Alert Systems for production Plants

A Methodology Based on Conflict Analysis

Nielsen, Thomas Dyhre; Jensen, Finn Verner

Published in:
Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty

Publication date:
2005

Document Version
Publisher's PDF, also known as Version of record

Alert Systems for production Plants: A Methodology Based on Conflict Analysis

Thomas D. Nielsen and Finn V. Jensen

Department of Computer Science
Aalborg University
Fredrik Bajers vej 7E
9220 Aalborg Ø, Denmark
tdn@cs.aau.dk and fvj@cs.aau.dk

Abstract. We present a new methodology for detecting faults and abnormal behavior in production plants. The methodology stems from a joint project with a Danish energy consortium. During the course of the project we encountered several problems that we believe are common for projects of this type. Most notably, there was a lack of both knowledge and data concerning possible faults, and it therefore turned out to be infeasible to learn/construct a standard classification model for doing fault detection. As an alternative we propose a method for doing on-line fault detection using only a model of normal system operation, i.e., it does not rely on information about the possible faults. We illustrate the proposed method using real-world data from a coal driven power plant as well as simulated data from an oil production facility.

1 Introduction

Most production plants are equipped with sensors providing information to a control room where operators monitor the production process. Based on skill and experience the operators are alerted if something unusual happens, and through inspection of sensor readings, or derivatives thereof (so-called soft sensors), a diagnostic process may be initiated.

In connection to a joint project with an energy consortium, we have been working on establishing an alert system for a coal driven power plant. By an alert system we mean a system that, based on sensor readings, raises a flag in case of an abnormal situation. We intended to base the system on a Bayesian network representation [15, 10] of the power plant, and to help establish the model we had access to process engineers and an extensive database of logged sensor data. However, during the course of the project we encountered several problems, which we believe are common for projects of this type:

1. The engineers' knowledge of the plant is not sufficient for providing a causal structure.
2. The production process is so complex that it is difficult for the engineers to specify the possible faults (abnormal situations) and, in particular, how these faults would manifest themselves in the sensor readings.

3. The time constants, describing the delay from event to effect, are difficult to determine.
4. Faults are so rare that statistics cannot be used to learn neither the structure nor the parameters of a model of the faults.
5. As there is a difference between a true value and its sensor reading, true values should appear as hidden variables.

Faced with these problems, one approach would be to get as much causal structure from the engineers as possible and to combine this information with a data driven learning method. Unfortunately, state of the art of structural learning algorithms cannot cope with domains with a massive set of hidden variables. Furthermore, due to the lack of knowledge about the possible faults it is not obvious how such a model should subsequently be used for classifying abnormal behavior.

In this paper we propose an alternative methodology for on-line detection of abnormal behavior in production systems. The method focuses on systems which are prone to the problems described above, and it has the desirable property that it does *not* require information about the possible faults nor a model of abnormal behavior. We illustrate the proposed method using real-world data from the above mentioned power plant as well as simulated data from an oil production facility.

2 The proposed methodology

As implied above, it is not obvious how to construct a classifier (encoding the possible faults) for detecting abnormal behavior; neither in the form of a causal model nor in the form of e.g. a Naïve Bayes model [7] or a tree augmented Naïve Bayes model [8].

Instead, we propose to learn a Bayesian network representing normal operation only. At each time step the model is then used to calculate the probability of the set of sensor readings for that time step. This probability is in turn used to evaluate whether the sensor readings are jointly outside the scope of normal operation. That is, the methodology we propose basically consists of two steps: (i) learning a model of the sensors for normal operation, and (ii) using the learned model to monitor the system, initiate alerts and perform on-line diagnostics. Note that the use of models for describing normal operation has also been explored in the model-based diagnosis community [6]: Based on a prespecified model of normality (formulated in first-order logic), each component in the system is assigned a state (either normal or abnormal) which is consistent with both the model and any observations made of the system.

2.1 Learning a model

The available database consists of sensor readings that have been logged during normal system operation; each instance in the database can be seen as a "snapshot" of the overall production process. In what follows we shall assume that

this production process is composed of an ordered collection $(C_1, C_2, \ldots, C_n)$ of components (or sub-processes). The output of component C_i serves as input to component C_{i+1}, and (for ease of exposition) each component, C_i, is assumed to be equipped with a single sensor, S_i. For instance, when tracking the coal in a power plant we can, at an abstract level, describe the overall production process as being composed of three components: the silo, the coal mill, and the furnace. Since the production process is a physical non-instantaneous process we also have a delay (or time constant) associated with each of the components C, i.e., the time it takes for a particular unit (e.g. a piece of coal) to pass through that component.

