Real-time Alarm Management System for Wide-Area Monitoring

Chumki Basu[1], Jagabondhu Hazra[1], Kaushik Das[2],
Deva P. Seetharam[1]
[1]IBM Research – India, Bangalore, India
[2]Technical University of Denmark, Roskilde, Denmark
{chumbasu,jahazra1,dseetharam}@in.ibm.com
kdas@dtu.dk

***Abstract*— Control center alarms for power systems usually arrive with low latency, quickly overwhelming operators. Previous approaches to dealing with alarm ordering include logical grouping (suppression and filtering) of alarms according to pre-defined rules and presentation of the resulting alarms to the operator. Knowledge-based approaches for alarm handling allow us to incorporate operator experience into the decision making process. However, there are limitations that impede their use in real-time systems. In this paper, we approach these limitations from two perspectives. First, we propose a mapping from knowledge representations such as network models (i.e., of a power system) and ontologies to the design of stream computing applications for efficient alarm processing. Secondly, we propose a new class of experiential, knowledge-based algorithms for ranking alarms and introduce a *contextual ranking algorithm* in this class. We also present initial evaluation results of this algorithm using a simulation-based metric.**

***Index Terms*—Wide-area monitoring, analysis and simulation, real-time control, intelligent alarm processing, protection systems.**

I. Introduction

Over decades-long research [1][2][4]-[8] has demonstrated that infusing intelligence into alarm processing is a promising approach. With large-scale deployments of Phasor Measurement Units (PMU), which provide low latency data at rates ranging from 30-120 samples per second, along with the availability of stream computing environments that enable processing “data in motion”, we note the suitability of a lightweight, knowledge-based software framework for alarm processing.

Operator alarm overload is a well studied problem, and it has been shown that a control center operator may not be able to handle alarm rates as low as 10 alarms per minute (alarm rates during the first few minutes of a major disturbance are on average closer to 80 alarms per minute) [1]. This scenario illustrates the challenge of taking necessary and effective control and protective actions in real-time.

Related work in intelligent alarm processing has suggested a number of approaches to tackle this problem [1]-[9]. Traditional knowledge-based approaches are subject to certain limitations, including the need for sufficient representative data of different alarm conditions to build accurate models and the slowness of processing speed incurred using existing knowledge representations and inference mechanisms [2]. Some recommendations for improvements based on these limitations are as follows [2]:

1. Analyzing contingencies faster and with more confidence
2. Utilizing more redundant data to enhance existing data and conclusions
3. Dealing with overwhelming amount of alarms by classifying them according to the causes
4. Archiving field data for future analysis of disturbances and related operator actions

This paper is focused primarily on addressing the first and the third recommendation. In spirit, the second recommendation suggests combining data from multiple data sources (for example, SCADA data in addition to PMU data), so we leave this for future work. Similarly, since archiving is more concerned with “data at rest” rather than “data in motion”, we also leave the fourth recommendation as a future extension. We address both the first and the third recommendation by showing how to enrich knowledge representations with experiential knowledge and how to translate this knowledge into software architectural decisions for designing the alarm management system. Sensor data from PMUs arrive as continuous streams with low latency. Power industry interest in stream computing arises with requirements to acquire, analyze, and respond to enormous numbers of complex events in real-time, and we follow in this tradition. We also build on prior art on ranking alarms according to their temporal and spatial characteristics and validating ranking algorithms to identify causal alarms [3].

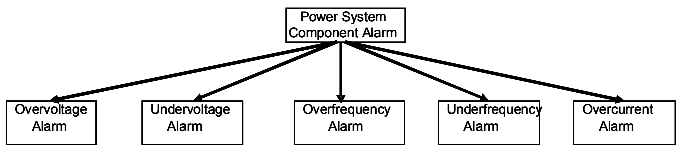

Figure 1. Partial taxonomy of alarm types.

II. Sources of Knowledge

A. Taxonomic knowledge

Taxonomic knowledge can be considered closely tied to a pre-defined set of links or relations between states and entities in a domain. A taxonomy is commonly used in classification, is hierarchical, and is usually part of an ontology, which offers a broader characterization of the natural/physical world.

In Figure 1, we present an extension of an alarm taxonomy that is part of an upper ontology of alarms and associated control actions [3] by characterizing a set of alarms according to "alarm type". We focus on alarm type since we directly map this "attribute" to an attribute that is represented in a data stream tuple as we will discuss in Section III.

