Proceedings of 2013 IEEE
International Conference on Mechatronics and Automation
August 4 - 7, Takamatsu, Japan

Online Detection of Deviation in Performance of Multichannel Dynamical Processes

Gancho Vachkov

School of Engineering and Physics
Faculty of Science, Technology and Environment
The University of the South Pacific (USP)
Laucala Bay, Suva, Fiji

gancho.vachkov@gmail.com; vatchkov_g@usp.ac.fj

***Abstract* – In this paper a computational method for detection of deviation in performance of multiple parallel working channels in a dynamical process is proposed and discussed. The method consists of the following computation steps: one-dimensional multi-agent clustering of the outputs of the channels; similarity analysis of all pairs of channels; calculating the weighted global distance for each channel; static and dynamic ranking of the deviation of all channels.**

The proposed method is intended to work in online mode by continuously processing Data Blocks with fixed length and the respective results are the form of the ranking positions of the channels. The process with Rank 1 is the most deviated process channel. An experimental example is given in the paper for detection and analysis of the deviation in performance of six batteries used for driving a small electric vehicle. Other possible applications for online monitoring and performance evaluation of dynamical processes are also discussed in the paper.

Index Terms – deviation detection, similarity analysis, multi-agent clustering, performance evaluation

I. Introduction and Problem Statement

There are many types of dynamical processes in the industry, such as chemical, metallurgical and machinery plants, where the equipment operates in a parallel way - as several parallel production lines. Then each line can be viewed as one "channel" or "sub-process" of the whole industrial process. Normally all the channels receive the same input load (such as material flow, energy) and work under the same regime parameters (such as temperature, pressure, rotation speed etc.). The main reason for designing such multichannel production structure is the need to increase the production output of the whole plant or to increase the output power. This is the case of the power generation plants or the case of electric battery, consisting of large number of parallel and serially connected battery cells. Another important reason to design a multichannel structure of the dynamical process is to increase the reliability of the whole production process. Then, if the performance of a certain production line (channel, battery cell) deteriorates significantly from that one of the other parallel lines, it can be safely isolated (shut-down) from the whole production process for maintenance, without shutting-down the whole plant.

It is clear that the problem of proper performance evaluation of the simultaneously working parallel lines (channels) is crucial for discovering the trend of abnormal deviation in the performance of some channels. Once a significant deviation is detected, then the subsequent action (such as shutting down the deviated channel and back-up with another reserved channel) could be considered.

The general issue in such *detection of deviation in performance* is that the channels are dynamical processes with dynamically changing (variable) outputs, depending on the respective changes of the load and the other input parameters. Therefore the process of detection of deviation should be also performed in a dynamical way and the results will vary with time.

It seems that the *real-time* detection of deviation could be the best method to apply, where the detection should be performed after each new portion of sampling data from all the channels is available. However such approach is not always possible, because it depends on the complexity of the method for deviation detection, i.e. the computational cost and time that need to be shorter than the pre-specified sampling period. In addition, in most practical cases, such "fast" real-time detection is even not necessary, because the deterioration in performance is usually a dynamical process with a slow gradual trend.

Therefore in this paper we propose a computational method for *Online Detection of Deviation* (ODD) in performance of parallel processes (channels, sub-processes). It is a kind of *quasi-real-time* method, in which the input information from each channel is collected in *online* mode for a relatively short time period, consisting of a pre-defined number of samplings (e.g. 5000 samplings). Each collection of such fixed number of data is further on called *Data Block* (DB). Then the information contained in each data block has to be processed sequentially by the proposed computational method and the results from detection of deviations are usually displayed in an easy-to-understand (numerical or graphical) way to the operator of the process.

The important point here is that the processing of the data from each data block should be completed before the data for the next DB are collected. Then it is clear that the size of the DB (the number of samplings) will depend on the complexity of the computation method for detection of deviation. It is a normal practical case when the size of the DB is within the range of hundreds to thousands of data samplings. Then the respective deviation results will be displayed periodically in short time intervals - usually dozen to hundreds of seconds).

978-1-4673-5558-2/13/$31.00 ©2013 IEEE

This frequency is usually sufficient to capture the slow trend of deterioration (deviation) of the process.

