A Heuristically Optimized Complex Event Processing Engine for Big Data Stream Analytics

Ignacio (Iñaki) Olabarrieta[1], Ana I. Torre-Bastida[1], Ibai Laña[1], Sergio Campos-Cordobes[1], and Javier Del Ser[1,2,3]

[1] TECNALIA, E-48160 Derio, Spain,
{ignacio.olabarrieta,isabel.torre,ibai.lana}@tecnalia.com
{sergio.campos,javier.delser}@tecnalia.com
[2] University of the Basque Country UPV/EHU, 48013 Bilbao, Spain
[3] Basque Center for Applied Mathematics (BCAM), 48009 Bilbao, Spain

Abstract. This paper describes a Big Data stream analytics platform developed within the DEWI project for processing upcoming events from wireless sensors installed in a truck. The platform consists of a Complex Event Processing (CEP) engine capable of triggering alarms from a predefined set of rules. In general these rules are characterized by multiple parameters, for which finding their optimal value usually yields a challenging task. In this paper we explain a methodology based on a meta-heuristic solver that is used as a wrapper to obtain optimal parametric rules for the CEP engine. In particular this approach optimizes CEP rules through the refinement of the parameters controlling their behavior based on an alarm detection improvement criterion. As a result the proposed scheme retrieves the rules parameterized in a detection-optimal fashion. Results for a certain use case – i.e. fuel level of the vehicle – are discussed towards assessing the performance gains provided by our method.

Keywords: Complex Event Processing; Big Data; Optimization

1 Introduction

Wireless Sensor Networks (WSN) have become an ubiquitous way of getting all sorts of information in many application scenarios. Among the myriad of research initiatives around technological advances on WSN, the DEWI (Dependable Embedded Wireless Infrastructure) project aims to provide key solutions for wireless seamless connectivity and interoperability in smart cities and infrastructures by considering everyday physical environments of citizens in buildings, cars, trains and airplanes [1].

One of the specific industrial domains of the DEWI project is the automotive realm, with an emphasis on trucks. There are at least two main reasons for incorporating wireless sensors within this specific class of vehicles: the fact that there are no wires to connect imply savings in terms of installation time and cost of cabling. Moreover, the replacement of wireless devices are performed in a much simpler, less invasive manner that their wired counterparts. Indeed many studies highlight the key impact that the extensive use of WSN in the automotive realm can imprint on safety, maintenance, and energy efficiency of the vehicle [11].

This paper focuses on a platform developed within the above project for analyzing upcoming data streams from these sensors. Such a platform acquires and analyzes data from fuel, water level, electric suspension and brake lining wear captured by means of wireless sensors deployed at the corresponding parts of the vehicle. The information from the GPS device installed in the vehicle is also sent to the platform. From the perspective of computation and processing the treatment of large volumes of data streams (analyzing Big Data in motion) is nowadays an utmost necessity, and a large number of technologies have emerged to carry out this task (e.g. [9]). The analysis performed at the platform side consists of triggering different alarms when the level exceeds specific thresholds, when indicating a possible fuel theft from the truck or when the truck rides over a bumpy road. In this context the technology selected for alarm detection in the DEWI platform is one of the dominant technologies for stream processing, Complex Event Processing. Unfortunately, the challenge underlying this specific rule-based technology for stream processing is the appropriate configuration of its compounding rules, which should ensure the efficient and accurate detection of alarms in the vehicle.

There are multiple papers in the literature which aim to optimize CEP technologies in different aspects such as efficiency [14], performance [12], features [7, 13, 5] or special application fields [10]. In order to settle an argument for our study, we build on two of the above related works: [12] (complex event processing queries over real-time RFID streams events) and [14] (improvement of pattern queries performance). Both relate to the improvement of CEP engine efficiency and performance, but in our case the optimization is based on an improvement of the CEP rules and not on enhancing the detection or event pattern matching optimization techniques themselves. There are also scarce references dealing with the application of CEP techniques to vehicular scenarios [6]. However, interestingly within the scope of this manuscript to the knowledge of the authors there is no baseline literature where meta-heuristic algorithms are applied to improve the event detection rules of CEP technologies or to optimize the CEP engine itself. This paper covers this research gap by explaining a methodology for finding optimal rules based on already acquired data that have been properly labeled as alarm or not alarm by a technician using the DEWI platform. The method is based on the Harmony Search (HS) algorithm [8], a meta-heuristic optimization procedure that permits to efficiently seek optimal parameters for the rules. We will show how this nature-inspired solver can efficiently tackle the optimization of CEP rules by exploring its performance when utilized for detecting sudden drops in the fuel level of the monitored vehicle.

