Identifying Patterns and Anomalies in Delayed Neutron Monitor Data of Nuclear Power Plant

Durga Toshniwal, Aditya Gupta
Department of Computer Science & Engineering
Indian Institute of Technology Roorkee
Roorkee, India
{durgatoshniwal, adityag} @gmail.com

Pramod K. Gupta, Vikas Khurana, Pushp Upadhyay
C&I and R&D-ES
Nuclear Power Corporation of India Ltd. Mumbai,
{pkgupta, vkhurana, pupadhyay} @npcil.co.in

Abstract— ***In nuclear fission, a delayed neutron is a neutron emitted by one of the fission products any time from a few milliseconds to a few minutes after the fission event. The counts of delayed neutrons constitute a time series sequence. The analysis of such time series can prove to be very significant for purpose of predictive maintenance in nuclear power plants. In this paper we aim to identify anomalies in neutron counts, which may be generated due to possible leaks in the nuclear reactor channel. Real world case data comprising of readings from Delayed Neutron Monitors (DNM) has been analyzed. The time sequences formed by the delayed neutrons have first been symbolically represented using Symbolic Approximation Algorithm (SAX), then anomaly detection and pattern detection algorithms have been applied on them.***

Keywords-Time Series; Anomaly Detection; Symbollic Approximate Algorithm; patterns; dataset; Delayed Neutron Monitor

I. INTRODUCTION

In nuclear engineering, a delayed neutron is a neutron emitted after a nuclear fission event, by one of the fission products (or actually, a fission product daughter after beta decay), any time from a few milliseconds to a few minutes after the fission event. Neutrons born within 10^{-14} seconds of the fission are termed "prompt neutrons."

If a nuclear reactor happened to be in critical state for prompt neutrons, the number of neutrons would increase exponentially at a high rate, and very quickly the reactor would become uncontrollable by means of cybernetics

Delayed neutrons play an important role in nuclear reactor control and safety analysis [16]. The Nuclear reactors operate in subcritical state as far as only prompt neutrons are concerned. The delayed neutrons come a moment later, just in time to sustain the chain reaction when it is going to die out. In that regime, neutron production overall still grows exponentially, but on a time scale that is governed by the delayed neutron production, which is slow enough to be controlled (just as an otherwise unstable bicycle can be balanced because human reflexes are quick enough on the time scale of its instability). Thus, by widening the margins of non-operation and super criticality and allowing more time to regulate the reactor, the delayed neutrons are essential to inherent reactor safety and even in reactors requiring active control [16].

In nuclear reactors, the fuel is encased in metal rods which are mounted in groups in fuel assemblies which in turn are massed together to form the reactor core. Reactor coolant in the form of a fluid passed through the core to absorb heat generated by nuclear reactions in the fuel is typically circulated through several external heat transfer loops. The reactor coolant may be ordinary water, heavy water, a gas or any other material like liquid sodium [16].

In any case, the cladding on the fuel rods is subjected to high temperatures and internal stresses generated as a result of the nuclear reactions in the fuel which can lead to failures in the cladding. Such breaches in the fuel rod cladding introduce fuel into the reactor coolant which carries the contamination throughout the heat transfer loops. Identifying and locating the breached fuel rod in a timely manner is important in order that appropriate action might be taken prior to the time that operational or safety problems are created by the failure [16].

The counts of delayed neutrons (obtained from Delayed Neutron Monitors in nuclear reactors) constitute a time series. Time series is a sequence of data points, measured typically at successive points in time spaced at uniform time intervals. Time series data have a natural temporal ordering. This makes time series analysis distinct from other common data analysis problems, in which there is no natural ordering of the observations.

In this paper, we aim to identify anomalies in neutron counts generated due to possible leaks in the nuclear reactor channel or other reasons. Such situations are very critical and need continues monitoring and attention. The motivation of this study is to predict such conditions to avoid failures in the reactor and other unwanted events.

Two real world datasets have been used for the present study. The dataset's comprises of readings taken from Delayed Neutron Monitor (DNM) over a period of five years (2005-2010).