B. Operational knowledge

By operational knowledge, we refer to knowledge about the power system at any given time. Knowledge about the power grid includes topological information (e.g., connections between lines), geo-spatial data (e.g., coordinates of substation locations), and sensor data (e.g., measurements of power flow on a line). The above types of knowledge can be static as well as dynamic – while the locations of substations are not likely to change, the power flow measured on a line is reported for a given time instant. Operational knowledge of the power system can also be systematically organized in the ontology. However, until an integrated organizational structure is imported into the alarm management system, operators (and we) rely on data models derived from disparate pieces of data (e.g., flat file(s) containing a model of the grid, etc).

C. Experiential knowledge

Experiential knowledge of a control center operator encapsulates social, geo-political, as well as economic variables that impact decisions on the grid at a given time. For example, while operational knowledge may suggest prioritizing alarms associated with generators compared to loads, experiential knowledge may indicate prioritizing alarms on lines that service critical customers such as hospitals, mines, railways, etc.

Operators must balance knowledge derived from all of the above sources as they make real-time decisions in response to changing conditions on the grid. Our work is not meant to prescribe a procedure for taking these decisions but rather to supply the operator with as many facets of a problem as possible so that an informed decision may be taken in time. Specifically, in Section III, we show how to map taxonomic and operational knowledge to hierarchical, functional data stream operators that directly consume and transform data in the alarm management system. In Section IV, we discuss a new class of ranking algorithms that are based on experiential knowledge.

III. Stream Computing

A real-time data stream is a continuous sequence of data items. These items are typically seen only once and may arrive in any order [9]. Stream computing is a framework for managing workloads that provides the ability to direct continuous streams of input data along possibly data-dependent paths to multiple compute resources typically without needing to store intermediate results obtained at each compute resource. Applications of stream computing in power systems include voltage stability analysis [10] as well as transient stability analysis [11].

Although several stream computing platforms have been developed recently, we are using InfoSphere Streams [12], a stream processing middleware that supports high-performance, reconfigurable stream computing. InfoSphere Streams supports structured as well as unstructured data stream processing and the execution of multiple applications from multiple users simultaneously. These applications can be scaled to a large number of compute nodes and can interact at run time through stream importing and exporting mechanisms. InfoSphere Streams applications take the form of dataflow processing graphs (as shown in Figure 2) consist of a set of processing elements (PEs) connected by streams, where each stream has a fixed schema and carries a series of tuples. PEs are containers that host operators implementing data stream analytic, and are distributed on compute nodes. Compute nodes are organized as a shared nothing cluster of workstations or as the execution nodes in a large supercomputer. PEs communicate with each other via their input and output ports, connected by streams.

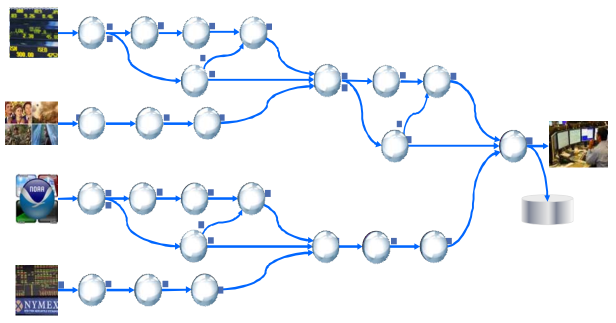

Figure 2. Data processing within a streams platform.

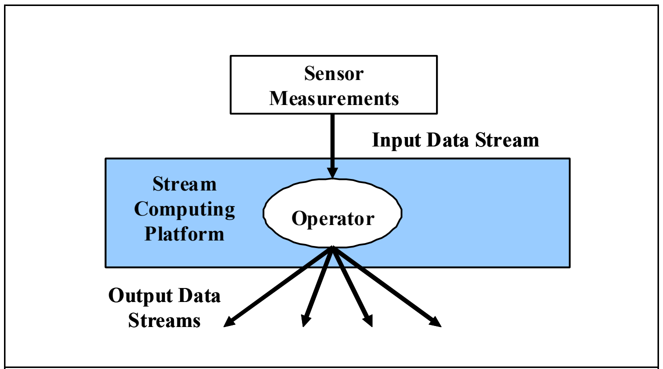

Figure 3. Initial decomposition of a data stream

IV. Data Management in Stream Computing

A. Definitions

An early and basic application of stream computing in sensor networks involved the analysis of power usage statistics sent to a power station with the goal of adjusting required power generation rate [13]. In that work, a stream of data is a continuous sequence of usage statistics that may be characterized across multiple dimensions such as user category, location, and time of day. Even then, it was important to be able to analyze a stream of sensor data based on not only the time of arrival of a measurement (temporal dimension) but also the locality of the sensor (spatial dimension).