Based on this perspective, we initially considered learning a model of the flow of one unit (e.g. coal) through the production plant. The variables in the learned model would then represent the sensors in the system. One approach for learning such a model would be to first transform the original database s.t. a case in the transformed database would correspond to the sensor readings related to one particular unit (this transformation is illustrated in Table 1). However, making such a transformation requires information about the time constants, and this information was unfortunately not available.

	S_1	S_2	$\cdots$	S_n
$\boldsymbol{c}_1$	$\underline{x_1^1}$	x_2^1	$\cdots$	x_n^1
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_j$	x_1^j	$\underline{x_2^j}$	$\cdots$	x_n^j
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_k$	x_1^k	x_2^k	$\cdots$	$\underline{x_n^k}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_N$	x_1^N	x_2^N	$\cdots$	x_n^N

$\Rightarrow$

	S_1	S_2	$\cdots$	S_n
$\boldsymbol{c}_1$	$\underline{x_1^1}$	$\underline{x_2^j}$	$\cdots$	$\underline{x_n^k}$
$\boldsymbol{c}_2$	x_1^j	x_2^l	$\cdots$	$x_n l'$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_k$	x_1^k	x_2^m	$\cdots$	$x_n^{m'}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$\boldsymbol{c}_{N'}$	$x_1^{N'}$	$x_2^{N'}$	$\cdots$	x_n^N

Table 1. The original database is transformed s.t. each case in the resulting database contains the sensor readings related to one particular unit in the system. Note that in the tables above we have assumed that the time delay between sensor S_1 and S_2 corresponds to the sampling delay between case/snapshot $\boldsymbol{c}_1$ and $\boldsymbol{c}_j$ in the original database.

An alternative approach would be to learn a dynamic Bayesian network model directly from the database by treating the cases as representing a trajectory through the system [9, 1]. Unfortunately, learning such a model also requires information about the time constants.

Instead, we simply focused on learning a Bayesian network model over the sensor variables directly from the database. This approach, however, has a potential computational drawback in the sense that we must expect the learned model to be very dense (this was also confirmed in the empirical experiments). To see this, consider Fig. 1 which illustrates a simplified temporal causal model of the data generation process for a production plant. Learning a model for the sensor variables can now conceptually be seen as learning a model that describes

the marginal distribution over the sensor variables S_i in a time slice. However, according to Fig. 1 we see that after very few time steps, each pair of variables in a time slice are dependent no matter how we condition on the other variables in the time slice. This is not only due to the hidden variables (modeling the components in the system), but also because standard learning methods treat the cases as being independent [4]; the latter corresponds to the past being unobserved.

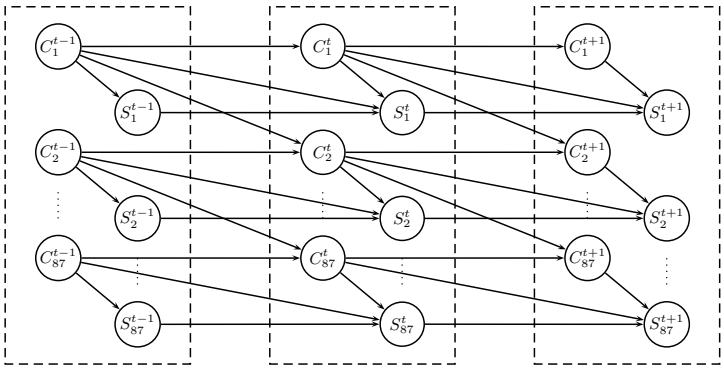


Fig. 1. The figure illustrates a dynamic Bayesian network representation of the data generation process for a production plant. The variable S_i represents the sensor associated with component C_i, and the arcs going into a sensor variable from a previous time slice models that the state of a sensor (correct, faulty or drifting) has an impact on the next sensor reading.

2.2 Initiation of alerts

The sensor readings are received in a constant flow, which is chopped up into time steps of, say, 1 second. This means that for every second we have evidence consisting of a value for each variable in the model.