It is clear from the above explanations that the problem of *detection of deviation in performance* belongs to the large group of diagnostics, performance evaluation and anomaly detection problems. In other words this is the problem of finding *one or more* process channels that *deviate significantly* in performance, compared to the performance of the other channels (considered as "normal"). This paper is a continuation of the research in our previous paper [1], with a new – simplified and faster computation procedure, especially the proposed clustering algorithm.

The rest of the paper is organized as follows. In Section II the main computational steps of the proposed method for detection of deviation are given. Section III explains the newly proposed algorithm for One-Dimensional Multi-Agent clustering, based on the weighted average. Section IV gives all the other computational details on the detection of deviation and the final procedure for static and dynamic ranking of the process channels. Section V shows the results from the proposed method for detection of deviation, on the experimental example of a set of six batteries for a small electric vehicle. Finally, the discussions and conclusions are given in Section VII.

II. The Proposed Method for Detection of Deviation

We assume that the performance of each channel (sub-process) can be evaluated by using the information from one representative output of the channel. It will be denoted as a dynamically *changing* variable *x(t)*. For example, in the case of a battery that consists of several battery cells, connected in parallel or in serial, the typical output of each cell (channel) is the output voltage of the cell. We also assume that all channels receive the *same* input load, which is basically a *time varying* parameter. For example, it could be the current in the serially connected battery cells, which will be changing with time, according to the consumer load.

Let us consider a dynamical process (plant) that consists of *L* sub-processes (channels). We propose her an *online* method for detection of deviations in performance of all the channels, which needs a collection of *N* data samplings from each of the channels in the form: $x(t_i)=x_i,\ i=1,2,...N$. According to the explanations in the previous section these data form one Data Block. Then the proposed method for detection of deviations consists of the following major computation steps:

- *One-Dimensional Clustering* of the output of each channel separately. This step is a kind of *information compression* [2,3] of the available data. Here we propose a weighted average based *multi-agent* clustering algorithm. Which is different from the classical clustering algorithms in [4,5]. It is able to automatically find the number of all clusters that represent the areas of data with the largest density from the current Data Block.

- *Similarity Analysis* of all pairs of channels. The aim of this computation step is to *distinguish* the pairs of channels that differ significantly from the other pairs of channels that behave in a closer (similar) way.

- Calculating the *Weighted Global Distance* for each channel. This is an individual characteristic of each channel, which shows how much it is different from the other channels. It allows the channel to be ranked within the group of all *L* channels

- *Static and Dynamic Ranking* of the channels. This is the final computation step for the current Data Block, which presents the list with results from the detection of deviation.

All the above computation steps are explained in details in the sequel of the paper.

III. One-Dimensional Clustering Based on Weighted Average

The objective of the newly proposed clustering method in this paper is to find the locations (one or more) within the whole range of the data samplings in the current DB that have *higher density* distribution of the data, compared with all other locations. These new locations are the *centers* of the clusters. The proposed method here uses the concept of *weighted average* and combines the features of two very popular clustering methods, namely: (1) the iterative computational structure of the classical *C-means* clustering algorithm [4,5] and (2) the flexibility of the group of *sequential* clustering algorithms [6-9] that are able to automatically find the proper number of clusters.

A. The Algorithm for Single Agent One-Dimensional Clustering

In this algorithm the so called *weighting functions W* is defined in the form of a *Gaussian* function with an initial center location $\mathbf{c}=\mathbf{c}_0$ and a predefined *width* (spread) σ. The function *W* can be viewed as a *Single Agent* that *moves* within the whole range of the one-dimensional space of *N* data, $[x_{\min}, x_{\max}]$, in order to find the location (coordinate) that has the biggest density of the data. The algorithm is iterative, which means that at each iteration the center $\mathbf{c}$ will be updated to a new location, until a stopping criterion is satisfied. The main computation steps of the proposed algorithm are:

Step1. Calculate the weighting function *W* in order to obtain the *weights* of all *N* data points from the Data Block that correspond to the current location $\mathbf{c}$ of the function:

$$w_i=\exp\left(-\frac{(c-x_i)^2}{2\sigma^2}\right), i=1,2,...,N \tag{1}$$

Step 2. Calculate the *weighted average Wav* of all *N* data:

$$Wav=\sum_{i=1}^{N} x_i w_i \Big/ \sum_{i=1}^{N} w_i \tag{2}$$

Step 3. Calculate the difference *Del* between the current position (center) $\mathbf{c}$ of the weighting function and the weighted average *Wav*, i.e.

$$Del = c - Wav \tag{3}$$

Step 4. Check for the Stopping condition:

If $|Del| > \varepsilon$ then

- Update the position of the Agent (the weighting function): $c \rightarrow c + Del$
- Go to Step 1 to continue the Iterations.

