Classifiers for Behavioral Patterns Identification Induced from Huge Temporal Data

Jan G. Bazan[1], Marcin Szpyrka[2,1], Adam Szczur[1], Łukasz Dydo[1], and Hubert Wojtowicz[1]

[1] Interdisciplinary Centre for Computational Modelling
University of Rzeszów
Pigonia 1, 35 - 310 Rzeszów, Poland
{bazan,ldydo}@ur.edu.pl,
{adamszczur8,hubert.wojtowicz}@gmail.com
[2] AGH University of Science and Technology
Department of Applied Computer Science
Mickiewicza 30, 30-059 Kraków, Poland
mszpyrka@agh.edu.pl

Abstract. A new method of constructing classifiers from huge volume of temporal data is proposed in the paper. The novelty of introduced method lies in a multi-stage approach to constructing hierarchical classifiers that combines process mining, feature extraction based on temporal patterns and constructing classifiers based on a decision tree. Such an approach seems to be practical when dealing with huge volume of temporal data. As a proof of concept a system has been constructed for packet-based network traffic anomaly detection, where anomalies are represented by spatio-temporal complex concepts and called by behavioral patterns. Hierarchical classifiers constructed with the new approach turned out to be better than ”flat” classifiers based directly on captured network traffic data.

Keywords: classifiers, huge temporal data, temporal patterns, state graphs, behavioral patterns, LTL temporal logic

1 Introduction

Classifiers (*decision algorithms*) constitute the kernel of *decision systems* that are ubiquitous in many areas of IT systems like data mining, knowledge discovery, expert systems etc. [12], [15]. There are numerous approaches to constructing classifiers to be found in literature. Due to the growth of volume of gathered data and complexity of analyzed concepts new methods of data mining, process mining and classifiers constructing are needed to meet the challenge of nowadays applications. Particularly, data more and more often concern complex processes which do not give in to classical modeling methods. Examples of such data include medical and financial data, data from vehicles monitoring, or data from telecommunication networks, e.g. information about packages flow. Methods of exploring such data are the center of attention of many powerful research centers in the world, and at the same time detection of models of complex processes and their properties (patterns) from data is becoming more and more attractive for applications [1], [9], [14], [17].

Making progress in this field is extremely crucial, among other things, for the development of intelligent systems processing huge volume of data. Therefore, developing methods of detecting process models and their properties from data and proving their effectiveness in different applications are of particular importance for further development of decision supporting systems in many domains such as medicine, finance, industry, transport, telecommunication, and others.

The paper deals with a problem of process mining and constructing classifiers from huge volume of temporal data. The presented approach combines automatic methods of detecting processes and their properties with domain knowledge obtained from experts. Interaction with domain experts facilitates guiding the process of discovering patterns and models of processes and makes the process computationally feasible. The novelty of introduced method lies in a multi-stage approach to constructing hierarchical classifiers that incorporates:

- a process mining – during the learning stage data are grouped and represented as a state graph, which reduces the data size significantly;
- a features extraction based on temporal patterns – which takes the form of LTL temporal logic formulas, may be defined by experts or discovered from the learning data;
- a classifier construction – a classifier is constructed with temporal patterns identified from the learning data i.e. it is based on information about presence or absence of individual behavioral patterns in the analyzed data.

The paper is organized as follows. Section 2 presents an overview of the considered approach to classifiers construction. Classifiers based on decision trees are described shortly in Section 3. Section 4 deals with a system for packet-based network traffic anomaly detection implemented with the new approach introduced in the paper. A short summary is given in the final section.

2 Constructing of hierarchical classifiers

The general scheme of constructing of hierarchical classifiers is given in Fig. 1. Let $A = \{a_1, \dots, a_k\}$ denote a set of attributes selected to describe important features of the system under consideration. The starting point of the presented approach are data describing values of the attributes in a sequence of time points $t_1, t_2, \dots$. These learning data can be presented in a table as shown in Fig. 2. The data are considered from two points of view. In the first stage the whole input data are used to construct the so-called *state graph*. This is the process mining stage. In the consecutive stages, we use the learning data sliced into pieces called *time windows*. In the network traffic anomaly detection system presented in Section 4 a constant length of time windows has been used. However, this limitation can be omitted and in general changeable time window length can be used.