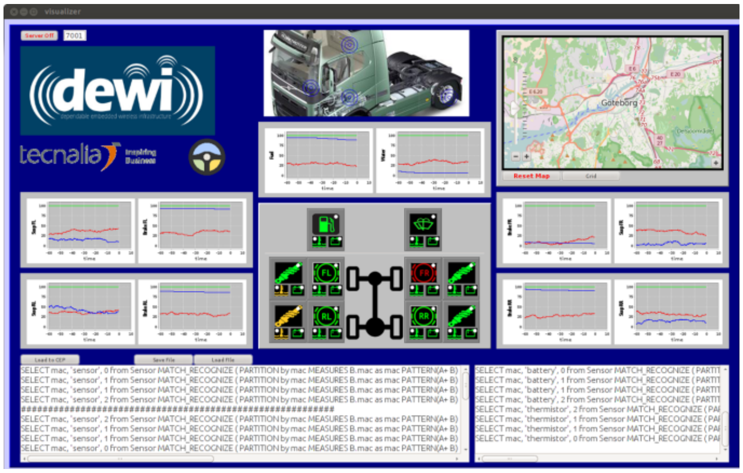
The rest of the paper is organized as follows: Section 2 provides an overview of the platform by explaining its compounding modules: the wireless network within the truck, the data acquiring platform and its Complex Event Processing (CEP) engine. Section 3 reviews the toy use cases and explain their implementation over CEP engines. The rule optimization methodology is explained in detail in Section 4. Finally, we present and discuss the results in Section 5, followed by Section 6 which concludes the work.

2 Platform Overview

The platform consists of three different elements: 1) the WSN, 2) the semantic middleware that receives the data from the sensors, and 3) the *Complex Event Processing*

(CEP) functionality, which analyzes their captured information, and that is integrated with a sub module dedicated to the optimization of the definition of the rules. For testing purposes a simulator has been also developed to emulate output data from WSN. The output from the Complex Event Processing, namely, the computed alarms, is visualized in a tool specially deployed for this purpose. The connection from the WSN to the semantic middleware is currently done via TCP/IP connections, but other protocols can be potentially supported.

We assume that the WSN is deployed on the tractor unit of a Volvo FH-16 truck, whose trailer is not sensorized anyhow. Each of the deployed sensors measures a particular parameter for which it has been specially designed, along with their battery level and temperature. The specific signals that are measured by this mesh of sensors are brake lining wear (4 sensors), electronically controlled suspension level (another set of 4 sensors), the fuel tank level and the washer fluid level. All sensors label their measured samples with a timestamp, the last two bytes of the MAC address of the sensor for identification purposes, other sensor specific measurements, the voltage level of its battery and the value of its thermistor.



```
SELECT mac, 'sensor', 0 from Sensor MATCH_RECOGNIZE ( PARTITION by mac MEASURES B.mac as mac PATTERN(A+ B)
SELECT mac, 'sensor', 1 from Sensor MATCH_RECOGNIZE ( PARTITION by mac MEASURES B.mac as mac PATTERN(A+ B)
SELECT mac, 'sensor', 1 from Sensor MATCH_RECOGNIZE ( PARTITION by mac MEASURES B.mac as mac PATTERN(A+ B)
SELECT mac, 'sensor', 2 from Sensor MATCH_RECOGNIZE ( PARTITION by mac MEASURES B.mac as mac PATTERN(A+ B)
##############################################################
SELECT mac, 'sensor', 2 from Sensor MATCH_RECOGNIZE ( PARTITION by mac MEASURES B.mac as mac PATTERN(A+ B)
SELECT mac, 'sensor', 1 from Sensor MATCH_RECOGNIZE ( PARTITION by mac MEASURES B.mac as mac PATTERN(A+ B)
SELECT mac, 'sensor', 1 from Sensor MATCH_RECOGNIZE ( PARTITION by mac MEASURES B.mac as mac PATTERN(A+ B)
SELECT mac, 'sensor', 0 from Sensor MATCH_RECOGNIZE ( PARTITION by mac MEASURES B.mac as mac PATTERN(A+ B)
```

```
SELECT mac, 'battery', 0 from Sensor MATCH_RECOGNIZE ( PARTIT
SELECT mac, 'battery', 1 from Sensor MATCH_RECOGNIZE ( PARTIT
SELECT mac, 'battery', 1 from Sensor MATCH_RECOGNIZE ( PARTIT
SELECT mac, 'battery', 2 from Sensor MATCH_RECOGNIZE ( PARTIT
SELECT mac, 'thermistor', 2 from Sensor MATCH_RECOGNIZE ( PAI
SELECT mac, 'thermistor', 1 from Sensor MATCH_RECOGNIZE ( PAI
SELECT mac, 'thermistor', 1 from Sensor MATCH_RECOGNIZE ( PAI
SELECT mac, 'thermistor', 0 from Sensor MATCH_RECOGNIZE ( PAI
```