If $|Del| \le \varepsilon$ then Stop the clustering. The cluster center is determined as the last (most recent) location of the center c of the weighting function. The parameter ε in this step is a small number (assumed as *0.001* in this paper) that defines the accuracy in location of the cluster center c.

It is obvious that the above algorithm will converge to one *local solution* to the problem, namely it will find just one cluster center with high data density within the range of all *N* data. This will be in fact the *nearest* cluster center to the initial (starting) position c_0 of the agent. In the general case of *multiple* cluster centers (several areas with higher data density), the single agent algorithm should be upgraded to the *multi-agent* case, as explained below.

B. *Multi-Agent One-Dimensional Clustering*

Here the objective is the find all areas (cluster centers) with higher data density within the full data range $[x_{\min}, x_{\max}]$. We propose here a simple idea, namely to use $1 < n < N$ agents, each of them performing the above single-agent one-dimensional algorithm, but from *different* initial center locations: $x_{\min} < c_{0i} < x_{\max}$, $i = 1,2,...,n$. A good idea for determining the initial locations of these agents is to place them uniformly within the data range $[x_{\min}, x_{\max}]$.

Then, after a finite number of iterations all the agents will converge to their respective cluster centers: $c_1, c_2, ..., c_n$. Note that in general the number of iterations until convergence will be different for the different agents.

We normally set the number *n* of agents to be greater than the expected (real) number of clusters *k*, i.e. $n > k$. Therefore there will be a *redundancy* of agents, which will definitely lead to situations, where two or more agents have converged to the same (or almost the same) value for a cluster center, for example: $c_i \approx c_j \approx c_m$, where $i, j, m \in \{1,2,...,n\}$. Then we have to define algorithmically the true (real) number *k* of cluster centers, by using a small predefined threshold that represents the notion of "almost the same" position.

The predetermined parameter *width* (spread) σ can be defined as a fixed fraction of the whole range $[x_{\min}, x_{\max}]$ of the data, for example *1/10* to *1/20*. It has the physical meaning of *resolution* of the clusters, i.e. a smaller width allows discovering clusters that are closer to each other.

Once the *true* number *k* of clusters is obtained with their respective cluster centers: $c_1, c_2, ..., c_k$, we need to calculate also the *relative weight* $RW_i, i = 1,2,...,k$ for each cluster. Then these two parameters: c_i and $RW_i, i = 1,2,...,k$ will be used to compare the performance of the different pairs of channels from the dynamical process.

For this purpose, first we need to calculate the *absolute weight AW* of each cluster. This is done by summation of the weights of all *N* data points in relation to this cluster:

$$AW_i = \sum_{j=1}^{N} \exp\left(-\frac{(c_i - x_j)^2}{2\sigma^2} \right), \; i = 1,2,...,k \qquad (4)$$

Then the relative weights of all *k* clusters are calculated as:

$$RW_i = AW_i \Big/ \sum_{j=1}^{k} AW_j, \; i = 1,2,...,k \qquad (5)$$

A test example with *N=574* synthetic data is shown in Fig. 1. These data represent a one-channel Data Block and they have two distinct areas of high density distribution: at *x = 5* and at *x= 10*. This can be easily noticed from the histogram of the data in Fig. 2.

After running the one-dimensional multi-agent clustering algorithm with *n =11* agents, it converged for less than 50 iterations and *k = 2* cluster centers were found, as follows: $c_1 = 5$ and $c_2 = 10$. The results are shown in Fig. 3. The clustering algorithm was executed multiple times with different values of the spread σ within the wide range: [*0.2, 1.5*]. All runs converged to the same two cluster centers, which proved the relative insensitivity of this parameter.

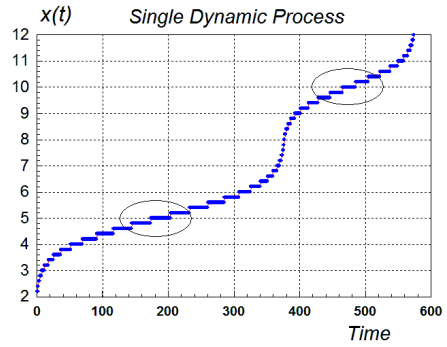

Fig. 1 Test example with *574* synthetic data that represent one data block for one dynamilcal channel.