In order to find the anomalies and patterns in time series, we first convert our dataset into a symbolic representation. We have used symbolic aggregate approximation (SAX) algorithm for this purpose [1]. Then we apply anomaly detection and pattern finding algorithm on the symbolic representation

The remainder of the paper is organized as follows. Section 2 describes the related work. It includes description of SAX algorithm and anomaly detection algorithm. Following section, section 3 details the framework of our project. Section 4 describes the real world dataset that we have used. Next, section 5 describes the experimental results and discussions. Final conclusion is stated in section 6.

II. RELATED WORK

As with most problems in computer science, the suitable choice of representation of time series greatly affects the ease and efficiency of time series data mining. With this in mind, a great number of time series representations have been introduced, including the Discrete Fourier Transform (DFT) [8], the Discrete Wavelet Transform (DWT) [9], Piecewise Linear, and Piecewise Constant models (PAA) [11], (APCA) [12, 11], and Singular Value Decomposition (SVD) [11].

All the above methods are similar in terms of indexing power [13]; however, the representations have other features that may act as strengths or weaknesses. As a simple example, wavelets have the useful multi resolution property, but are only defined for time series that are an integer power of two in length [9].

One important feature of all the above representations is that they are real valued. This limits the algorithms, data structures and definitions available for them. For example, in anomaly detection we cannot meaningfully define the probability of observing any particular set of wavelet coefficients, since the probability of observing any real number is zero [14]. Such limitations have lead researchers to consider using a symbolic representation of time series. One main disadvantage is none of the above techniques allows a distance measure those lower bounds a distance measure defined on the original time series. For this reason, the various generic time series data mining approaches are of little utility.

We have used Symbolic Approximation Algorithm to represent out dataset in symbolic form. The main advantage of this algorithm is that it allows the lower bounding of the true distance. SAX also allows dimensionality/ numerosity reduction, and distance measures to be defined on the symbolic approach that lower bound corresponding distance measures defined on the original series. Now we can take advantage of the generic time series data mining model, and of a host of other algorithms, definitions and data structures which are only defined for discrete data, including hashing, Markov models, and suffix trees. The SAX algorithm is discussed in detail in section 2.

For anomaly detection in most real valued time series problems such as motif discovery [15], longest common subsequence matching, sequence averaging, segmentation, indexing [13], etc. have approximate or exact analogues in the discrete world, and have been addressed by the text processing or bioinformatics communities. For identifying time series anomalies in discrete datasets, Heuristically Ordered Time series is the best algorithm. The algorithm is discussed in section 2.

A. *Symbollic Aggregate Approximation (SAX)*

Symbolic Aggregate Approximation (SAX) algorithm [1] produces symbolic representation of time series. This representation is unique because it allows dimensionality/numerosity reduction, and it also allows distance measures to be defined on the symbolic approach that lower bound corresponding distance measures defined on the original series.

SAX allows time series of arbitrary length n to be converted into strings of length w such that $w<=n$. The alphabet size is also an integer a such that $a >=2$. Converting time series data into SAX representation is a two-step process. We first transform the data into the Piecewise Aggregate Approximation (PAA) [1] representation and then symbolize the PAA representation into a discrete string. There are two important advantages to doing this:

1. Dimensionality Reduction: We can use the well-defined and well-documented dimensionality reduction power of PAA [4, 5], and the reduction is automatically carried over to the symbolic representation.
2. Lower Bounding: Proving that a distance measure between two symbolic strings lower bounds the true distance between the original time series is non-trivial. The key observation that allows us to prove lower bounds is to concentrate on proving that the symbolic distance measure lower bounds the PAA distance measure. Then we can prove the desired result by transitivity by simply pointing to the existing proofs for the PAA representation itself [5].

So, to reduce the time series from n dimensions to w dimensions, the data is divided into w equal sized "frames." The mean value of the data falling within a frame is calculated and a vector of these values becomes the data-reduced representation. This representation is the PAA representation of the time series. Also we normalize each time series to have a mean of zero and a standard deviation of one before converting it to the PAA representation, since it is well understood that it is meaningless to compare time series with different offsets and amplitudes [6, 10].