The process mining stage concerns building a *state graph*. However, before the process mining is started, rows from the learning data table (called here *time points*) are grouped with a metric that describes the similarity (distance) between time points. In this paper we use a well-known in literature k-means method of clustering. The

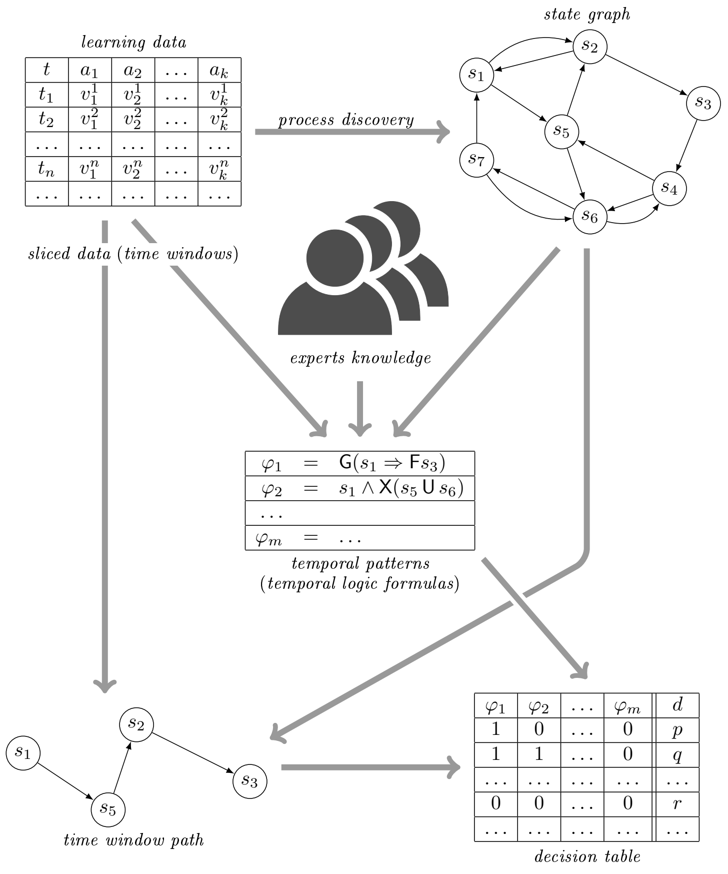
state graph
learning data
process discovery
sliced data (time windows)
experts knowledge
$\varphi_1 = \mathsf{G}(s_1 \Rightarrow \mathsf{F}s_3)$
$\varphi_2 = s_1 \wedge \mathsf{X}(s_5 \,\mathsf{U}\, s_6)$
$\varphi_m = \ldots$
temporal patterns
(temporal logic formulas)
time window path
decision table

Fig. 1. Scheme of constructing of hierarchical classifiers

t	a_1	a_2	$\ldots$	a_k	
t_1	v_1^1	v_2^1	$\ldots$	v_k^1	time window 1
t_2	v_1^2	v_2^2	$\ldots$	v_k^2	
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	
t_p	v_1^p	v_2^p	$\ldots$	v_k^p	
t_{p+1}	v_1^{p+1}	v_2^{p+1}	$\ldots$	v_k^{p+1}	time window 2
t_{p+2}	v_1^{p+2}	v_2^{p+2}	$\ldots$	v_k^{p+2}	
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	
t_{2p}	v_1^{2p}	v_2^{2p}	$\ldots$	v_k^{2p}	
t_{2p+1}	v_1^{2p+1}	v_2^{2p+1}	$\ldots$	v_k^{2p+1}	
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	

Fig. 2. General scheme of input data

metric is based on time point attributes. Our method of process mining works on the data after clustering, i.e., on the data that can be represented by a sequence of groups (clusters). For such data we use the method from [2]. As a result, each group of time points obtained from clustering process is represented as a node in the state graph. If two consecutive time points belong to two different groups an arc is included into the graph that connects corresponding nodes. Multiple arcs going from node s_i to s_j are represented by a single arc. A state graph is generated using all learning data. The size of the graph is crucial for the next stage, so the metric function should be adjusted in order to reduce the state graph size if it is too complex. This stage allows us to cope with huge amount of time points and makes the approach scalable.