A logical stream is the result of applying an operator such as *group-by* to a physical data stream. For example, if a physical data stream is a time-ordered sequence of PMU measurements (voltage magnitude and phase angle) for some service area, then a sample logical stream could be the subset of those measurements that correspond to a substation. The *group-by* operator is one of a class of continuous operators known as *multiplexing/de-multiplexing* operators for decomposing/merging logical streams [9].

In Figure 3, we show a simple decomposition of a physical input data stream into multiple output streams as described above. In this figure, we assume that the sensor measurements have been collected and time synchronized, e.g., by a phasor data concentrator, and received as a single input data stream. Alternatively, the stream computing platform can also accept multiple input PMU data streams, one per reporting substation. For the purposes of discussion, we assume the former scenario.

Prior art on analyzing e-mail streams modeled latent hierarchical structure in "bursts" of activity and associated natural meaning in terms of stream content with these bursts [14]. Unlike e-mail, bursts associated with PMU data are more likely to reflect the operational status of a PMU/network conditions. However, using the stream computing platform, we can impose a hierarchical structure on how physical data streams are decomposed into logical streams and how operators that apply functions on the logical streams can be composed for efficient real-time processing of alarms.

B. Top-Down Decomposition of PEs

Each severity index computation is a ranking function, R_i, that is applied to a stream of PMU data. Since every alarm is flagged in the data, we aggregate all the alarms and assign a rank score to each alarm, which is appended to its corresponding tuple. Our operators transform tuples by applying functions (called *functors*) to their attributes [16]. After a pre-defined time interval, we re-order the aggregated alarms based on their rank scores according to each ranking function.

In a stream computing platform, a ranking function is represented as a processing element (PE). Each PE computes an alarm ordering, ρ_1 (in parallel, with other PEs). The result of this comparison, computed by a Rank Selector PE, enables operators to respond in real-time (or near real-time) to alarms and enact mitigating procedures.

Whereas alarm grouping or filtering has been commonly discussed in the literature, typically, this has been applied with respect to static criteria specified in pre-defined rules. We present a customizable solution where the filtering criteria (captured in taxonomic and operational knowledge) are specified by domain experts externally, and consequently, are used by the alarm management system software to organize hierarchically and distribute the processing load across compute nodes. This organization facilitates the efficient generation of multiple rank orderings of alarms for evaluation, and eventually, selection of a rank ordering for alarm servicing and resolution in real-time.

C. Composing Operators

We defined severity indices (*SI*) [3], starting with indices that were *event-based*, i.e., based on the percentage of deviation from nominal value for each of the following events observed on bus_i at a given point in time. We defined a severity index for each alarm type shown in Figure 1: overvoltage (*SI_ov*), undervoltage (*SI_uv*), overfrequency (*SI_of*), underfrequency (*SI_uf*), overcurrent (*SI_oc*). For each severity index, the percentage of deviation from nominal value for the alarm type is specified in the ontology. In this section, we discuss how to represent severity index computation as a composition of data stream operators within a PE for a bus.

We defined event-based severity indices for each bus in the IEEE 30 bus test system shown in Figure 4. These indices do not have any spatial or temporal (time series) component. To compute event-based indices, we first introduce a processing element (PE) that splits the data stream into multiple streams using a *group-by* "bus" operator. For each bus, we also introduce a PE that repeats this procedure by splitting the input data stream into multiple streams, one per alarm type. The latter PE also contains a *functor* operator that computes the value of an event-based index.

We note the extensibility of our approach since we map specifications that reside outside of the alarm management system (in external data files containing the network grid model and the ontology, respectively) to the *group by* "attributes" – "bus" and "alarm type". The structure of the grid model and alarm taxonomy can change, but we do not have to change the design of the alarm management software to customize the solution in order to process a new alarm type or to ignore a bus that is not live due to maintenance. Instead, we update the ontology with a new alarm type, for which a new *functor* operator is instantiated, and a specification of the percentage deviation to compute the severity for this alarm type (percentage deviation maps directly to an argument of the *functor* operator). Similarly, we re-label the status of a bus with pertinent information from the network grid model.

