	1.000	1.000	0.927	0.000	0.000	0.000	0.000	0.000	0.000	0.000	perturbed
FDA_17	0.000	1.000	1.000	0.000	0.000	0.000	0.986	0.000	0.147	0.000	0.000	perturbed
FDA_20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	normal (f.p)
FDA_23	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	normal (f.p)
FDA_25	0.000	0.000	0.694	0.537	1.000	1.000	0.832	0.832	1.000	1.000	0.000	perturbed
FDA_31	0.000	0.336	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	perturbed (f.n)
FDA_34	0.000	0.006	0.990	0.992	0.827	1.000	1.000	1.000	0.169	1.000	0.000	perturbed
FDA_38	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	normal
FDA_43	0.000	0.000	0.000	0.000	0.077	0.000	0.249	0.000	1.000	0.000	0.000	perturbed

- Second experiment
  - We perform K-fold cross validation (K = 10), and experimental runs are also used to select variables
  - For each experimental run R ($R \in \mathcal{E}$ = {FDA_14, FDA_15, … , FDA_43}), variables are selected from the model runs and the remaining experimental runs $\mathcal{E} - \{R\}$

Table 4 $\max\{F(\hat{R}_D)\}$ values obtained in the second experiment.

	1	2	3	4	5	6	7	8	9	10	11	Decision
FDA_14	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	normal
FDA_15	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	perturbed
FDA_17	0.000	1.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	perturbed
FDA_20	0.000	0.855	0.913	0.992	0.998	0.991	0.000	0.905	0.978	0.992	0.000	perturbed
FDA_23	0.000	0.759	0.571	0.957	0.983	0.956	0.000	0.559	0.870	0.981	0.000	perturbed
FDA_25	0.000	0.000	0.997	0.904	0.815	1.000	0.997	0.997	0.985	1.000	0.000	perturbed
FDA_31	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.105	0.000	0.000	0.000	normal
FDA_34	0.000	0.125	1.000	1.000	0.504	1.000	1.000	1.000	0.855	1.000	0.000	perturbed
FDA_43	0.000	0.000	0.000	0.000	0.000	0.009	0.213	0.213	0.995	0.024	0.000	perturbed

We achieved 100% accuracy without any false positive or false negative!

- Third experiment
  - We perform our clustering algorithm with $\varepsilon = 0$ (toughest)
  - Experimental runs in the same cluster have similar sources of faults
  - Our algorithm can be used in investigating the source of any anomaly in semiconductor manufacturing processes

Clusters	Experimental runs	Fault Classification
Cluster 1	FDA_14&31&38	Normal runs
Cluster 2	FDA_15&17	Pressure control system
Cluster 3	FDA_20&23	Gas delivery system
Cluster 4	FDA_25&34	RF power system
Cluster 5	FDA_43	Process chamber leak

Conclusions

- Proposed algorithms
  - Fault detection algorithm, which is a modification of the discord detection algorithm called HOT SAX [9]
  - A method to select minimal number of variables assuring accurate results of our fault detection algorithm based on DST
  - An algorithm for clustering experimental runs using the result of our fault detection algorithm
- Evaluation of our algorithms
  - Our fault detection algorithm accurately distinguished the normal and the perturbed runs incurring no false positive or false negative
  - Our clustering algorithm generated good clusters of experimental runs having similar sources of faults

References

1. J. R. Boston, "A signal detection system based on Dempster-Shafer theory and comparison to fuzzy detection," *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 30, No. 1, pp. 45-51, Feb. 2000.
2. K. Chan and A. W. Fu, "Efficient Time Series Matching by Wavelets," In *Proc. of the IEEE Int'l Conf. on Data Engineering (ICDE)*, Sydney, Australia, pp. 126-133, Mar. 1999.
3. B. Chiu, E. Keogh, and S. Lonardi, "Probabilistic Discovery of Time Series Motifs," In *Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Washington DC, USA, pp. 493-498, Aug. 2003.
4. H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures," In *Proc. of the VLDB Endowment (PVLDB)*, pp. 1542-1552, Aug. 2008.
5. C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases," In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, Minneapolis, MN, USA, pp.419-429, May 1994.
6. P. Geurts, "Pattern Extraction for Time Series Classification," In *Proc. of the European Conf. on Principles of Data Mining and Knowledge Discovery*, Freiburg, Germany, pp. 115-127, Sep. 2001.
7. S. J. Hong, G. S. May, J. Yamartino, and A. Skumanich, "Automated Fault Detection and Classification of Etch Systems Using Modular Neural Networks," In *Proc. of the SPIE*, Vol. 5378, Santa Clara, CA, USA, pp. 134-141, Feb. 2004.
8. E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," In *Proc. of ACM SIGMOD Conf. on Management of Data*, Santa Barbara, CA, USA, pp. 151-162, May 2001.
9. E. Keogh, J. Lin, and A. Fu, "HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence," In *Proc. of the IEEE Int'l Conf. on Data Mining (ICDM)*, Houston, TX, USA, pp. 226-233, Nov. 2005.

10. E. Keogh, S. Lonardi, and C. Ratanamahatana, "Towards Parameter-Free Data Mining," In *Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Seattle, WA, USA, pp. 206-215, Aug. 2004.
11. J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A Symbolic Representation of Time Series, with Implications for Streaming Algorithms," In *Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, San Diego, CA, USA, pp. 2-11, Jun. 2003.
12. J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: a Novel Symbolic Representation of Time Series," *Data Mining and Knowledge Discovery (DMKD)*, Vol. 15, No. 2, pp. 107-144, Aug. 2007.
13. G. S. May and C. J. Spanos, "Automated malfunction diagnosis of semiconductor fabrication equipment: a plasma etch application," *IEEE Trans. Semiconductor Manufacturing*, Vol. 6, No. 1, pp. 28-40, Feb. 1993.
14. P. Patel, E. Keogh, J. Lin, and S. Lonardi, "Mining Motifs in Massive Time Series Databases," In *Proc. of the IEEE Int'l Conf. on Data Mining (ICDM)*, Maebashi City, Japan, pp. 370-377, Dec. 2002.
15. K. Sentz and S. Ferson, *Combination of Evidence in Dempster-Shafer Theory, Sandia National Laboratories*, SAND 2002-0835, Apr. 2002.
16. J. Shieh and E. Keogh, "iSAX: Indexing and Mining Terabyte Sized Time Series," In *Proc. of the ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, Las Vegas, NV, USA, pp. 623-632, Aug. 2008.
17. L. Wei, N. Kumar, V. N. Lolla, E. Keogh, S. Lonardi, and C. A. Ratanamahatana, "Assumption-Free Anomaly Detection in Time Series," In *Proc. of the Int'l Scientific and Statistical Database Management Conf. (SSDBM)*, Santa Barbara, CA, USA, pp. 237-240, Jun. 2005.
18. H. Wu, M. Siegel, R. Stiefelhagen, and J. Yang, "Sensor Fusion Using Dempster-Shafer Theory," In *Proc. of IEEE Instrumentation and Measurement Technology Conference*, Anchorage, AK, USA, pp. 7-12, May 2002.


Q&A