Temporal patterns take the form of temporal logic formulas. LTL [3], [11] logic has been used for experiments described in the paper. Aside from the propositional logic operators, the temporal operators G (globally), F (finally), X (next), U (until) can be used in LTL. Temporal patterns represent some temporal dependencies between nodes of the state graph e.g. presence or absence of some nodes (clusters) in a path, nodes order in a path etc. In case of the network anomaly detection system temporal patterns have been defined by experts using the generated state graph. A module for automatic patterns extraction will be developed in the future. Let $\Phi = \{\varphi_1, \ldots, \varphi_m\}$ denote the set of temporal patterns. They are treated as input attributes for the classification problem. For a given time window a path is generated and for each behavior pattern it is checked whether the corresponding LTL formula holds for the path or not. Thus for a given path a sequence of m Boolean values is evaluated. Due to the fact that for learning data the values of the decision attributes are known, each time window provides a row for *high level learning data* described by conditional attributes $\varphi_1, \ldots, \varphi_m$ and the decision attribute d. The high level learning data are used for constructing the classifier.

3 Classifier based on a decision tree

The hierarchical classifier considered in the paper is based on the high level learning data described in the previous section. We consider the classifier that is based on the so-called decision tree of the local discretization (see, e.g., [7], [6], [16]). It is a binary tree, created by multiple binary partitions of the set of objects into two groups (e.g., cases, states, processes, patients, observations, vehicles) with the value of a selected attribute. During construction of the binary tree, the method of choosing an attribute and its value (for numeric attributes often called *cut*), that we use in the partition, is a key element of the discussed local discretization tree construction method and should involve the analysis of decision attribute values for training objects. Thus, one of the most important concepts presented in the strategy is a binary partition of the set of objects based on the attribute and its value. Formally, a *cut* is a pair (a, v) that is defined for a given *decision table* $\mathbf{A} = (U, A, d)$ in Pawlak's sense (see, e.g., [18]), where $a \in A$ (A is a set of attributes or columns in the data set) and v is the value of the attribute a that defines the partition of a set of attribute's values into two subsets. For numeric attributes, a cut (a, v) defines a partition of a set of objects into two subsets – the first set is a set of objects for which the a attribute value is less than v, and the second is a set of objects for which the a attribute value is greater than or equal to v.

Meanwhile, for symbolic attributes the first one is a set of objects for which the value of the attribute a is equal to v, and the second set is a set of objects for which the a attribute value is different from v. Moreover, any cut (a, v) defines two templates, where by a template we understand a description of some set of objects. In case of numerical attributes, the first template defined by a cut (a, v) is a formula $T_{(a,v)} = (a(u) < v)$ and the second template defined by a cut (a, v) is a formula $\neg T_{(a,v)} = (a(u) \geq v)$. In case of symbolical attributes, the first template defined by a cut (a, v) is a formula $T_{(a,v)} = (a(u) = v)$ and the second template defined by a cut (a, v) is a formula $\neg T_{(a,v)} = (a(u) \neq v)$.

As a measure of the binary partition quality the number of pairs of objects distinguishable by partition and having different values of the decision attribute can be used. For example, if a partition (a, v) divides objects into two sets of sizes M and N, and the first of these collections have M_0 and M_1 objects from the class C_0 and C_1 respectively, and in the second one we have N_0 and N_1 objects from the decision class C_0 and C_1, then the number of pairs of objects discerned by the partition is given by: $N_1 \cdot M_0 + M_1 \cdot N_0$. If we determine the value of this measure for all possible cuts, then we can choose one of the cuts and divide the entire set of objects into two parts on its basis. Of course, this approach can be easily generalized to the case of more than two decision classes.