Fig. 1. Snapshot of the application tool to visualize the data from the sensors and the alarms. The visualization area has several areas: on the top right there is a map to visualize the current position of the truck, at the center there are icons denoting the level of alarm for each of the sensors, including the level of its battery and thermistor, around the center display the values acquired. Finally at the bottom of the visualization tool there is a text area where new EPL-coded rules can be added to the CEP.

Aiming to assess the processing capabilities of the platform, a simulator has been developed to replicate the behavior of these sensors and the GPS signal of the truck in different scenarios. One of the scenarios emulates the normal use of the truck in the Göteborg area in which the fuel level, the washer fluid level and the brake lining wear decrease gradually over time, while the suspension sensor have small random variations. Other scenario that can be simulated by means of the developed simulator is the theft of fuel from the tank when the truck is stopped. This effect is replicated by a sudden reduction of the tank level. The truck driving through a rough road with uneven pavement is emulated by producing a high frequency and high amplitude random noise added independently to each of the sensors attached to the suspension.

The CEP engine used in this project has been developed using the EsperTech libraries [2]. Some of the advantages of using a complex event processing rely on its stream event management capabilities, the fact that it can analyze a massive amount of information really fast (e.g. we have been able to analyze the information arriving from over 10^5 simulated sensors per second) and without requiring to store any information in a database. Other appreciated characteristics of CEP are its scalability and a rich declarative language for behavioral configuration.

The CEP is configured to detect different alarms and to forward them upon being triggered to a visualization tool for their output and inspection; a snapshot of the WSN monitoring system to visualize these alarms can be seen in Fig. 1. In order to interpret these alarms properly by the visualizer they are all defined to have a standardize data structure. The alarm is composed by two strings and an integer within the range $\mathbb{N}[0, 6]$. The first string denotes the last two bytes of the MAC address associated with the sensor that triggers the alarm, the second string denotes the quantity associated to the alarm (three have been considered: “sensor”, “thermistor” and “battery”), and finally an integer value denoting the level of alarm. The implementation of the rules which characterize the alarms defined in the previous paragraph is performed via the so-called Event Processing Language (EPL), which is a standard SQL language with extensions in order to provide aggregating functions, pattern matching, event windowing and joining.

Finally, the platform includes a labeling tool that allows tagging captured data. With the help of this tool a human technician can check, evaluate and properly review previously acquired data. These labels are crucial for the heuristic process described in Section 4.

3 Use Cases

We have considered two different use cases in order to test the ability of the platform to capture and analyze data and to assess the goodness of our rules optimization methodology. In the rest of this section we explain in detail the EPL rules for each of the scenarios that we want to detect. These involve detecting any fuel theft, when the truck drives along a rough road or when any of the sensor values changes above or below predefined thresholds. All the rules that we present here select the structure of the alarm that we mentioned above, two strings and an integer value, so they can be interpreted by the visualizer.

3.1 Fuel Theft Detection

The detection of fuel theft is performed by the following EPL rule:

```
SELECT mac, 'sensor', 4
FROM Sensor(mac="AAA8").win:length(10)
HAVING
  max(sensorValue)-min(sensorValue) > 10 & sensorValue < avg(sensorValue)
```

This rule filters out the information arriving to the stream of data Sensor by the MAC of the fuel sensor (i.e. MAC = "AAA8") and considers a data window of 10 measurements. The alarm is triggered in this case if the difference between the maximum and minimum values attained in the window is larger than ten units and the last measurement is less than the average within the window. Basically this rule tries to capture a sudden variation in the fuel level while the general trend is decreasing.