Let the evidence be $\bar{e} = \{e_1, \ldots, e_n\}$, where e_i is a sensor reading. We can now calculate the conflict measure for the evidence as [11]:

$$\text{conf}(\bar{e}) = \log \left[\frac{P(e_1) \cdot \ldots \cdot P(e_n)}{P(\bar{e})} \right].$$

The probabilities $P(e_i)$ can be read directly from the Bayesian network in its initial state, and it does not require any propagation. As all variables in the model are instantiated, $P(\bar{e})$ is also very easy to calculate: It is simply the product of the appropriate entries in the conditional probability tables of the Bayesian network, and no propagation is required, i.e, the complexity is linear in the number of variables in the model.

Since the learned model represents normal system operation we would in general expect that sensor readings recorded during normal operation are positively correlated (i.e., $\text{conf}(\bar{e}) \leq 0$) relative to the model. Thus, when $\text{conf}(\bar{e}) > 0$ then this is an indication of an abnormal situation, and an alert may be triggered,

see also [13, 12]. The conflict measure can also be interpreted as a soft measure of inconsistency: If a case is inconsistent with the model, then it has probability 0, and if it is close to being inconsistent then it has an unusual low probability; "unusual" is for this measure calculated relative to the model for complete independence. For the conflict measure above, we expect a rather constant level for conf(·) under stable normal operation. When the process is changed, and it transforms from one mode of normal operation to another, we should expect oscillations in the conflict values until the changes have propagated and resulted in a new stable mode of normal operation.

As noted above, a positive conflict value is an indication of an abnormal situation. On the other hand, a negative conflict value does not necessarily imply that we have a normal situation as it may hide a serious conflict: If the sensors are strongly correlated during normal operation, the conflict level will be very negative, and a few conflicting sensor readings may therefore not cause the entire conflict to be positive. This can also be seen from the following proposition.

Proposition 1. *Let* $\bar{e}^x = \{e_1^x, \ldots, e_n^x\}$, $\bar{e}^y = \{e_1^y, \ldots, e_m^y\}$, *and* $\bar{e} = \bar{e}^x \cup \bar{e}^y$. *Then*

$$\mathit{conf}(\bar{e}) = \mathit{conf}(\bar{e}^x, \bar{e}^y) + \mathit{conf}(\bar{e}^x) + \mathit{conf}(\bar{e}^y),$$

where $\mathit{conf}(\bar{e}^x, \bar{e}^y) = \log\left[\frac{P(\bar{e}^x)P(\bar{e}^y)}{P(\bar{e})}\right]$.

So, it may happen that $\bar{e}^x$ and $\bar{e}^y$ are internally so strongly correlated that they dominate a conflict between the two sets. Thus, even when the conflict is negative, we shall watch out for jumps in the conflict level that may indicate a potential abnormal situation.

When an alert has been triggered, the system can start tracing the source of the alert. Various ways of tracing the conflict may be used. In our case we perform a greedy conflict resolution: recursively remove the sensor reading that reduces the conflict the most, and continue until the conflict is below a predefined threshold. This procedure can be performed very fast by exploiting lazy propagation [14] or fast retraction [5], as can be seen from the following proposition.

Proposition 2. *Let* $\bar{e}$ *be evidence,* X *a variable with evidence* e_x, *and* $\bar{e}^{-x}$ *the remaining evidence. Then*

$$\mathit{conf}(\bar{e}) = \log\left[\frac{P(e_x)}{P(e_x|\bar{e}^{-x})}\right] + \mathit{conf}(\bar{e}^{-x}).$$

That is, the reading with lowest normalized likelihood given the other readings contributes the most to the conflict. Note that as the Markov blanket of X is instantiated, the calculation of $P(e_x|\bar{e}^{-x})$ can be performed locally.

3 Empirical results

The proposed methodology has been tested on real-world data from a coal based power plant as well as simulated data from an oil production facility; in the latter

case the data was generated based on a model that includes the dynamics of the facility as well as control loops.

3.1 Power plant data

We received data about the power plant under normal system operation with load average $90-100\%$, i.e., the power plant operated between 90% and 100% of its full capacity. The data set contains 9600 cases, and each case consists of 87 simultaneous observations with no missing values.[1] The cases does not only contain actual sensor values, but they also include soft sensors, i.e., artificial "sensors" that have been computed based on the values of other sensors, as well as set-points and other indirect signals. As a preprocessing step, all data sets were naively discretized using equal width binning, where the number of bins were chosen (based on several tests) to be 3. Based on the preprocessed data, we learned a Bayesian network model as described in Section 2.1; the actual learning was performed using the software tool PowerConstructor with a 0.1-threshold for the conditional independence tests [2, 3].[2] Since the database is complete, the parameters of the model could simply be estimated using frequency counts.