It should be noted that this measure of the quality of the binary partition of set of objects can be calculated for the given cut in time $O(n)$, where n is the number of objects in the decision table (see, e.g., [6]). But determining the optimal cut requires the calculation of quality measures for all potential cuts. For this purpose it is necessary to check all potential cuts, including all conditional attributes in a specific order. This can be done with various methods. One of such methods for numerical attributes firstly sorts the objects of the given attribute for which we seek the optimal partition. This allows us to determine the optimal cut. Sorting a collection of objects results in the fact that the calculation of the optimal partition is done in time $O(n \cdot \log\ n \cdot m)$, where n is the number of objects, and m is the number of conditional attributes. This is the method we have implemented in our own computational library RS-lib.

The quality of cuts may be computed for any subset of a given set of objects. In the local strategy of discretization, after finding the best cut and dividing the objects set into two subsets of objects (matching both templates mentioned above for a given cut), this procedure is repeated for each object from the set separately until a stop condition holds. At the beginning of the procedure we have the whole set of objects at the root of the tree. Then, we recursively apply the same splitting procedure to the emerging parts that we assign to tree nodes at higher and higher levels. Stop condition of partition is designed so that the given part is not divided (becomes a leaf tree) if it contains only objects from one decision class (optionally the objects from the given class constitute a certain percentage, which is treated as a parameter of the method) or the considered cut does not have any effect, i.e., there are no new pairs of objects from different decision classes separated by the cut.

In this paper, we assume that the partition stops when all objects from the current set of objects belong to the same decision class. Hence, the local strategy can be realized by using *decision tree* (see Fig. 3).

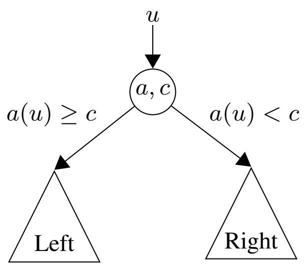

Fig. 3. Decision tree used in local discretization

Besides the number of pairs of objects distinguishable by partition, in our experiments we also use two other measures of the quality of cuts well known in literature that were used in other methods of decision trees construction. It is a measure called *a Gini index* (used in the algorithm CART [10]) and a measure called *an information gain* (entropy; used in the algorithm C4.5 [19]).

The decision tree computed during local discretization can be treated as a classifier for the concept C represented by decision attribute from a given decision table $\mathbf{A}$. Let u be a new object and $\mathbf{A}(T)$ be a subtable containing all objects matching the template T defined by the cut from the current node of a given decision tree (at the beginning of algorithm run T is the template defined by the cut from the root). We classify object u starting from the root of the tree as shown in the algorithm presented in Fig. 4.

```
1: Step 1
2: if $u$ matches template $T$ found for $\boldsymbol{A}$ then
3:     go to subtree related to $\boldsymbol{A}(T)$
4: else
5:     go to subtree related to $\boldsymbol{A}(\neg T)$
6: end if
7: Step 2
8: if $u$ is at the leaf of the tree then
9:     go to Step 3
10: else
11:     repeat 1–2 substituting $\boldsymbol{A}(T))$ or $\boldsymbol{A}(\neg T))$ for $\boldsymbol{A}$
12: end if
13: Step 3
14: Classify $u$ using the decision value attached to the leaf
```

Fig. 4. Classification by decision tree (see [6])

Note that the above decision tree can be treated directly as a classifier, as test objects can be classified by stating to which leaf of the tree they belong. This is possible because, thanks to the designated partitions, one can trace membership of an object in the path from the root to the leaf, and then classify the object to the decision class whose objects dominate in the leaf.

4 Usability studies

As a proof of concept for the approach considered in the paper a system for packet-based network traffic anomaly detection has been constructed. Network anomaly detection is becoming an essential area of research. The growing number of IP networks threats and the growing volume of transmitted data require new methods of network traffic data analysis [8], [13].