Having transformed a time series database into PAA, we can apply a further transformation to obtain a discrete representation. It is desirable to have a discretization technique that will produce symbols with equal-probability. This is easily achieved since normalized time series have a Gaussian distribution [7]. Given that the normalized time series have highly Gaussian distribution, we can simply determine the "breakpoints" that will produce a equal-sized areas under Gaussian curve [7]. These breakpoints may be determined by looking them up in a statistical table. For example, Table 1 gives the breakpoints for values of a from 3 to 10.

Once the breakpoints have been obtained we can discretize a time series in the following manner. We first obtain a PAA of the time series. All PAA coefficients that are below the smallest breakpoint are mapped to the symbol "a," all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol "b," etc.

TABLE 1: A LOOKUP TABLE THAT CONTAINS THE BREAKPOINTS THAT DEVIDE A GAUSSIAN DISTRIBUTION IN A NUMBER (3 TO 10) OF EQUIPROBABLE REGIONS

β_i \ a	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Figure 2 illustrates the three steps of SAX generation algorithm. 'C' is the name of the time series. First we obtain the PAA representation of C, which is represented by C-bar. Now we select alphabet size 3. So we introduce two breakpoints. The PAA points lying below the first breakpoint are labeled as 'a', the PAA points lying between the first and the second breakpoint are labeled 'b' and the points lying beyond the third breakpoint line are labeled 'c'.

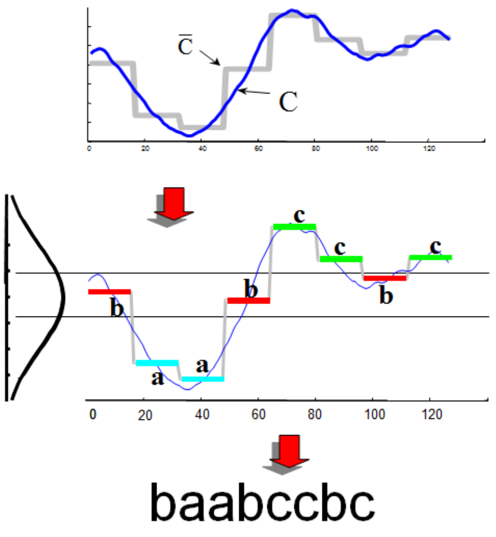


Figure 2: A time series is discretized by first obtaining a PAA approximation and then using predetermined breakpoints to map the PAA coefficients into SAX symbols. In the example above, with n = 128, w = 8 and a = 3, the time series is mapped to the word baabccbc

Now we have to define the distance measure on SAX representation. i.e. how do we calculate the distance between two SAX strings. The distance between two SAX strings can be calculated by Equation 1:

$$MINDIST(\hat{Q},\hat{C}) \equiv \sqrt{\tfrac{n}{w}}\sqrt{\sum_{i=1}^{w}\left(dist(\hat{q}_i,\hat{c}_i)\right)^2} \quad (1)$$

Where *dist()* function calculates the distance between two SAX coefficients. The dist() function can be implemented using a table lookup as illustrated in Table 3.

TABLE 3. LOOKUP TABLE USED BY MINDIST FUNCTION. THIS TABLE IS FOR AN ALPHABET OF CARDINALITY 4.

	a	**b**	**c**	**d**
a	0	0	0.67	1.34
b	0	0	0	0.67
c	0.67	0	0	0
d	1.34	0.67	0	0

The value in cell (r,c) for any lookup table can be calculated by the following Equation 2.

$$cell_{r,c} = \begin{cases} 0, & if\ |r-c| \le 1 \\ \beta_{\max(r,c)-1} - \beta_{\min(r,c)}, & otherwise \end{cases} \quad (2)$$

The question still remains, what values of *w* and *a* should we choose? There is a clear tradeoff between the parameter *w* controlling the number of approximating elements, and the value *a* controlling the granularity of each approximating element.

We choose the value of *a* and *w* such that the following ratio in equation 3 is maximized (close to 1).