In addition to the data sets for normal system operation, we received three data sets that each contained 1441 cases. Two of the data sets covered actual errors/abnormal situations whereas the last represented an "unusual behavior" that it would be interesting to detect:

- The fall-pipe leading coal into the power plant becomes clogged.
- A temperature sensor becomes faulty.
- A load change (from $60-75\%$ to $90-100\%$) occurs while the water concentration is high.

We have tested the proposed methodology by simulating on-line performance using the "clogged fall-pipe" data set as well as the "faulty-sensor" data set. Both tests were performed "blind-folded", i.e., we first analyzed the data and then, *after* the analysis, we discussed our findings with the domain experts.

A plot of the conflict measures for the "clogged fall-pipe" data set is depicted in Fig. 2. From the plot we see that we have positive conflict measures from observation 1136 and forward, i.e., the conflict measures indicate that the system makes a transition from a normal to an abnormal system state at 1136. This is also consistent with the information provided to us, namely that the system entered an abnormal state (the fall-pipe became clogged) between 1100 and 1144. Another interesting aspect of the plot is the fluctuations in the conflict measure that appears around observation 700 and lasts until approximately 780. We were later told that in this interval the system actually made a short change in load average from 99% to 84% and then back again.

[1] Since each case contains sensors readings for a particular point in time, the database can also be interpreted as a sequence of "snapshots" of the plant.

[2] The structure of the learned model is not included in this paper, since it is only used as a factorization of the joint probability distribution and should not be subject to interpretation from e.g. a causal point of view.

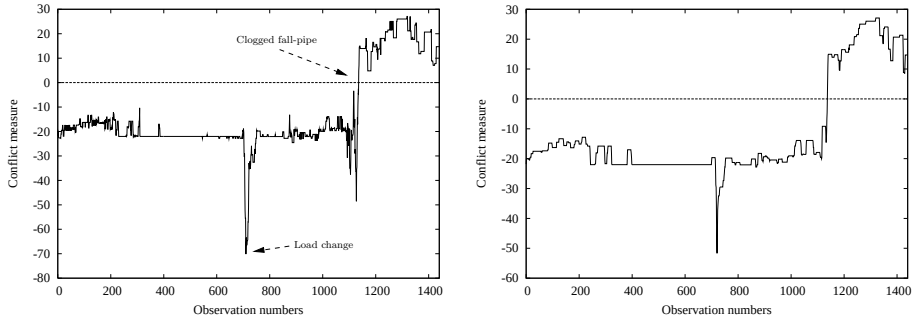


Fig. 2. The left hand figure shows a plot of the conflict measure for each case in the "clogged fall-pipe" data set; a value above 0 indicates a conflict. Note how the conflict measure is affected by the load-change and the fall-pipe becoming clogged. To reduce the noise in the data, the right hand figure shows the 0.9 percentile of the last 30 cases.

When performing conflict resolution, the algorithm indicates that the sensor measuring the water-percentage in the coal can explain all the conflicts. Ideally, we would have liked the system to pinpoint that the fall-pipe is clogged, however, this would require a sensor placed at that location. Since the system does not include such a sensor, we interpret the result as indicating that there is an inconsistency in the energy balance of the system and that this inconsistency can best be explained by the water percentage in the coal; this was also consistent with the analysis by the engineers.

A similar test was made on the "faulty sensor" data set, where the conflict measures can be seen in Fig. 3. As suggested by the plot, the conflict measure indicates that the system entered the abnormal state prior to the first observation; this was later confirmed by the engineers. We were also informed that in the beginning of the data set and around observation 600, there were two quick changes in the load averages (from $90-100\%$ to 80% and back again); these changes are reflected as quick changes in the calculated conflict measures. Finally, we were told that around observation 600 the temperature drops from 100°C to 90°C (at which level it stays for the remaining observations). Observe, that around this observation we also see a permanent drop in the conflict measure.

When performing conflict resolution we found that after observation 600, there were six significant sensors that could explain the conflict. We were informed that four of the sensors were actually significant for this scenario, but that the other two "sensors" should not have been picked out since they were set-points rather than sensors. However, the identification of these sensors actually makes sense as there is a conflict between the system sensors and the set-points. A simple approach for solving this problem could be to take such prior knowledge into account during conflict resolution.