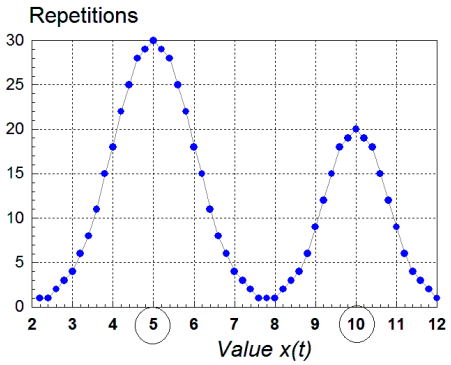

Fig. 2 Histogram of the data from the test exmaple that show the data distribution.

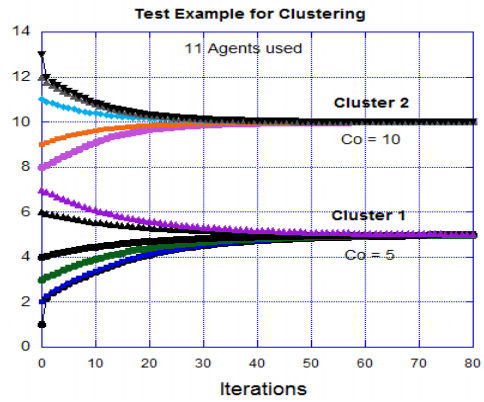

Fig. 3 Result from the Muti-Agent One-Dimansional clustering by using $n = 11$ agents that start from different initial positions. The results is: $k = 2$ different clusters found: $c1 = 5.0$ and $c2 = 10.0$.

The relative weights of the two clusters from Fig. 2, calculated by (5) are: Cluster 1: $RW1 = 0.6071$ and Cluster2: $RW2 = 0.3929$.

IV. Algorithm for Detection of Deviation in Performance

The above one-dimensional multi-agent clustering is performed for all L sub-processes (channels) separately for the given portion of N data from the current Data Block. The results are the *cluster centers* $c_1, c_2, ..., c_k$ and their *relative weights*: $RW_i, i=1,2,...,k$ for all channels. Normally k is a small number of clusters (sometimes just $k=1$), but it is different for each channel. Here we propose the following further computation steps for detection of the deviation in performance of all L channels:

Step 1. Similarity analysis of all pairs of channels.

In this step estiamtion of *how similar* are two channels (for example the channels p and q) from the list of all L channels is made. One simple way to measure the *dissimilarity DS* between the channels p and q could be to calcualte the *mean* of all *pair distances* between the clusters from the different channels:

$$DS_{pq} = \sum_{i=1}^{K_p}\sum_{j=1}^{K_q}\left|c_i - c_j\right| \Big/ (k_p\, k_q) \qquad (6)$$

The idea of calculating the dissimilarity DS between two channels, based on the pair distances between the clusters is shown in Fig. 4.

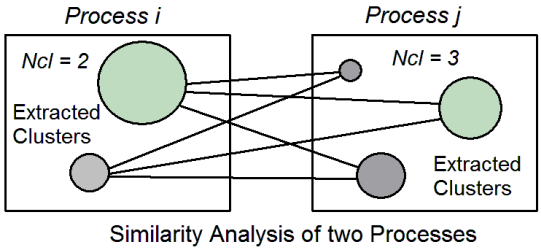

Fig. 4 Calculating the pair distances between the clusters from two channels.

However it is obvious that the simple measure of dissimilarity DS in (6) is not very realistic representation of the dissimilarities between two channels, because it does not take into account the size (the relative weight) of the clusters. Therefore we have included the *relative weights* (4) of each cluster from both channels in calcualting the new dissmilarity measure, called *weighted dissimialrity WDS*. It is calculated foe each pair $\{p,q\};\ p,q \in [1,2,...,L]$, as follows:

$$WDS_{pq} = \sum_{i=1}^{K_p}\sum_{j=1}^{K_q}\left|c_i - c_j\right| \Big/ \sum_{i=1}^{K_p}\sum_{j=1}^{K_q} RW_i\, RW_j \qquad (7)$$

Obviously, a *smaller* value of WDS suggests that the channels p and q are *more similar* (have similar perfomance) and a larger value of WDS represents the case, where these channelsare different in performance.