3.2 Bumpy Road Detection

In order to detect a road in bad shape the EPL rule reads:

```
SELECT mac, 'sensor', 5
FROM Sensor
(mac in("AAA4","AAA5","AAA6","AAA7")).
win:time(5 sec)
HAVING
(stddev(sensorValue,mac="AAA4") > 10 |
 stddev(sensorValue,mac="AAA5") > 10 |
 stddev(sensorValue,mac="AAA6") > 10 |
 stddev(sensorValue,mac="AAA7") > 10)
```

In this case the MAC addresses that are considered are the ones of the electronic controlled suspensions and what is measured is the standard deviation within a window of 5 seconds of duration. If the standard deviation for any of those sensors exceeds 10 units the alarm is deactivated.

4 Rule Optimization Procedure

The data flow entering the platform is defined by $F = \{f_j\}$, with $j \in (-\infty, ..., 0]$ and $f_j \in \mathbb{R}$ denoting the sample obtained at time index j. We consider j to be negative, the last obtained sample having index $j = 0$. In general the functions implemented in the CEP engine are applied not to all the history of data but to a limited time window. At a given time index j the time window w_j considered is defined by the measurements $\{f_i|_j \equiv f_{j-i}\}$ with $i \in [0, ..., W]$. We also assume that the human operator has labeled the inflow of data with its correspondent alarms $\{H_j\}$ such that:

$$H_j = \begin{cases} 1 & \text{if alarm should have been deactivated at time } j, \\ 0 & \text{else.} \end{cases} \qquad (1)$$

Having a set of previously obtained data adequately labeled by $\{H_j\}$, and a parametric family of rules, the aim of our study is to design a heuristic procedure to obtain the parameters that fit best the alarms labeled by the operator. The method presented here relies on the Harmony Search (HS) algorithm. The underlying philosophy behind the general class of bio-inspired optimization algorithms (to which HS belongs) is to modify the group of candidates by operators that mimic processes observed in Nature. In these techniques candidate solutions compete, combine, mutate and evolve towards regions of the solution space of progressively increased optimality [4]. HS [8] is encompassed in this broad field of bio-inspired optimization, and is motivated by the idea of achieving harmony within the musical improvisation process undertaken by jazz bands.

Harmonies represent sets of variables to optimize, whereas the aesthetic quality of the improvised harmony is given by the fitness function of the optimization problem at hand. In the rest of this section we provide two particular examples, two different families of rules, of the method applied to the fuel theft use case.

4.1 2-Parameter Rule

The first example considered in this work is an extension of the rule introduced in Section 3.1 with 2 parameters. Specifically we consider the rule $\mathcal{H}_j$ at time index j as:

$$\mathcal{H}_j\left(\{f_j\}\right) = \begin{cases} 1 & \text{if } (\max_{i \in W_j}\{f_i\} - \min_{i \in W_j}\{f_i\} > \mu) \text{ and } f_j < \beta \langle f_i \rangle_{i \in w_j}, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $\mu \in \mathbb{R}[0, \max\{f_j\})$ and $\beta \in (0, ..., 1]$ are the parameters to be optimized. We assume the processing window length W_j to be fixed and equal to 10. Note that the rule introduced in Section 3.1 fits into this family with $\beta = 10$ and $\mu = 1$. The optimization process of this rule relies on the maximization of the F_1 score [3]:

$$F_1 \triangleq 2\,(\mathcal{P} \cdot \mathcal{R})/(\mathcal{P} + \mathcal{R}) \tag{3}$$

where $\mathcal{P}$ and $\mathcal{R}$ are the precision and the recall given by

$$\mathcal{P} = t_p/(t_p + f_p), \mathcal{R} = t_p/(t_p + f_n) \tag{4}$$

with t_p being the number of true positives (alarms that have been properly captured by the rule), f_p the false positives (number of times that the rule has predicted an alarm erroneously) and f_n is the number of false negatives (ratio of occurrences when the rule does not declares any alarm and there is one). These can be expressed as:

$$t_p = \left|\left\{\mathcal{H}_j = 1 \;\&\; H_j = 1\right\}\right|,\ f_p = \left|\left\{\mathcal{H}_j = 1 \;\&\; H_j = 0\right\}\right|,\ f_n = \left|\left\{\mathcal{H}_j = 0 \;\&\; H_j = 1\right\}\right|, \tag{5}$$

where $|\cdot|$ denotes set cardinality. The F_1 score can be thought of as a weighted average of $\mathcal{P}$ and $\mathcal{R}$. Alternatively, depending on the particular application of the rule, a different fitness function can be used such as the maximization of the precision $\mathcal{P}$, the minimization of the recall $\mathcal{R}$ or any other metric suited to the detection requirements of the specific scenario under scope.