After defining operators for event-based severity indices, we introduced three severity index operators that incorporated temporal (time series) knowledge: *percentage occurrence in time*, *critical deviation* and *area*. Percentage occurrence measures the fraction of samples, *s*, which are alarms of type τ, a_τ, in a fixed time window, W_{t-t0}.

$$O = \frac{|a_\tau \; in \; W_{t-t_0}|}{|s \; in \; W_{t-t_0}|} \tag{1}$$

Critical deviation measures the magnitude of deviation, d_t, for the current sample relative to the maximum deviation observed in the time window.

$$D_{crit} = \frac{d_t}{\max\{d \; in \; W_{t-t_0}\}} \tag{2}$$

Area is a normalized value of the area under the curve violating the limit (calculated using the trapezoidal method), where *N* represents the number of equally spaced intervals within a time window.

$$A = \frac{\frac{t-t_0}{2N}(f(x_1)+2f(x_2)+\ldots+2f(x_N)+f(X_{N+1}))}{(\max\{d \; in \; W_{t-t_0}\})t-t_0} \tag{3}$$

To compute each of the above indices, we apply function (1), (2), or (3) to a window of measurements per bus – the window for each temporal alarm is defined in terms of logical units (range of time covered by a window rather than a fixed number of tuples in a window) [15]. Each function is represented as a *functor* operator.

We then computed a combined temporal severity index for each *bus$_i$* by summing the individual indices in an *aggregation* operator.

$$SI_{Temporal_i} = SI\_ov + SI\_uv + SI\_of + SI\_uf + SI\_oc + O + D_{crit} + A \tag{4}$$

Finally, we computed a spatial severity index for each *bus$_i$* by summing a weighted average of the temporal indices across all buses *h* (max = 5) hops away, where weight is given by the inverse of the number of hops between buses. The spatial severity index is computed by applying an *aggregation* function on the *union* of data streams.

$$SI_{Spatial_i} = \sum_{h=1}^{5} \frac{SI_{Temporal_i}}{h} \tag{5}$$

The hierarchical decomposition of operators is illustrated in Figure 5.

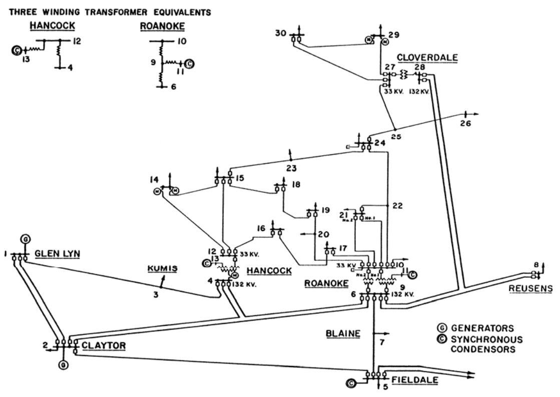

Figure 4. IEEE 30 bus test system.

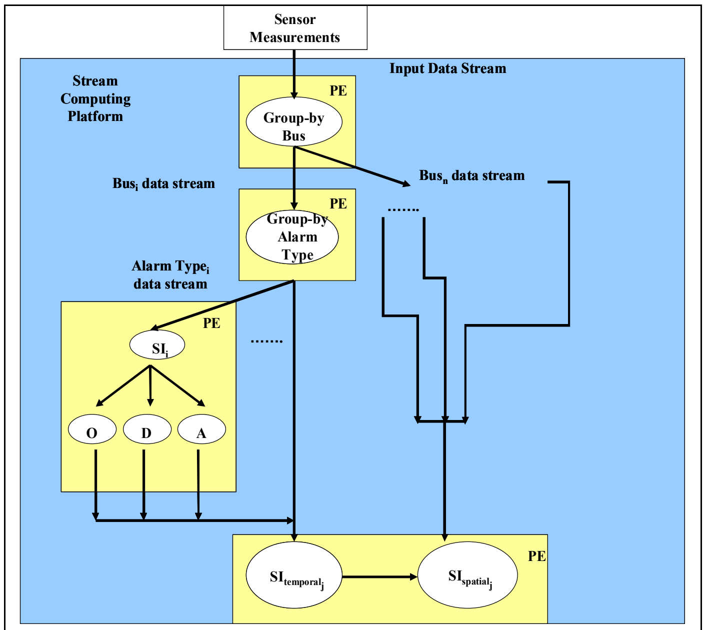

Figure 5. Hierarchical decomposition of data streams.