Step 2. Calculate the Weighted Global Distances WGD for all Channels.

The objective of this step is to estimate "how close" (in performance) is each channel compared with the performances of all the other channels. We have to construct a kind of *measure of distance* for each channel separately. If the distance is small, this means that the channel is "within" the group of all other channels, i.e. it is similar in performance to the other channels. A larger channels.

Here we propose a plausible measure of distance, namely calcualting the *weighted global distance WGD* for a given channel p as the sum of the weighted dissimilarities (7) between the channel p and all the other $L-1$ channels, as follows:

$$WGD(p) = \sum_{\substack{q=1 \\ q\neq p}}^{L} WDS_{pq},\ p=1,2,...,L \qquad (8)$$

The concept of the WGD is depicted graphically in Fig. 5. for the case of one channel ($p=1$).

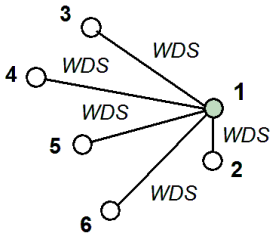

Fig. 5 Graphical representation of the Weighted Global Distance (WGD).

Step 3. Ranking of the channels according to the amount of their deviation in performance.

This is the last computation step. We need to finally *rank* the performance of all the channels and to extract *one or more* channels that *deviate most* from the other channels (that are assumed to be *normal* channels). We propose here two kinds of ranking procedures, namely *instant* (static) ranking and *cumulative* (time-varying, dynamic) ranking, as explained below.

A) *Instant (Static) Ranking*. This is in fact a sorting procedure in *descending* order $r_1, r_2, ..., r_L$ (from *max* to *min*) of the weighted global distances *WGD* of all *L* channels: The channel r_1 has the *maximal WGD* and the channel r_L has the *minimal WGD*. This ranking list can be displayed for appropriated visualization, together with the so called *Contribution CON* that is calculated as the difference between the *WGD* of the current channel and the *WGD* of the last channel (with rank r_L).

$$CON(r_i) = WGD(r_i) - WGD(r_L),\ i = 1, 2, ..., L \qquad (9)$$

B) *Cumulative (Dynamic) Ranking*. This is another way of displaying the results in the form of ranking list that is made by *adding* the individual contributions (9) of all *L* channels: $CON(1), CON(2), ..., CON(L)$ separately. Since this is done for each Data Block in a time sequence, the contributions of all channels will be monotonously increasing functions of time, which can be easily visualized. Thus the *trends* of some channels to deviate *faster* can be easily noticed by the operator for respective actions.

V. Experimental Example for Detection of Deviation and Results

The proposed computational scheme for deviation detection in this paper is illustrated on the example of a dynamical process with *L = 6* channels. The process here is a set of *six serially connected batteries*, each of them with nominal output of *6* Volts (three battery cells in each Battery). This makes a total output of *36* volts. The whole set of *6* batteries is used to drive the DC motor of a small electric vehicle (similar to a golf cart). Each of the six batteries is considered as one channel, so we have a *six-channel* process. The *load* of the process is the *current,* measured in *Ampers* and it has the same value for all *6* channels. In our experiments the load fluctuates from large *positive* values (*charging* the Batteries) to large *negative* values (*discharging* the Batteries through the DC motor).

The six batteries (channels) are not exactly identical, because some of them could be new and in a good condition, while others could be older and in a worse condition. As a result, they react in a different way, with different output voltage (below and above *6* volts).

During the experiments, the common *load* (in *Ampers*), as well the *output* from each battery (in *Volts*) have been measured and recorded with sampling period of *0.1* sec. For experimental purpose, the load has been changed in wide margins, from positive to negative, as shown in Fig. 6. The total length of the experiment was *40000* samplings, i.e. *4000 sec*. The respective changes in the voltage from each battery are shown in Fig. 7. The voltage from all *6* batteries represents the respective outputs of all six channels.

The following preliminary information was available about the current condition of the batteries: Batteries *3* and *5* are relatively older and in a worse condition than the other batteries: *1,2,4* and *6*. This information was used only for confirmation (validation) of the results obtained from our method for detection of deviation.

We have used Data Blocks with two different sizes, namely *5000* samplings (*500 sec*) and *2500* samplings (*250 sec*) for detection of deviation. This has leaded to analyzing the results from a sequence of *8* and *16* blocks respectively.