4.2 Pattern Correlation Rule

We will consider a different family of rules in our second example. In this case the aim is to compare the incoming flux of data F with a pattern $\mathcal{A}(t)$ modeled as

$$\mathcal{A}(t) = \sum_{i=0}^{W} \alpha_i \cdot \delta\,(t - i \cdot \varDelta t)\,, \tag{6}$$

where $\delta(t)$ is the Dirac's delta function and we assume that the number W of parameters α_i agrees with the size of the CEP processing window. We will consider $\alpha_i \in [0, ..., 1]$, therefore $\mathcal{A}(t) \in [0, ..., 1]$. The rule family can be written in the following way:

$$\mathcal{H}_j = \begin{cases} 1, & \text{if } W^{-1} \sum_{i \in w_j} \left(\tilde{f}_i|_j - \alpha_i \right)^2 \leqslant \epsilon \\ 0, & \text{otherwise,} \end{cases} \tag{7}$$

where ϵ has an small value. In this case $\mathcal{H}_j$ fires and alarm whenever the pattern function $\mathcal{A}$ is close to the transformed incoming flux $\{\tilde{f}_i\}_{i \in w_j}$. We consider that $\{\tilde{f}_i\}_{i \in w_j}$ are related to $\{f_i\}_{i \in w_j}$ by the transformation:

$$\tilde{f}_i|_j = \begin{cases} a f_i|_j + b & \text{if} \quad (1/a) \triangleq \max_{i \in w_j}(\{f_i\}) - \min_{i \in w_j}(\{f_i\}) > \mu, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

with $b = -a \cdot \min_{i \in w_j}(\{f_i\})$. This transformation is realized in order to normalize the incoming flux to the range $[0, ...1]$ and therefore to remove the dependence on the actual range of F. In order to accommodate small disturbances in the incoming flux we have included a parameter $\mu > 0$. The rule defined by Expression (7) compares the shape of the incoming flux with the shape of the pattern $\mathcal{A}$ as long as the variation of F is larger than μ in a particular processing window.

As it is explained in Section 4.1, we could fix ϵ to a particular value and use the F_1 score as the maximization criterion to find the pattern function parameters $\{\alpha_i\}$. Instead we choose to find $\{\alpha_i\}$ by minimizing the distance of $\mathcal{A}$ to the transformed incoming flux whenever the alarm should be deactivated. The resulting fitness function to be minimized can be written as

$$\mathcal{E}(\{\alpha_i\}) = \sum_j H_j \sum_{i \in w_j} \left(\tilde{f}_i|_j - \alpha_i \right)^2 . \tag{9}$$

The benefit of using this particular equation for the fitness is that as a byproduct the befitted $\bar{\epsilon}$ to be used in Expression (7) can be estimated by computing

$$\bar{\epsilon} = \max_j \left\{ H_j \sum_{i \in w_j} \left(\tilde{f}_i|_j - \bar{\alpha}_i \right)^2 \right\}, \tag{10}$$

where $\bar{\alpha}_i$ are the values that minimize (9). Note that, in general, by minimizing (9) we minimize $\bar{\epsilon}$. On the other hand (9) has a remarkable caveat: it does not consider the behavior when $H_j = 0$, which implies that if the pattern $\mathcal{A}$ realized by $\{\alpha_i\}$ is found in the data when $H_j = 0$, $\mathcal{E}$ will not penalize those values. In other words f_p is not considered at all in finding $\{\bar{\alpha}_i\}$.

5 Experiments and Results

In order to test the methodology we have considered the fuel level data shown in Fig. 2. The data represents approximately 2600 captures of the level of the fuel tank. For $j \in [-2600, -1700]$ normal operation with a small decrement of the level is shown. At

$j \approx -1700$ and ≈ -1650 there are two sudden reductions which are labeled by an alarm (in red), after the second reduction again normal operation occurs. Two more alarms are labeled happening at $j \approx -900$ and ≈ -400. Refueling can be noted at $j \approx -1300$ and ≈ -350.