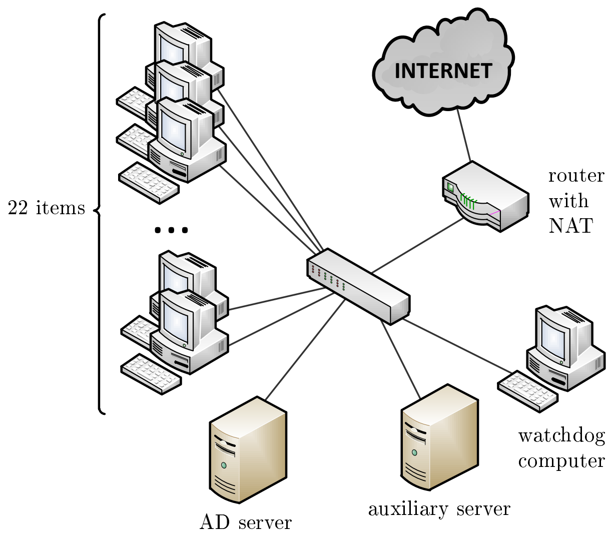

Fig. 5. Network topology

For the purpose of this work, a part of the university network was selected to capture data for analysis. The network topology is given in Fig. 5. The experiment environment consists of 22 work stations with Windows 7 operating system, Active Directory server, a watchdog computer (Windows 7) and auxiliary server with Windows Server 2008 R2 operating system. The NAT router has been used to separate the network from the whole university network and to provide an access to the Internet. The router is equipped with a mirror port used to send copies of all packets to the watchdog computer. The auxiliary server provides FTP (port 21), RDP (port 3389) and MySQL (port 3306) services.

The Wireshark 1.10.7 software was used to monitor the network traffic. It provides the possibility of real time observing of sending and receiving packets for the given interface and to backup them to *pcapng* files. The network traffic was monitored from May 26 to 28 using one of laboratories of the Interdisciplinary Centre for Computational Modelling at University of Rzeszow. The typical network traffic generated by students lessons was captured as the legitimate traffic. Further to that, each day we generated four different network traffic anomalies including *network scan*, *IP-spoofed scanning* and *brute force*. The result of network traffic capturing was three 24-hours data sets in the form of *pcapng* files presented in Table 1.

Table 1. Profile of captured data

Date	Number of packets	Data size
26.05.2014	6265001	3GB
27.05.2014	1236346	1GB
28.05.2014	9488419	4.7GB

Captured data were converted into *csv* files. Received time points were described with attributes presented in Table 2.

Table 2. Data attributes

Attribute name	Description
id	packet identifier
srcIP	source IP
srcPort	source port
destIP	destination IP
destPort	destination port
protocol	protocol
length	packet length (bytes)
time	packet transmission time
relTime	time from starting monitoring
info	short information about packet (from Wireshark)
srcMAC	source MAC address
dstMAC	destination MAC address
deltaTime	time difference between current and previous packet
ipFlags	IP flags
ttl	packet Time To Live
tcpFlags	TCP flags
icmpType	type of ICMP traffic
udpLength	UDP packet length

The metric function defined for the data is based on *ipFlags, length, ttl, tcpFlags, icmpType* and *udpLength* attributes. The distance between objects a_i and a_j is defined as follows:

$$\begin{aligned} D(a_i, a_j) = \frac{1}{6}(d_{ipFlags}(a_i, a_j) + d_{length}(a_i, a_j) + d_{ttl}(a_i, a_j) + \\ + d_{tcpFlags}(a_i, a_j) + d_{icmpType}(a_i, a_j) + d_{udpLength}(a_i, a_j)), \end{aligned} \tag{1}$$

where $D \in [0, 1]$ and

$$d_p(a_i, a_j) = \begin{cases} 0: & p(a_i) = p(a_j), \\ 1: & p(a_i) \neq p(a_j), \end{cases} \tag{2}$$

$$d_q(a_i, a_j) = \begin{cases} 0: & q(a_i) \in X_j \wedge q(a_j) \in X_j, \\ 1: & q(a_i) \in X_j \wedge q(a_j) \in X_k \wedge j \neq j, \end{cases} \tag{3}$$

where equation (2) is used for attributes *ipFlags*, *tcpFlags* and *icmpType*, while equation (3) is used for attributes *length*, *ttl* and *udpLength* using ranges presented in Table 3.