Finally, we have made a tentative analysis of the "load-change" data set. A difficulty with this data set is that the learned model only covers normal operation during load average $90-100\%$. Hence, we have only considered the

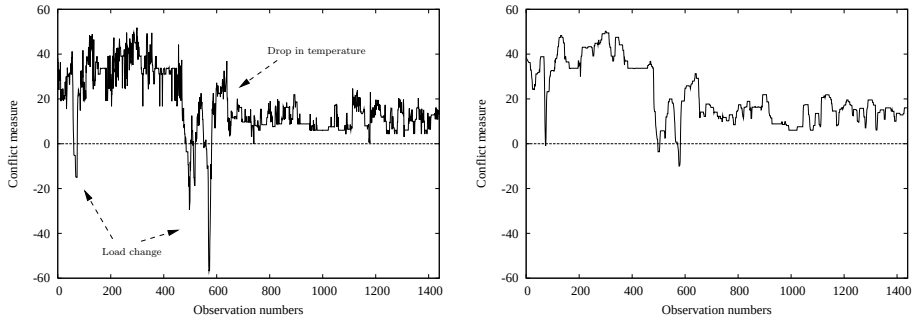


Fig. 3. A plot of the conflict measure for each case in the "faulty-sensor" data set; a value above 0 indicates a conflict. Note how the conflict measure is affected by the load-changes and the drop in temperature. The right hand figure shows the 0.9 percentile of the last 30 cases.

observations made after the load change has been completed, and where the distinguishing characteristics of the data set is that the coal has a high water concentration. That is, the data set has *not* been produced from a system state which should be classified as being abnormal, but rather an unusual system state that it would be interesting to detect (in case it would eventually result in an abnormal state). Fig. 4 shows a plot of the conflicts after observation 550 where the load change has been completed.

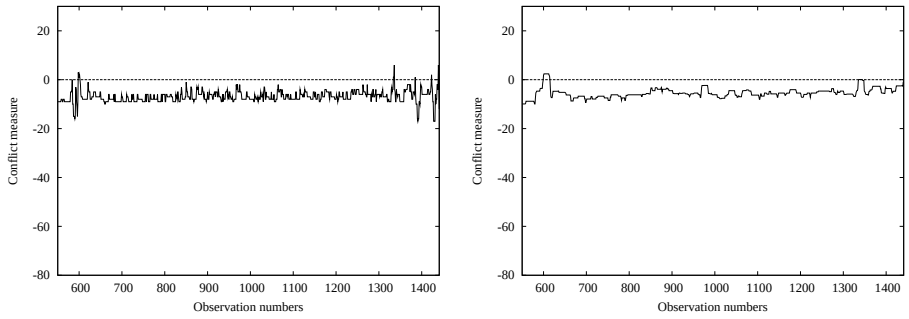


Fig. 4. A plot of the conflict measure for the "load-change" data set after the change has taken effect. The system is correctly classified as *not* being in an abnormal state. The right hand figure shows the 0.9 percentile of the last 30 cases.

As can be seen from the figure, the conflict values are all below 0 (except for a few single cases). This is consistent with the system not being in an abnormal state. However, from the measurements we can also see that the average

conflict value is higher than for normal operation: For the "load-change" data set, the average conflict value is -7.44, but during normal operation in the "clogged-fall-pipe" data set the average conflict value is between -10.34 and -22.8 with an average of -19.96. That is, you may be able to discriminate between different types of normal system operation by also considering the value of the conflict measure and not only whether it is positive or negative.

3.2 Oil production data

We have received a database with 10000 simulated cases for normal system operation for an oil production facility; each case in the database covers 140 sensors with white noise added to the sensor values.[3] The database was generated from a temporal causal model, which also simulated standard process variations. Hence, the database shares the same characteristics w.r.t learning as the power plant database (see Section 2.1). All of the sensor values appeared as real-valued output, so as a preprocessing step all variables/sensors were discretized. The actual discretization was performed using cross-validation to find the number of bins (with a maximum of 5) that maximizes the estimated likelihood of the data; the actual discretization was performed using Weka [16].

In order to test the proposed methodology in this setting, we used two other data sets both containing 10000 cases. The first data set had been generated by simulating faults in the pumping system whereas the second data set had been generated by simulating faults in the cooling system (see also Table 2).