First, the multi-agent one-dimensional clustering from Section IIIB was run with *n = 10* Agents and a spread $\sigma = 0.1$. It produced small number of clusters: *k = 1,2* or *3* for the different channels. Then all the other steps from the algorithm for *deviation detection* in Section IV have been performed.

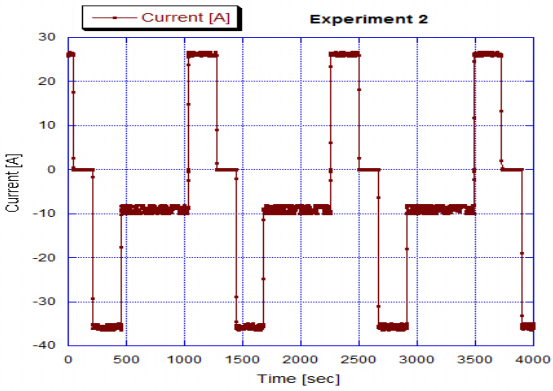

Fig. 6 Changes of the load (current) during the experiments.

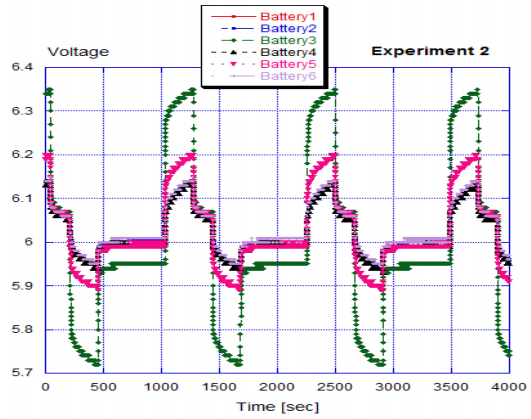

Fig. 7 Respective changes of the six outputs (voltage) from all batteries.

The results from the static and dynamic ranking for all Data Blocks are shown in Fig. 8, 9 and 10. They show that Battery *3* is the distinct "winner" over all the other batteries, in a sense that it deviates significantly from the behavior (performance) of all the other batteries. Another battery, which also shows significant deviation in performance, is Battery *5*, even if it is in some cases close to the behavior of Battery *6*. This is seen from the static ranking in Fig. 8 (they often exchange the *Rank 2* position) and from the dynamic ranking in Fig. 9 and 10. The remaining three batteries, namely: *1, 2* and *4* have *very similar* performance as seen from Fig. 9 and 10. They are considered as "normal batteries".

The results from the detection of deviation match well the preliminary known information about the current condition of the batteries. It was also noticed that the change in the size (length) of the Data Blocks (from *500* to *250* sec) did not change the final ranking results.

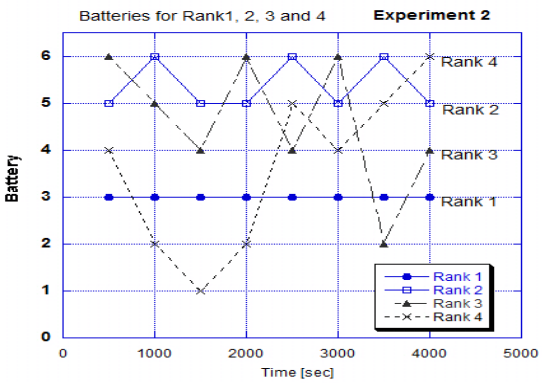

Fig. 8 Results from the Instant (Static) ranking, based on data blocks with length *500 sec*.

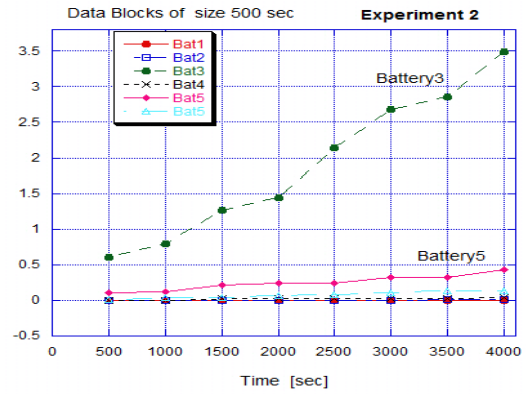

Fig. 9 Results from the Cumulative (Dynamic) ranking, based on data blocks with length *500 sec*.