Table 3. Attributes ranges

X_i	*length*	*ttl*	*udpLength*
X_1	0–19		null
X_2	20–39	1–31	1–62
X_3	40–79	32–47	63–125
X_4	80–159	48–63	126–250
X_5	160–319	64–71	251–500
X_6	320–639	72–95	501–1000
X_7	640–1279	96–111	1001–2000
X_8	1280–2559	112-127	2001–4000
X_9	2560–5119	128–143	4001–8000
X_{10}	5120–4294967295	144–159	8000–65535
X_{11}		160–191	
X_{12}		192–255	

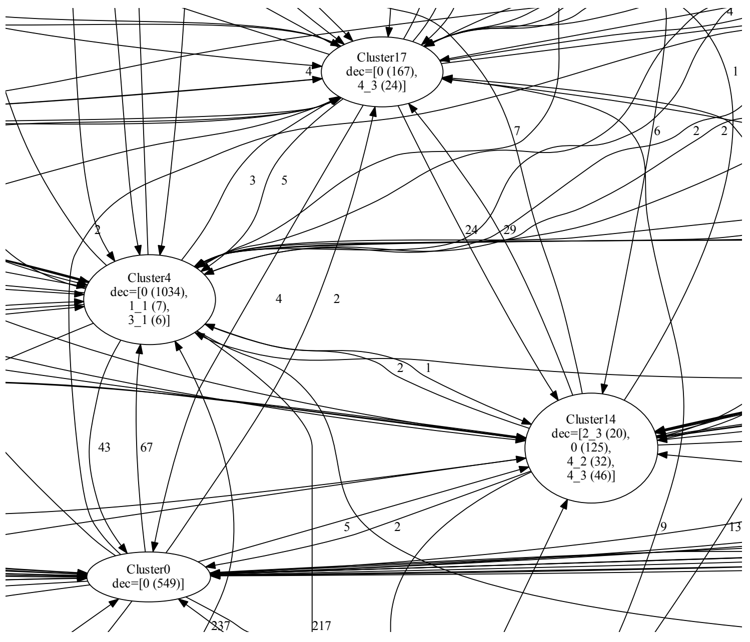

Fig. 6. Part of the state graph

$\mathsf{F}c0$	$\mathsf{F}(c7 \Leftrightarrow \mathsf{X}c10)$	$\mathsf{F}(c17 \Leftrightarrow \mathsf{X}c7)$
$\mathsf{F}c2$	$\mathsf{F}(c7 \Leftrightarrow \mathsf{X}c14)$	$\mathsf{F}(c17 \Leftrightarrow \mathsf{X}c10)$
$\mathsf{F}c3$	$\mathsf{F}(c7 \Leftrightarrow \mathsf{X}c16)$	$\mathsf{F}(c17 \Leftrightarrow \mathsf{X}c14)$
$\mathsf{F}c4$	$\mathsf{F}(c7 \Leftrightarrow \mathsf{X}c17)$	$\mathsf{G}(c1 \Rightarrow \mathsf{X}c8)$
$\mathsf{F}c5$	$\mathsf{F}(c8 \Leftrightarrow \mathsf{X}c1)$	$\mathsf{G}(c7 \Rightarrow \mathsf{X}c16)$
$\mathsf{F}c6$	$\mathsf{F}(c10 \Leftrightarrow \mathsf{X}c7)$	$\mathsf{G}(c7 \Rightarrow \mathsf{X}(c14 \vee c16))$
$\mathsf{F}c7$	$\mathsf{F}(c10 \Leftrightarrow \mathsf{X}c14)$	$\mathsf{G}(c7 \Rightarrow \mathsf{X}(c10 \vee c14 \vee c16 \vee c17))$
$\mathsf{F}c9$	$\mathsf{F}(c10 \Leftrightarrow \mathsf{X}c16)$	$\mathsf{G}(c8 \Rightarrow \mathsf{X}c1)$
$\mathsf{F}c11$	$\mathsf{F}(c10 \Leftrightarrow \mathsf{X}c17)$	$\mathsf{G}(c10 \Rightarrow \mathsf{X}(c7 \vee c14 \vee c16 \vee c17))$
$\mathsf{F}c12$	$\mathsf{F}(c14 \Leftrightarrow \mathsf{X}c7)$	$\mathsf{G}(c14 \Rightarrow \mathsf{X}c7)$
$\mathsf{F}c13$	$\mathsf{F}(c14 \Leftrightarrow \mathsf{X}c10)$	$\mathsf{G}(c14 \Rightarrow \mathsf{X}(c7 \vee c10 \vee c17))$
$\mathsf{F}c15$	$\mathsf{F}(c14 \Leftrightarrow \mathsf{X}c17)$	$\mathsf{G}(c16 \Rightarrow \mathsf{X}c7)$
$\mathsf{F}c18$	$\mathsf{F}(c16 \Leftrightarrow \mathsf{X}c7)$	$\mathsf{G}(c16 \Rightarrow \mathsf{X}(c7 \vee c14))$
$\mathsf{F}c19$	$\mathsf{F}(c16 \Leftrightarrow \mathsf{X}c14)$	$\mathsf{G}(c16 \Rightarrow \mathsf{X}(c7 \vee c14 \vee c17))$
$\mathsf{F}(c1 \Leftrightarrow \mathsf{X}c8)$	$\mathsf{F}(c16 \Leftrightarrow \mathsf{X}c17)$	$\mathsf{G}(c17 \Rightarrow \mathsf{X}(c7 \vee c19 \vee c14))$