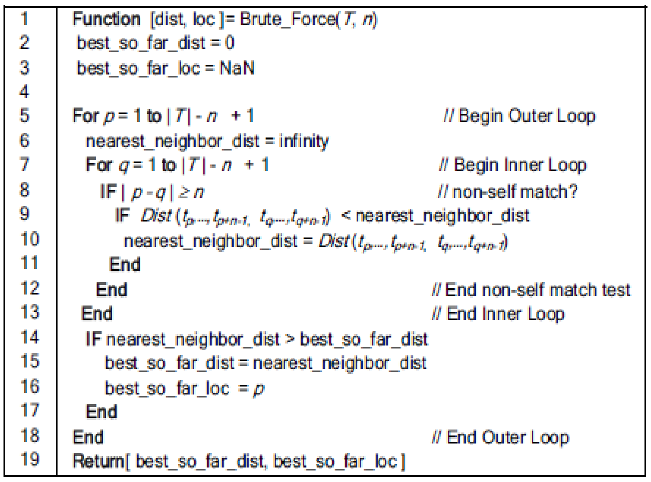
$$Tightness\ of\ Lower\ Bound = \frac{MINDIST(\hat{Q},\hat{C})}{D(Q,C)} \quad (3)$$

So in order to choose the value of a and w, we conduct the following experiment. We find the tightness of lower bound for the time series by calculating the above ratio for every possible combination of substring possible and then averaging the ratio. The result of this experiment are shown in section 5.

B. *Algorithm for detecting anomalies*

Time series anomalies are subsequences of longer time series that are maximally different to all the rest of the time series subsequences. They thus capture the sense of the most unusual subsequence within a time series. Before discussing the algorithm, we must first discuss what are *non-self-match*. Given a time series T, containing a subsequence C of length n beginning at position p and a matching subsequence M beginning at q, we say that M is a non-self match to C at distance of Dist(M,C) if | p – q| >=n. [2]

The brute force algorithm for finding anomalies is simple and obvious. We simply take each possible subsequence and find the distance to the nearest non-self match. The subsequence that has the greatest such value is the discord. This is achieved with nested loops, where the outer loop considers each possible candidate subsequence, and the inner loop is a linear scan to identify the candidate's nearest non-self match. The pseudo code for algorithm is shown in Figure 3.



```
Function [dist, loc ]= Brute_Force( $T$, $n$)
 best_so_far_dist = 0
 best_so_far_loc = NaN

For $p$ = 1 to $|T| - n + 1$                                  // Begin Outer Loop
  nearest_neighbor_dist = infinity
  For $q$ = 1 to $|T| - n + 1$                                // Begin Inner Loop
    IF $|p - q| \geq n$                                       // non-self match?
      IF $Dist(t_p,...,t_{p+n-1}, t_q,...,t_{q+n-1})$ < nearest_neighbor_dist
        nearest_neighbor_dist = $Dist(t_p,...,t_{p+n-1}, t_q,...,t_{q+n-1})$
      End
    End                                                       // End non-self match test
  End                                                         // End Inner Loop
  IF nearest_neighbor_dist > best_so_far_dist
     best_so_far_dist = nearest_neighbor_dist
     best_so_far_loc = $p$
  End
End                                                           // End Outer Loop
Return[ best_so_far_dist, best_so_far_loc ]
```

Figure 3: Algorithm for identifying discords in time series

The advantage of this algorithm is that it requires only one parameter, that is the length of the subsequence as input and it finds the anomaly. The algorithm has square complexity. In order to improve the running time of the algorithm, we implement the following optimization: We don't really need to find the true nearest neighbor for every candidate. As soon as for any candidate, we find that its 'nearest neighbor distance' is less than 'best so far' we abandon the instance of the inner loop, safe in the knowledge that current candidate cannot be the time series discord. The algorithm in figure 3 allows several potential weaknesses for the sake of simplicity. First, it assumes a single anomaly in the dataset. Second, in the first few iterations, the measure needs to note the difference a small anomaly makes, even when masked by a large amount of surrounding normal data. A simple solution to these problems is to set a parameter W, for number of windows. We can divide the input sequence into W contiguous sections, and identify anomaly for each sensor in each of the windows. [3]

III. Proposed Framework

In the proposed framework (Figure 1), symbolic aggregate approximation algorithm is applied on the raw dataset. This helps in discretizing the dataset, and allows us to use various algorithms used in text data mining.