V. Ranking Alarms Based on Experiential Knowledge

In Section III, we discussed alarm ranking methods that incorporated both taxonomic knowledge (of alarm types) and operational knowledge (of the grid network). In this section, we present a third class of alarm ranking methods based on operators' experiential knowledge. This knowledge includes social, geo-political, and economic factors.

In keeping with our extensible approach to engineering a real-time alarm management solution, we present a generic method that can be applied across multiple socio-economic dimensions. Unlike the earlier indices where we computed absolute measures, here we compute an index that is based on the relative importance of the alarms.

Time (sec)	Spatial severity alarm ranking					Temporal severity alarm ranking					Contextual (adjacency, hotlines, etc.) severity alarm ranking				
70	7	5	28	27	22	5	2	1	7	4	10	2	1	20	21
80	7	5	28	27	22	7	5	2	1	3	10	2	1	20	21
90	7	5	28	27	22	7	5	2	1	3	10	2	1	20	21
100	7	5	28	27	22	7	5	2	1	3	10	2	1	20	21
110	7	5	28	27	22	7	5	2	1	3	10	2	1	20	21
120	7	28	5	27	22	7	5	2	1	3	10	2	20	21	17
130	28	27	24	22	30	7	5	1	2	3	10	2	1	20	21
140	7	28	5	27	22	2	1	30	26	29	2	10	1	20	21
150	28	27	24	30	22	7	5	2	1	3	10	2	20	21	17
160	28	7	27	8	22	7	5	6	28	4	2	10	1	20	21

Table 1. Alarm Rank Orderings Resulting from Spatial, Temporal, and Contextual Ranking Algorithms

At any given time, there can be multiple alarms on the grid. We can compare them based on their severity, their spatial adjacency to each other, their proximity to hotlines, sensitive areas, and special events, their respective economic costs, etc. We outline an approach to compute a relative measure of the importance of an alarm based on any number of experiential criteria, each represented by some cost function – we refer to this ranking algorithm as *contextual ranking*.

To compute contextual ranking, we itemize the costs associated with each alarm, a_i. We compute a matrix, A, where the rows and columns are alarms and where we assign cell, a_{ij}, a value of 1 *iff* alarm, $a_{i,}$, is collocated with alarm, a_j. Similarly, we define a matrix, H, where the rows are alarms and the columns are an enumeration of the hotlines. In matrix H, a cell, h_{ij}, is assigned a value of 1 *iff* alarm, a_i, is connected to a hotline, h_j (i.e., alarm is on a hotline or alarm is on a bus connected to a hotline). Since the ranking method is generic, we also incorporate the (event-based) severity of the alarm (based on alarm type l) in the cost function shown in (6). Note that the values of the severity indices are in per unit. (We plan to extend this formulation in the future by defining links between alarms and equipment representing economic cost, e.g., incorporating the cost of an outage associated with an alarm at a generator vs. a load.)

In keeping with the "experiential" nature of this method, we give the operator flexibility in weighting each of the criteria (costs). Currently, we assign equal weight, w_p (=1), to these costs and compute a cost function, C_i, for contextual ranking by taking the sum over the individual cost functions in (6). As part of future work, we will investigate fuzzy methods for assigning weights to combine the individual cost functions.

$$C_i = w_1 \sum_{j=1}^{m} a_{ij} + w_2 \sum_{k=1}^{n} h_{ik} + w_3 * s_{il} \tag{6}$$