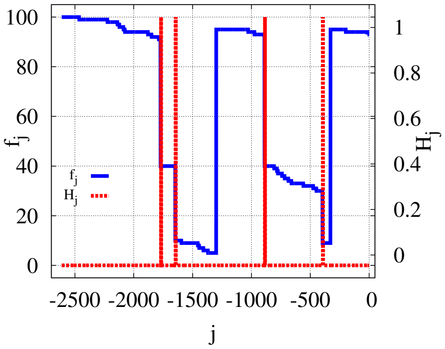


Fig. 2. Data obtained from the wireless fuel sensor f_j and the labels H_j denoting the fuel theft events. In these data 4 alarms can be distinguished. While a sudden decrease of the level of fuel occurs, these correspond to fuel theft events. In between the normal usage, small decrements of the level by the normal use of the truck can be noted plus two instances when the tank is refueled.

5.1 2-Parameter Rule

We have used the previous data in order to obtain the best parameters β and μ. An example of how the HS optimization process converges is shown in the left side of Fig. 3 where it can be observed how the fitness $\mathcal{E}$ goes to 1 as the iterative process evolve and the parameters reach some values in the solution space. One of things that is worth mentioning is the fact that there is no single solution in the solution space, actually we found that there exits a range of solution for this particular problem. The HS optimization process obtains a different solution depending on the initial random values choose for the parameters. In order to obtain a sense of the solution space we have performed thousands of HS optimization varying randomly the initial parameters. The region of the solution space can be hinted in the right subplot of Fig. 3 $\beta \times \mu \in \mathbb{R}(0.887, 0.99994) \times \mathbb{R}(1.003, 20.9993)$. The average number of iterations to attain a fitness of 1.0 is equal to 60 ± 1.

5.2 Pattern Correlation Rule

In this case we use the same set of data to optimize the fitness derived for the pattern correlation rule. If we assume that in normal operation the level of the fuel does not decrease, the optimal solution corresponding to decreasing jumps in the fluid level corresponds to $\{\alpha_i\}_{i=0}^{3} = 1$ and $\{\alpha_i\}_{i=4}^{7} = 0$. Despite unrealistic it is logical to assume that the parameters for a real solution (meaning that in normal operation the fuel decreases smoothly) approach these values. The results for a single solution are presented in Fig. 4. Note that the HS optimization process takes a significant larger amount of iterations

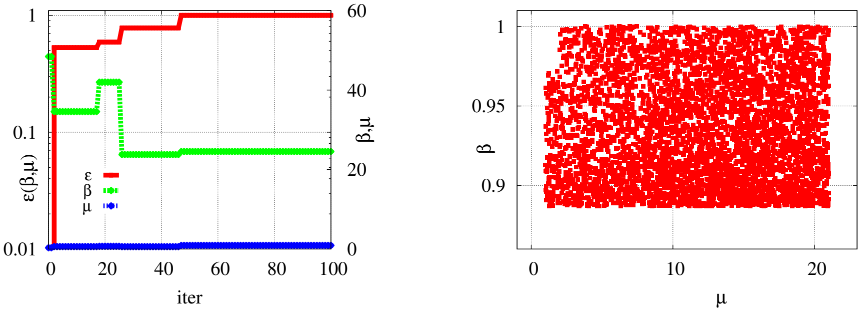


Fig. 3. On the left, an example of how the fitness $\mathcal{E}\{\beta,\mu\}$ converge to 1 (in red against the left axis) and the parameters β and μ (green and blue against the right axis) reach a point in the solution space. For a given set of training data the actual values obtained for the parameters depend on the initial seed used for the HS optimization process. On the right hand side there is a collection of solutions in the $\mu-\beta$ space giving hints of the range of the solution space.

to converge, ∼ 8000 iterations to reduce the fitness 3 orders of magnitude. The actual values that we found for $\{\alpha_i\}$ corresponds to:

$$\{\alpha_i\}_{i=0}^{7} = \{0.976, 0.995, 0.968, 0.963, 0.991, 0.050, 0.040, 0.041, 0.074, 0.026\}, \quad (11)$$

which qualitatively agree with the logic inferred from the ideal solution.