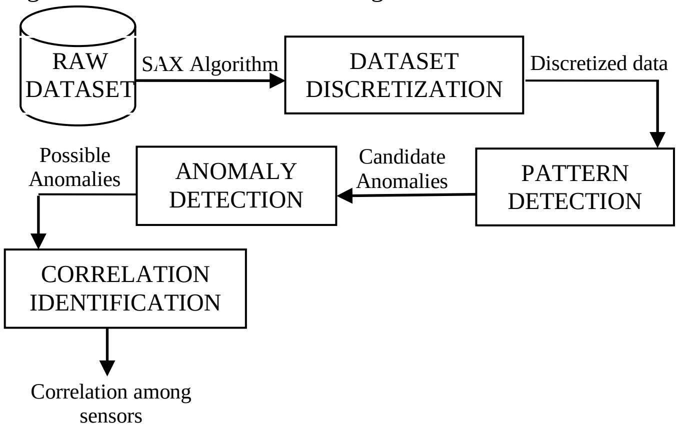


Figure 1. Framework for our experiment

After the dataset has been discretized, we apply the Heuristically Ordered Time series algorithm to find the anomalies in the dataset. We analyze the anomalies found and find the correlation among sensors, that is the probability of one sensor failing given that another sensor has failed.

IV. Dataset

There are two datasets that are analyzed. The first dataset (DP1) consists of readings for 5 years from 2006- 2010. The second dataset (DP4) consists of readings for 6 years 2006 – 2010. Each of these datasets contains readings recorded from 28 sensors on certain days.

Table 2 shows the number of days when readings are recorded in each of the dataset during the period of 2005-2010. For each of these days, a set of 14 readings have been considered for each of the 28 sensors deployed in the nuclear reactor. So in DP1 dataset, there are 434*14*28=1,70,128 (Days multiplied by number of readings each day) readings and in DP4 dataset there are 479*14*28=1,87,768 readings.

TABLE 2: NUMBER OF READINGS FOR EACH YEAR AND EACH DATASET

	2005	2006	2007	2008	2009	2010
DP1	-	19	107	126	96	86
DP4	27	76	106	77	87	106

It has also been assumed that the reactor channel is circular in shape, and the neutron count towards the center of the channel is greater when compared to the neutron count towards the circumference.

V. Experimental Results & Discussion

A. *Finding Optimal SAX Representation*

In SAX representation of a dataset, the most important point to consider is what should be the value of word size (*w*) and alphabet size (*a*). *w* is the size of SAX string, i.e. the time series string of length *n* is converted into SAX representation of length *w*. *w* is less than or equal to the length of original time series *n*. Very small values of *w* are not preferred as it leads to loss of accuracy. Also very large values of *w* are also avoided as then there is no reduction in the size of the dataset. [1]

Alphabet size *a* controls the granularity of each approximating element. So an alphabet size of 3 means that each approximating element can take 3 values i.e. 'a', 'b', 'c'.

One of the most important properties of SAX representation is that it lower bounds the distance of two symbolic representations when compared to the distance between the original series. Lower bounding means if *A* and *B* are original time series and distance between them is *X* ; *Q* and *R* are their symbolic approximations and distance between them is *Y* then *Y* lower bounds *X*. i.e. *Y*<=*X* always! A lower bounding symbolic approach would allow us to use suffix trees, hashing, Markov models, text processing and bioinformatics algorithms on symbolic approximation.[1] The closer is *Y* to *X* more accurate is our approximation.

Hence we wish to choose variables a and w such that there is tightest possible lower bound between the symbolic approximation and time series. Equation 3 shows the equation for tightness of lower bound.

In this equation D and C are two time series subsections and D(Q,C) is the distance between these subsections. MINDIST(Q,C) finds the difference between the SAX approximations of Q and C. Hence we can see that the above equation will always be less than one, since MINDIST lower bounds D(Q,C) function.

In our experiment, we choose different values of *a* and *w* and for all possible combinations of time series subsequences find their *Tightness of Lower Bound*. Finally we take the mean for all the lower bounds to represent the property for a given *a* and *w*.