Fig. 7. Temporal patterns

Finally, we received the state graph with 20 nodes. The graph is stored in the *dot* format and can be visualized automatically with *xdot* or similar software. Part of the state graph for the considered system is presented in Fig. 6. Let ci denote the i-th node (cluster) in the graph. Based on the expert knowledge the temporal patterns presented in Fig. 7 were used in the system. After a few experiments the time windows length equal to 15 was chosen. Data captured May 26 were used as the learning data, while data captured May 27 were used to check the hierarchical classifier accuracy. The hierarchical classifier was compared with a ”flat” classifier built for learning data without using grouping, state graphs and temporal patterns. Both classifiers (hierarchical and flat) are based on binary trees described in Section 3, but three measures of cut quality were used: pair indiscernibility, entropy and Gini index (see Section 3). The decision attribute takes one of five values 0–4, where 0 denotes the legitimate traffic, while values from 1 to 4 denote four types of network traffic anomalies. It is worth noticing that each of these decision values represents certain spatio-temporal complex concept, which describes the specific behavior of a network user in a certain period of time. This behavior we call here as a behavioral pattern. Besides, the prediction of such a decision value we call an identification of a behavioral pattern. Note that each behavioral pattern can be represented as a subgraph of the state graph (see [4], [5]), but such an approach is not used in this paper.

The results of experiments are given in Table 4, where *class* stands for the *anomaly type* and *cov* for coverage.

Table 4. Results of experiments

	Hierarchical classifier			Flat classifier		
Measure	Class	Accuracy	Cov.	Class	Accuracy	Cov.
Pair indisc.	0	0.956	1.0	0	0.973	1.0
	1	0.0	1.0	1	0.0	1.0
	2	0.005	1.0	2	0.011	1.0
	3	0.972	1.0	3	0.972	1.0
	4	0.989	1.0	4	0.985	1.0
	overall	0.899	1.0	overall	0.755	1.0
Information gain (entropy)	0	0.972	1.0	0	0.941	1.0
	1	0.0	1.0	1	0.067	1.0
	2	0.011	1.0	2	0.017	1.0
	3	0.972	1.0	3	0.204	1.0
	4	0.986	1.0	4	0.100	1.0
	overall	0.913	1.0	overall	0.613	1.0
Gini index	0	0.973	1.0	0	0.964	1.0
	1	0.0	1.0	1	0.067	1.0
	2	0.011	1.0	2	0.017	1.0
	3	0.972	1.0	3	0.204	1.0
	4	0.985	1.0	4	0.035	1.0
	overall	0.913	1.0	overall	0.613	1.0