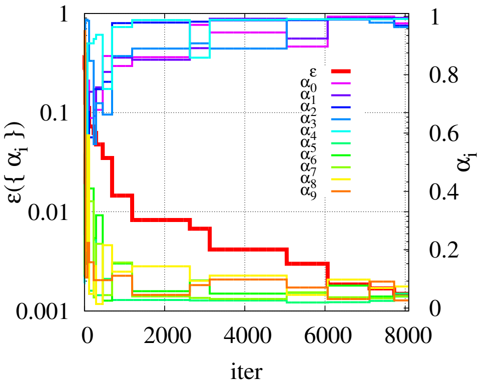


Fig. 4. Convergence of the fitness $\mathcal{E}(\{\alpha_i\})$ (in red and against the left axis) to 0 and the $\{\alpha_i\}$ (other colors and against the right axis) to their solution values.

6 Conclusion

In this paper we have presented a platform that enables analyzing data originated in sensors within a truck tractor unit for the fuel level, the washer fluid level, the suspension and the brake lining wear. The platform is capable of detecting different complex alarms as proven using the data streamed by the simulator for several toy examples. A procedure to find optimal parametric rules based on HS optimization has been developed which has been tested for the fuel theft use case using two different families of parametric rules and different fitness functions, proving itself useful.

The next step within the project involves completing the connection of the actual sensors installed within the truck to the platform. Moreover a key ingredient in working with real signals will be to analyze the real signals and to develop machine learning algorithms capable of the finding the specific patterns which characterize the scenarios that want to be detected. Steps are already being taken to complete the overall system.

Acknowledgments

The research from DEWI project (www.dewi-project.eu) [1] leading to these results has received funding from ARTEMIS Joint Undertaking under grant agreement no.621353. The authors want to specially thank Parthasarathy Dhasarathy, from Volvo Technology AB, who helped in the use cases and the platform infrastructure definition.

References

1. Dewi project. http://www.dewiproject.eu, accessed: 15-10-2016
2. Espertech. http://www.espertech.com/, accessed: 15-10-2016
3. F1 score. https://en.wikipedia.org/wiki/F1_score, accessed: 15-11-2016
4. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evolutionary computation 1(1), 1–23 (1993)
5. Björne, J., Heimonen, J., Ginter, F., Airola, A., Pahikkala, T., Salakoski, T.: Extracting complex biological events with rich graph-based feature sets. In: Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task. pp. 10–18. Association for Computational Linguistics (2009)
6. Bruns, R., Dunkel, J., Billhardt, H., Lujak, M., Ossowski, S.: Using complex event processing to support data fusion for ambulance coordination. In: Information Fusion (FUSION), 2014 17th International Conference on. pp. 1–7. IEEE (2014)
7. Ding, L., Chen, S., Rundensteiner, E.A., Tatemura, J., Hsiung, W.P., Candan, K.S.: Runtime semantic query optimization for event stream processing. In: 2008 IEEE 24th International Conference on Data Engineering. pp. 676–685. IEEE (2008)
8. Geem, Z.W., Kim, J.H., Loganathan, G.: A new heuristic optimization algorithm: harmony search. Simulation 76(2), 60–68 (2001)
9. Hirzel, M., Andrade, H., Gedik, B., Jacques-Silva, G., Khandekar, R., Kumar, V., Mendell, M., Nasgaard, H., Schneider, S., Soulé, R., et al.: Ibm streams processing language: Analyzing big data in motion. IBM Journal of Research and Development 57(3/4), 7–1 (2013)
10. Liu, H.L., Chen, Q., Li, Z.H.: Optimization techniques for rfid complex event processing. Journal of computer science and technology 24(4), 723–733 (2009)
11. Lu, N., Cheng, N., Zhang, N., Shen, X., Mark, J.W.: Connected vehicles: Solutions and challenges. IEEE internet of things journal 1(4), 289–299 (2014)
12. Wu, E., Diao, Y., Rizvi, S.: High-performance complex event processing over streams. In: Proceedings of the 2006 ACM SIGMOD international conference on Management of data. pp. 407–418. ACM (2006)
13. Yan, Y., Yang, Y., Meng, D., Liu, G., Tong, W., Hauptmann, A.G., Sebe, N.: Event oriented dictionary learning for complex event detection. IEEE Transactions on Image Processing 24(6), 1867–1878 (2015)
14. Zhang, H., Diao, Y., Immerman, N.: On complexity and optimization of expensive queries in complex event processing. In: Proceedings of the 2014 ACM SIGMOD international conference on Management of data. pp. 217–228. ACM (2014)