We performed such tests on DP1 dataset for year 2006. In all we found mean for lower bound for 171 subsequences of the data set. We averaged these results to find the final results. We conducted these experiments for all the sensors. Figure 4, Figure 5 and Figure 6 shows the results. In these results the tightness of lower bound is shown on z axis whereas x and y axis contain alphabet size and word size respectively.

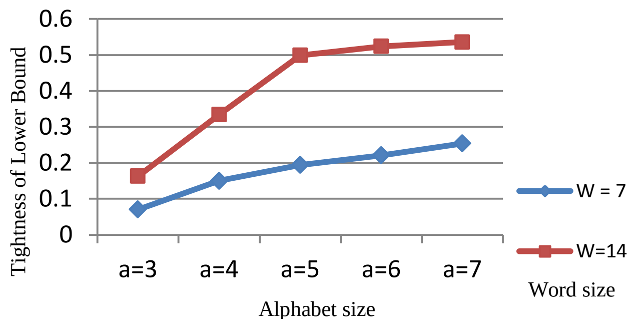


Figure 4: Tightness of Lower Bound for different values of alphabet sizes (a) and for word sizes of 7 and 14, when calculated for time series generated by SENSOR 1

The results suggest that using a low value for *a* results in weak bounds, but that there are diminishing returns for large values of *a*. The results also suggest that the parameters are not too critical; an alphabet size in the range of 5 to 7 seems to be a good choice.

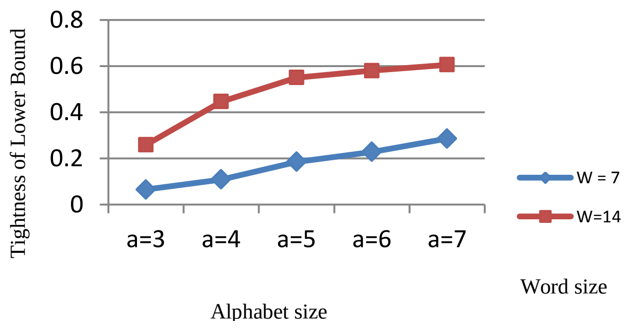


Figure 5 Tightness of Lower Bound for different values of alphabet sizes (a) and for word sizes of 7 and 14, when calculated for time series generated by SENSOR 26

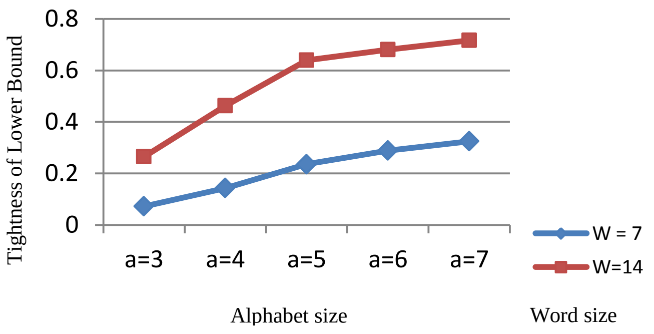


Figure 6 Tightness of Lower Bound for different values of alphabet sizes (a) and for word sizes of 7 and 14, when calculated for time series generated by SENSOR 12.

Based on these results we have chosen word size of 14 and alphabet size of 5 to represent the time series dataset by symbolic representation. Using the SAX algorithm we convert the entire dataset into symbolic representation.

B. *Finding Anomalies in the Dataset*

Once we have converted the time series dataset, we apply the algorithm discussed in section 2 to find the discords. It must be noted that this algorithm takes only the length of the anomaly as the input and identifies the subsequence of that length that is most different from other subsequences. We have performed our experiment for all anomaly sizes varying it from 3 to 14. We have found that strongest anomalies are detected for anomaly size of 11.

This algorithm has two potential weaknesses that we must solve. First, it assumes a single anomaly in the dataset. Second, in the first few iteration, the measure needs to note the difference a small anomaly makes, even when masked by a large amount of surrounding normal data. A simple solution to these problems is to set a parameter W, for number of windows.

We can divide the input sequence into W contiguous sections and apply our algorithm on each of these windows. In our experiment, we have taken the window size to be 17, hence we are finding the most anomalous subsequence of length 11 for each of the sensors in data taken across 17 days. It must be noted that, now in a time series there are 14 readings for each day and in all there are 17 days, so for each sensor we have 238 readings and we are trying to find the subsequence of length 11 that is most different from the others.