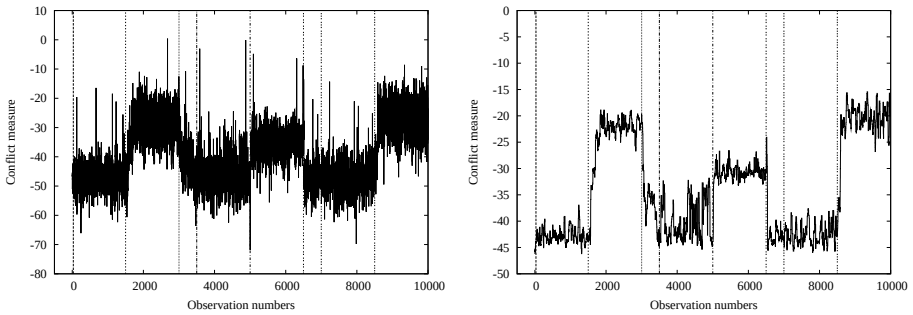
Time:	"Pump" data set	"Cooling" data set
30	Small leak in the pump	Small external leak in the cooling system
1500	Large leak in the pump	Large external leak in the cooling system
3000	Normal operation	Normal operation
3500	Small degradation of motor efficiency	Small internal leak in the cooling system
5000	Large degradation of motor efficiency	Large internal leak in the cooling system
6500	Normal operation	Normal operation
7000	Small degradation of pump efficiency	Moderate fouling
8500	Large degradation of pump efficiency	Significant fouling

Table 2. The table summarizes the changes in the production process for the "Pump" data set and the "Cooling" data set, respectively. Note that the changes in the two scenarios are initiated at the same points in time.

A plot of the conflict measure for the "Pump" data set is depicted in Fig. 5(a); as in the previous section, Fig. 5(b) shows the 0.9-percentile over the last 30 cases. The vertical lines in the two plots correspond to the points in time where

[3] Similar to the power plant database, the database can be interpreted as a sequence of "snapshots" of the facility.

changes are initiated (see Table 2). As can be seen from Fig. 5, there are significant changes in the conflict measure at time 1500, 3000, 5000, 6500 and 8500, which either correspond to large errors in system operation or changes back to normal system operation. From Table 2 we see that the changes appearing at 30, 3500 and 7000 correspond to small errors in the system operation and, accordingly, they are also less apparent in the plots. In particular, the change which appears at 3500 occurs before the system has settled into stationary normal system operation.



(a) (b)

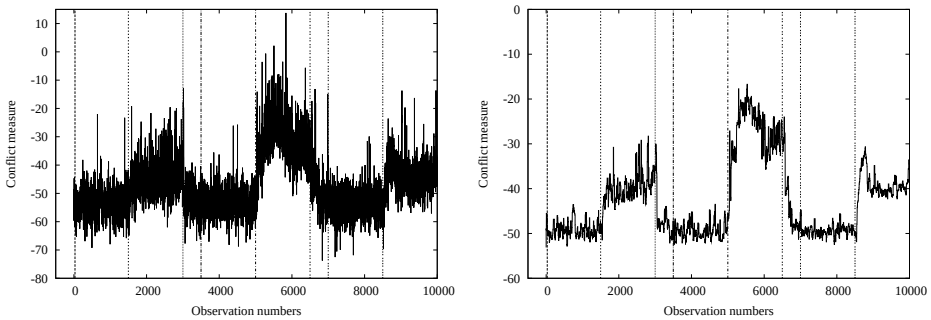
Fig. 5. The left hand figure shows a plot of the conflict measure for each case in the "Pump" data set. The vertical lines indicates when a change in the production process is initiated as specified in Table 2. The figure to the right shows the 0.9 percentile of the last 30 cases.

A similar plot of the conflict measure for the "Cooling" data set is depicted in Fig. 6(a). Analogously to the previous data set, there is a significant change in the conflict measure for all errors except at time 30, 3500 and 7000.

Observe that the conflict measures for both databases are all negative, which is a consequence of the decomposition property (Proposition 1) as discussed in Section 2.2. Thus, in order to detect changes in system operation we need to track jumps in the conflict measure. However, a method for performing this analysis is a subject for future research.

4 Conclusion and future work

We have proposed an alert system methodology based on conflict analysis. A distinguishing characteristic of the proposed methodology is that it only relies on a model for normal system operation, i.e., knowledge about the possible faults is not required. Moreover, the computational complexity of the algorithm ensures that on-line analysis is feasible. The methodology has been successfully tested on both real-world data from a power plant and simulated data from an oil production facility.