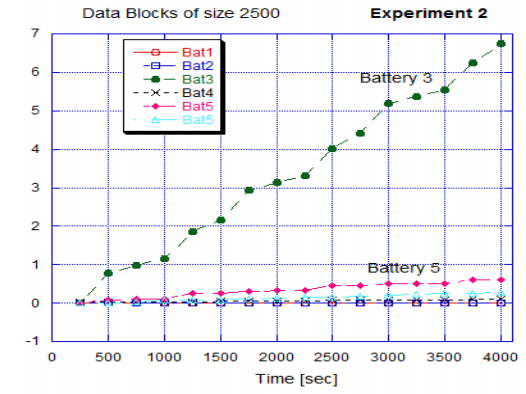

Fig. 10 Results from the Cumulative (Dynamic) ranking, based on data blocks with length *500 sec*.

VI. Discussions and Conclusions

The proposed method for detection of deviation in the performance of multiple channels of dynamical processes is relatively inexpensive from a computational viewpoint. Therefore it is intended to be used for *online detection* and *monitoring*, by processing the information from a sequence of Data Blocks, each of them with a fixed length of data. The information in one Data Block contains the outputs from all channels. The processing time obviously depends on the computational method, but it should be shorter than the time needed to collect the data for the next Data Block.

Since this method is for *monitoring*, but not for real-time control, longer Data Blocks with hundreds to thousands of data can be used and processed. This makes the result from the detection more plausible, since we need to detect important *trends* in the deviation (deterioration) of some channels.

We have introduced here a simple one-dimensional *multi-agent* clustering algorithm that automatically finds the number of clusters that represent in a compressed way the outputs from each channel. These clusters are further on used for similarity analysis of all pairs of channels and for extracting the characteristics of each channel, in the form of weighted global distance. Finally the channels are ranked in two ways: *instant* and *cumulative* ranking, in order to show the trends of changing in the deviation of each channel.

One typical example for application is given in this paper, namely detection of deviation of six batteries in an electric vehicle, but there could be many other applications. For example, in continuous industrial processes that have parallel production lines, this method could be used as a tool for *online monitoring* and *early detection* of the trends of deviation and deterioration, in order to take respective measures and prevent further serious problems and malfunction of the whole process.

There is at least one problem that should be solved in a more objective and practical way as a future research, namely: to separate by a *borderline* the group of "normal" channels from the remaining group of "deviated" channels. Currently it is done visually, in a subjective way, by looking at the plot of the cumulative ranking, but another way (possibly a fuzzy inference or clustering) could give a better objective solution.

References

[1] G. Vachkov, S. Byttner, M. Svensson, "Battery aging detection based on sequential clustering and similarity analysis", Proc. of the 6th IEEE Int. Conference "Intelligent Systems", IS-12, Sofia, Bulgaria, Sep., 2012, pp. 42-47.

[2] T. Martinetz, S. Berkovich and K. Schulten, "Neural-Gas network for vector quantization and its application to time-series prediction", *IEEE Trans. Neural Networks*, Vol. 4, No. 4, pp. 558-569. 1993.

[3] L. Xu, A. Krzyzak and A. Oja, "Rival penalized competitive learning for clustering analysis, RBF net and curve detection", *IEEE Trans. Neural Networks*, Vol. 4, No. 4, pp. 636-649, 1993.

[4] J.C. Bezdek, Pattern recognition with fuzzy objective function algorithms, New York: Plenum Press, 1981

[5] A. Jain, "Data clustering: 50 years beyond K-means", *Pattern Recognition Letters*, vol. 31, pp. 651-666, 2010.

[6] R. Yager and D. Filev, "Approximate clustering via the mountain method", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 24, No. 8, pp.1279-1284, 1994.

[7] A. Clark and D. Filev, "Clustering techniques for rule extraction from unstructured text fragments", CD-ROM Proc. of the NAFIPS'05 Conference, Ann Arbor, USA, July 2005.

[8] P. Angelov, "An Approach for fuzzy rule-based adaptation using on-line clustering", *International Journal of Approximate Reasoning*", vol. 35, no. 3, pp. 275-289, 2004.

[9] G. Vachkov, "Temporal and spatial evolving knowledge base system with sequential clustering", Proc. of the 2010 World Congress WCCI 2010 (FUZZ-IEEE 2010), Barcelona, Spain, pp. 3010-3017. July, 2010.