So we consider readings from each sensor to be part of a time series. We divide readings for each sensor in group of 17 days and apply the algorithm shown in figure 3. So for each sensor, we find the day when the sensor has been most anomalous (with respect to other 16 days in the window).

We repeat the above process for each window of each sensor. We get large number of results for our experiment, a snapshot of part of the results is shown below in table 4.

In all there are two datasets having data for a number of years. So we perform our experiment on the entire datasets.

TABLE 4: RESULT OF ANOMALY DETECTION ALGORITHM FOR A WINDOW IN YEAR 2008, DP4 DATASET

S.No.	Sensor No.	Timestamp
1	1, 14, 20	06/06 /2008
2	2,7, 12	05/05 /2008
3	3,11	02/06 /2008
4	4	14/05 /2008
5	5,24	21/05 /2008
6	6, 25	09/05 /2008
7	7,8,9,13	11/04 /2008
8	10	16/05 /2008
9	15, 22	28/05 /2008
10	18	07/04 /2008
11	19	26/05 /2008
12	21	18/04 /2008
13	23	4/9/2008

So as it can be seen above, in window of 17 days in year 2008, we have identified days when the sensor has been most anomalous. Also there are sensors that do not show any anomaly at all, for example sensor number 16, 17, 26, 27 and 28 don't show any anomaly!

In the second part of anomaly detection process, we try to identify a single day when each of the sensors has been most anomalous. In order to do this, we compare the most anomalous day in each window of set of 17 days. The results of this process for DP1 dataset for year 2008 are shown below in table 5.

TABLE 5: THE MOST ANOMALOUS DAY FOR EACH SENSOR DURING THE YEAR 2008 IN DP1 DATASET

2008- DP1	Days
Sensor 7	1/9/2008
Sensor 11	2/6/2008
Sensor 12	2/7/2008
Sensor 10	4/8/2008
Sensor 22	4/8/2008
Sensor 26	4/8/2008
Sensor 28	4/8/2008
Sensor 6	4/8/2008
Sensor 23	4/9/2008
Sensor 14	16/6/2008
Sensor 20	16/6/2008
Sensor 1	6/8/2008
Sensor 17	6/8/2008
Sensor 13	6/10/2008
Sensor 18	7/4/2008
Sensor 5	9/7/2008
Sensor 3	10/24/2008
Sensor 8	11/4/2008
Sensor 9	11/4/2008
Sensor 19	11/7/2008
Sensor 24	11/17/2008
Sensor 15	20/7/2008
Sensor 2	20/8/2008
Sensor 25	24/9/2008
Sensor 4	28/7/2008
Sensor 21	30/7/2008
Sensor 16	30/7/2008
Sensor 27	30/7/2008

From the above table, we derive a very useful result. We have derived the most anomalous day for each of the sensor independently. That is we considered data for each sensor as an independent time series, still there are group of days when multiple sensors are showing anomalies simultaneously. We can see that sensor 6, 10, 26, 28 and 22 show maximum anomalies on the same day. Below in table 6, we summarize this result. Hence we can deduce that there must be some correlation among sensors. That is, when one sensor fails, there is certain probability that other sensors with whom it has high correlation also fail. Hence in the next section we explore this and try to find sensors with high correlation.

TABLE 6: SENSORS SHOWING MAXIMUM ANOMALY ON SAME DAY

S.No.	Sensor No.'s	Timestamp of Anomaly
1	10, 22, 26, 28, 6	4/8/2008
2	14,20	16/6/2008
3	1,17	6/8/2008
4	8,9	11/4/2008
5	21,16,27	30/7/2008

C. *Finding Correlation Among Sensors*

When we analyze the discords found, we find some interesting patterns, like some sensors are related to each other. That is they show discords on same days. For these sensors, we find the probability of failure on same day. For example for DP4 dataset, and year 2008, we obtain the following observation as shown in table 7.