VI. Experiments

To demonstrate the feasibility of our approach, we evaluated it on the benchmark IEEE 30 bus test system [17] as shown in Figure 4. So that we could compare our results to prior work, we used the cascading scenario presented in [3], starting with a 3-phase-to-ground fault (dead short circuit) for 100 ms near bus_2 on $line_{2\text{-}5}$. A fault was initiated at 60 seconds and was cleared at 60.1 seconds by tripping $line_{2\text{-}5}$. Along with the tripping of $line_{2\text{-}5}$, 20 MW load was lost at bus_2. There was a sudden voltage dip due to the short circuit. Even though voltage recovered, the generators were accelerating gradually due to electromechanical dynamics. At time t =120 seconds, $line_{2\text{-}6}$ tripped due to power swing. The tripping of two important lines, $line_{2\text{-}5}$ and $line_{2\text{-}6}$, resulted in severe overload on $line_{2\text{-}4}$ and $line_{1\text{-}3}$. Eventually, the overload relay tripped $line_{2\text{-}4}$ and $line_{1\text{-}3}$ soon after $line_{2\text{-}5}$ tripped. The cascaded tripping of 4 lines – $line_{1\text{-}3}$, $line_{2\text{-}4}$, $line_{2\text{-}6}$ and $line_{2\text{-}5}$ – within a few minutes divided the system into two islands. Bus_1 and bus_2 formed one island and the remaining buses formed the other island. The first island collapsed due to overfrequency (i.e., generation was much higher than the load). The second island also collapsed due to underfrequency (i.e., load was higher than generation). For this scenario, multiple alarms were generated and the severity-based alarm rankings for all three methods (Top 5 alarms per ranking method) are shown in Table 1.

We observed that the alarms for spatial and temporal rankings shown in Table 1 belong to different groups [3]. For example, in the case of spatial severity ranking, alarms on bus_5 and bus_7 are overcurrent alarms whereas alarms on bus_{28}, bus_{27} and bus_{22} are undervoltage alarms. For temporal severity ranking, alarms on bus_5 and bus_7 are overcurrent alarms, alarms on bus_1 and bus_2 are overfrequency alarms, and the alarm on bus_4 is an undervoltage alarm. Additionally, in Table 1, we observe similar behavior for contextual ranking – alarms on bus_{10}, bus_{20}, and bus_{21} are undervoltage alarms whereas alarms on bus_1 and bus_2 are overcurrent. It is also interesting to note that only one of the Top 2 alarms returned for contextual ranking appear in the Top 2 for the other methods in only two instances (for temporal severity ranking), thereby demonstrating that the contextual ranking method is exploiting different sources of information.

As in [3], we validate each alarm ranking in terms of the *reduction in the total number of alarms*. We do this by simulating the appropriate control actions, one at a time. For spatial severity ranking, for example, the alarm on bus_7 is serviced by minimizing congestion on $line_{5\text{-}7}$ by decreasing generation at bus_5 and increasing generation at bus_2. Even though this resolves the second alarm, the remaining alarms still exist. For temporal ranking, if we service alarms on bus_5 and bus_2 by 20 MW generation backing at *bus1* and generation rescheduling at bus_2 and bus_5, all the remaining alarms are resolved automatically. For contextual ranking, the under voltage alarm at bus_{10} was serviced by connecting a shunt capacitor at bus_{10}. However, it did not help in resolving the remaining alarms. As a next step, the overcurrent alarm at bus_2 was serviced by reducing generation at bus_2 and increasing generation at bus_5. Even servicing alarm 2 did not help in resolving the remaining alarms. Next, we tried to service alarm 1, which resolved the remaining alarms. In this example, we have shown how proactively resolving alarms through simulation can not only identify the "better" alarm ranking (in this case, the temporal ranking), but also pinpoint the "causal alarms" – alarms which when resolved minimize the total number of remaining alarms.

The simulation-based metric has been validated by comparing it to other rank evaluation metrics [3]. The simulation-based metric may be considered a time-based metric – in an operational system, the user (operator) would want to minimize the total number of alarms, and consequently, the time needed to bring the power system back to a stable state faster. To summarize, using this metric, the operator is able to evaluate alarm orderings using different ranking criteria. With the parallelism provided by the stream computing platform, it is possible to generate multiple alarm orderings at run-time and to choose the "best one".

VII. Related Work

Early alarm management systems were focused on logical analysis or rule-based processing of alarms [6]. For example, one way to reduce alarms is by logically grouping related alarms – this has sometimes been referred to as alarm "suppression" or "filtering" in the literature [1]. There are multiple rules for alarm filtering ranging from duration-based to static priority assignment [5]. More recent approaches for alarm reduction have focused on the extraction of features from raw data to understand the causal conditions behind alarms [2] and the use of inductive learning to find the root cause (failure tree) behind alarms [18].