5 Conclusions

A new approach for constructing classifiers from huge volume of temporal data has been presented in the paper. Hierarchical classifiers considered in the paper combine process mining, extraction of attributes on the basis of temporal patterns and constructing classifiers based on decision trees methods. A system for network traffic anomaly detection has been constructed as a proof of concept. The hierarchical classifier constructed for the anomaly detection system turned out to be better than a classifier built for learning data without using grouping, state graph and temporal patterns. Omitting traffic anomalies classes 1 and 2, where both approaches failed to identify the anomalies, for remaining classes our method was even 30% better. The approach can be used in many domains such as medicine, finance, industry, transport, telecommunication, and others. The network traffic was chosen due to the possibility of capturing huge volume of learning data.

Acknowledgement

This work was partially supported by the Polish National Science Centre grant DEC-2013/09/B/ ST6/01568 and by the Centre for Innovation and Transfer of Natural Sciences and Engineering Knowledge of University of Rzeszów, Poland.

References

1. van der Aalst, W.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer Publishing Company, Incorporated (2011)
2. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '98). pp. 469–483 (1998)
3. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press, London, UK (2008)
4. Bazan, J.G.: Behavioral pattern identification through rough set modeling. Fundamenta Informaticae 72(1-3), 37–50 (2006)
5. Bazan, J.G.: Hierarchical classifiers for complex spatio-temporal concepts. Transactions on Rough Sets 5390(IX), 474–750 (2008)
6. Bazan, J.G., Nguyen, H.S., Nguyen, S.H., Synak, P., Wróblewski, J.: In: Polkowski, L., Lin, T.Y., Tsumoto, S. (eds.) Rough Set Methods and Applications: New Developments in Knowledge Discovery in Information Systems, Studies in Fuzziness and Soft Computing, vol. 56, pp. 49–88. Springer-Verlag/Physica-Verlag, Heidelberg, Germany (2000)
7. Bazan, J., Bazan-Socha, S., Buregwa-Czuma, S., Pardel, P.W., Sokolowska, B.: Predicting the presence of serious coronary artery disease based on 24 hour holter ecg monitoring. In: Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS 2012), September 9-12, Wroclaw, Poland. pp. 279–286 (2012)
8. Berezański, P., Szpyrka, M., Jasiul, B., Mazur, M.: Network anomaly detection using parameterized entropy. In: Proceedings of the 13th International Conference on Computer Information Systems and Industrial Management Applications CISIM 2014. LNCS, Springer-Verlag (2014)
9. Borrett, S., Bridewell, W., Langley, P., Arrigo, K.: A method for representing and developing process models. Ecological Complexity 4(1–2), 1–12 (2007)
10. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Trees. Chapman And Hall/CRC Press, Boca Raton, FL (1984)
11. Clarke, E., Grumberg, O., Peled, D.: Model Checking. The MIT Press, Cambridge, Massachusetts (1999)
12. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference and prediction. Springer, 2 edn. (2008)
13. Jasiul, B., Śliwa, J., Gleba, K., Szpyrka, M.: Identification of malware activities with rules. In: Proceedings of the Federated Conference on Computer Science and Information Systems. Warsaw, Poland (2014)
14. Langley, P.: Cognitive architectures and general intelligent systems. AI Magazine 27, 33–44 (2006)
15. Maimon, O., Rokach, L.: Data Mining and Knowledge Discovery Handbook. Springer-Verlag, Secaucus, NJ, USA (2005)
16. Nguyen, H.S.: Approximate boolean reasoning: Foundations and applications in data mining. LNCS Transactions on Rough Sets V 4100, 334–506 (2006)
17. Pancerz, K., Suraj, Z.: Discovery of asynchronous concurrent models from experimental tables. Fundamenta Informaticae 61(2), 97–116 (2003)
18. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Sciences 177, 3–27 (2007)
19. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Francisco, CA (1992)