TABLE 7. PROBABILITY OF SENSORS FAILING SIMULTANEOUSLY

Sensor No.	Sensor No.	Probability
26	28	0.50
4	23	0.50
6	26	0.50
6	28	0.50
9	13	0.50
21	27	0.50

VI. CONCLUSION

The entire dataset has been discretized using SAX representation. Then anomaly finding algorithm was applied on the datasets. For both the datasets, days were identified when there is an anomaly in the neutron flow counts. These anomalies may be generated due to possible leaks in the nuclear reactor channels or other reasons.

Also correlation among sensors was found based on the result of anomalies. Hence the probability of two sensors showing anomalies simultaneously has been calculated. We have also ranked the sensors based on the mean of their readings and used the basic information given to us about dimensions of the devise to infer the locations of the sensors in the device.

In our future work, we will be applying motif discovery algorithms to identify patterns that repeat themselves in the dataset.

ACKNOWLEDGMENT

We would like to thank Department of Atomic Energy for providing the domain knowledge, dataset and for partially funding the research work and program. Research Project Grant Number DAE-603-ECD.

REFERENCES

[1] Jessica Lin , Eamonn Keogh , Stefano Lonardi , Bill Chiu, A symbolic representation of time series, with implications for streaming algorithms, Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, June 13-13, 2003, San Diego, California.

[2] Eamonn Keogh , Jessica Lin , Ada Fu, HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence, Proceedings of the Fifth IEEE International Conference on Data Mining, p.226-233, November 27-30, 2005 [doi>10.1109/ICDM.2005.79]

[3] Eamonn Keogh , Stefano Lonardi , Chotirat Ann Ratanamahatana, Towards parameter-free data mining, Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, August 22-25, 2004, Seattle, WA, USA [doi>10.1145/1014052.1014077]

[4] Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra, S. (2001). Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In proceedings of ACM SIGMOD Conference on Management of Data. Santa Barbara, CA, May 21-24. pp 151-162.

[5] Yi, B, K., & Faloutsos, C. (2000). Fast Time Sequence Indexing for Arbitrary Lp Norms. In proceedings of the 26st Int'l Conference on Very Large Databases. Sep 10-14, Cairo, Egypt. pp 385-394.

[6] Keogh, E. & Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. In proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. July 23 - 26, 2002. Edmonton, Alberta, Canada. pp 102-111.

[7] Larsen, R. J. & Marx, M. L. (1986). An Introduction to Mathematical Statistics and Its Applications. Prentice Hall, Englewood, Cliffs, N.J. 2nd Edition

[8] Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). Fast Subsequence Matching in Time-Series Databases. *In proceedings of the ACM SIGMOD Int'l Conference on Management of Data*. May 24-27, Minneapolis, MN. pp 419-429.

[9] Chan, K. & Fu, A. W. (1999). Efficient Time Series Matching by Wavelets. In *proceedings of the 15th IEEE Int'l Conference on Data Engineering*. Sydney, Australia, Mar 23-26. pp 126-133.

[10] Geurts, P. (2001). Pattern Extraction for Time Series Classification. In *proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*. Sep 3-7, Freiburg, Germany. pp. 115-127.

[11] Keogh, E., Chakrabarti, K., Pazzani, M. & Mehrotra, S. (2001). Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In *proceedings of ACM SIGMOD Conference on Management of Data*. Santa Barbara, CA, May 21- 24. pp 151-162.

[12] Datar, M. & Muthukrishnan, S. (2002). Estimating Rarity and Similarity over Data Stream Windows. In *proceedings of the 10th European Symposium on Algorithms*. Sep 17-21, Rome, Italy.

[13] Keogh, E. & Kasetty, S. (2002). On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration. *In proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. July 23 - 26, 2002. Edmonton, Alberta, Canada. pp 102-111.

[14] Larsen, R. J. & Marx, M. L. (1986). An Introduction to Mathematical Statistics and Its Applications. Prentice Hall, Englewood, Cliffs, N.J. 2nd Edition.

[15] Chiu, B., Keogh, E. & Lonardi, S. (2003). Probabilistic Discover of Time Series Motifs. In the 9th SIGKDD Conference on Knowledge Discovery and Data Mining. pp 493-498.

[16] Locating a breached fuel assembly in a nuclear reactor on-line. **http://www.google.com/patents/EP0258958A1?cl=en**.