VIII. Conclusions

In this paper, we have described how external knowledge captured in formal representations such as network models and ontologies can be mapped to the design of a stream computing application for efficient alarm processing. We also introduced a class of experiential knowledge-based algorithms for ranking alarms on the grid and described a representative algorithm from this class for alarm ranking based on contextual information. We evaluated this algorithm on a cascading fault scenario on a standard test bus system and compared it to other severity-based ranking algorithms. Although it did not perform as well as temporal severity ranking, this method exploits different sources of information (an example of the range of considerations that should be assessed before taking control decisions, as stated in [5]), thereby, giving operators the flexibility to experiment with different criteria/costs and eventually to find the best operational combination.

References

[1] M. L. Bransby and J. Jenkinson, "The Management of Alarm Systems: A Review of Current Practice in the Procurement, Design and Management of Alarm Systems in the Chemical and Power Industries," Bransby Automation Ltd, Cirencester, England, 1998.

[2] M. Kezunovic and Y. Guan. "Intelligent Alarm Processing: From Data Intensive to Information Rich," *Proceedings of HICSS 2009 (42nd Hawaii International Conference on System Sciences)*, Hawaii, pp. 1-8, Jan. 2009.

[3] C. Basu, K. Das, J. Hazra, and D. P. Seetharam. "Enhancing Wide-Area Control with Intelligent Alarm Handling," to appear in *Proceedings of ISGT Europe*, Denmark, Oct. 2013.

[4] M. Kezunovic and Yufan Guan, "Intelligent Alarm Processor", ERCOT Project Final Report, TEES, Mar. 2008.

[5] E. Kyriakides, J. W. Stahlhut, and G. Heydt. "A Next Generation Alarm Processing Algorithm Incorporating Recommendations and Decisions on Wide Area Control," *IEEE Power Engineering Society General Meeting*, pp. 1-5, 2007.

[6] R. W. Bijoch, S. H. Harris, T. L. Volkmann, J. J. Bann, and B. F. Wollenberg, "Development and implementation of the NSP intelligent alarm processor," *IEEE Trans. Power Systems*, vol. 6, no. 2, pp. 806-812, May 1991.

[7] A. H. Shoop and S. Silverman, "A real time alarm processor," *Electrical Power and Energy Systems*, vol. 14, no. 2/3, pp. 108-113, 1992.

[8] R. Khosla and T. Dillon, "Learning knowledge and strategy of a neuro-expert system architecture in alarm processing," *IEEE Trans. Power Systems*, vol. 12, no. 4, pp. 1610-1618, Nov. 1997.

[9] L. Golab and M. T. Oszu. "Issues in Data Stream Management," *Proceedings of SIGMOD Record*, Vol. 32, No. 2, 2003.

[10] J. Hazra, K. Das, D. P. Seetharam, A. Singhee, "Stream Computing-based Synchrophasor Application for the Power Grid", *Proceedings of the first international workshop on High performance computing, networking and analytics for the power grid, pp. 43--50*, 2011.

[11] J. Hazra, R. K. Reddi, K. Das, D. P. Seetharam, A. K. Sinha. "Power Grid Transient Stability Prediction using Wide-Area Synchrophasor Measurements", *IEEE PES Innovative Smart Grid Technologies (ISGT) Europe*, 2012.

[12] R. Rea. IBM InfoSphere Streams. "Redefining Real Time Analytic Processing", http://public.dhe.ibm.com/software/data/sw-library/ii/whitepaper/InfoSphereStreamsWhitePaper.pdf.

[13] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. "Multi-Dimensional Regression Analysis of Time-Series Data Streams," *Proceedings of the International Conf. on Very Large Data Bases*, 2002, pp. 323-334.

[14] J. Kleinberg. "Bursty and Hierarchical Structure in Streams," *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 91-101, New York, NY, 2002.

[15] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. "Models and Issues in Data Streams Systems", *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, New York, NY, 2002.

[16] J. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker, and S. Zdonik. "High Availability Algorithms for Distributed Stream Processing", *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, Washington D.C., pages 779-790.

[17] http://www.ee.washington.edu/research/pstca/pf30/pg_tca30bus.htm

[18] O. Aizpurua, R. Galan, and A. Jimnez. "A New Cognitive-Based Massive Alarm Management System in Electrical Power Administration", *Proceedings of the 7th International Caribbean Conference on Devices, Circuits and Systems*, Mexico, Apr. 28-30, 2008.