(a) (b)

Fig. 6. The figure to the left shows a plot of the conflict measure for each case in the "Cooling" data set. The vertical lines indicates when a change in the production process is initiated as specified in Table 2. The right hand figure shows the 0.9 percentile of the last 30 cases.

As part of ongoing research and future work, we are working on establishing alternative straw models in order to perform a more refined conflict analysis; see also the discussion in [13, 12] concerning the independence straw model [11]. Having an alternative straw model might also reduce the effect of the decomposition property. I.e., when faulty sensors' impact on the conflict measure is dominated by strongly correlated sensors. Furthermore, we are considering procedures for tracking changes in the actual value of the conflict measure in order to perform early fault detection by identifying trends in the behavior of the system being monitored, e.g. if the system "drifts" towards an abnormal state.

Acknowledgments

We would like to thank Rasmus Madsen and Babak Mataji from ELSAM engineering for providing us with data from the power plant. We would also like to thank John-Morten Godhavn from Statoil ASA for supplying us with data from the oil production facility, and Erling Lunde from Dynamica AS for helpful comments regarding the technical layout of the facility. Finally, we would like to thank Helge Langseth for valuable discussions and comments, and `Hugin Expert` (www.hugin.com) for giving us access to the *Hugin Decision Engine* that forms the basis of our implementation.

References

1. Xavier Boyen, Nir Friedman, and Daphne Koller. Discovering the hidden structure of complex dynamic systems. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the Fifthteenth Conference on Uncertainty in Artificial Intelligence*, pages 91–100. Morgan Kaufmann Publishers, 1999.

2. Jie Cheng, David A. Bell, and Weiru Liu. Learning belief networks from data: An information theory based approach. In *Proceedings of the Sixth ACM International Conference on Information and Knowledge Management*, pages 325–331, 1997.
3. Jie Cheng, Russell Greiner, Jonathan Kelly, David Bell, and Weiru Liu. Learning Bayesian networks from data: an information-theory based approach. *Artificial Intelligence*, 137(1-2):43–90, 2002.
4. Gregory F. Cooper and Edward Herskovits. A Bayesian Method for Constructing Bayesian Belief Networks from Databases. In Bruce D. D'Ambrosio, Philippe Smets, and Piero P. Bonissone, editors, *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 86–94, 1991.
5. A. Philip Dawid. Applications of a general propagation algorithm for a probabilistic expert system. *Statistics and Computing*, 2:25–36, 1992.
6. Johan de Kleer and James Kurien. Fundamentals of model-based diagnosis. In *Proceedings of the fifth IFAC symposium on Fault Detection, Supervision, and Safety of technical Processes (Safeprocess)*, pages 25–36, 2003.
7. Richar O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.
8. Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2–3):131–163, 1997.
9. Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the Structure of Dynamic Probabilistic Networks. In Gregory F. Cooper and Serafin Moral, editors, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, 1998.
10. Finn V. Jensen. *Bayesian networks and decision graphs*. Springer-Verlag New York, 2001. ISBN: 0-387-95259-4.
11. Finn V. Jensen, Bo Chamberlain, Torsten Nordahl, and Frank Jensen. Analysis in hugin of data conflict. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, 1990. Also published in Uncertainty in AI 6, 519-528, North-Holland, Amsterdam, 1991.
12. Young-Gyun Kim and Marco Valtorta. On the detection of conflicts in diagnostic Bayesian networks using abstraction. In Philippe Besnard and Steve Hanks, editors, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 362–367. Morgan Kaufmann Publishers, 1995.
13. Kathryn Blackmond Laskey. Conflict and surprise: Heuristics for model revision. In Bruce D. D'Ambrosio, Philippe Smets, and Piero P. Bonissone, editors, *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 197–204. Morgan Kaufmann Publishers, 1991.
14. Anders L. Madsen and Finn V. Jensen. Lazy evaluation of symmetric Bayesian decision problems. In Kathryn B. Laskey and Henri Prade, editors, *Proceedings of the Fifthteenth Conference on Uncertainty in Artificial Intelligence*, pages 382–390. Morgan Kaufmann Publishers, 1999.
15. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Representation and Reasoning. Morgan Kaufmann Publishers, San Mateo California, 1988. ISBN 0-934613-73-7.
16. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 2000. Version 3.4.3.