ANN-based secure task scheduling in computational clouds

Jacek Tchórzewski
AGH University of Science and Technology
30-059 Cracow, Poland
Cracow University of Technology
31-155 Cracow, Poland

Ana Respício
CMAFCIO
Faculdade de Ciências
Universidade de Lisboa
1749-016 Lisboa, Portugal

Joanna Kołodziej
Research and Academic Computer Network (NASK)
Kolska st 12, 01-045 Warsaw, Poland

KEYWORDS

Computational Clouds; Cloud Security; Tasks Scheduling; Artificial Neural Networks; Intelligent Security System.

ABSTRACT

Assuring the security of services in Computational Clouds (CC) is one of the most critical factors in cloud computing. However, it can complicate an already complex environment due to the complexity of the system architecture, the huge number of services, and the required storage management. In real systems, some security parameters of CC are manually set, which can be very time-consuming and requires security expertise.

This paper proposes an intelligent system to support decisions regarding security and tasks scheduling in cloud services, which aims at automating these processes. This system comprises two different kinds of Artificial Neural Networks (ANN) and an evolutionary algorithm, and has as main goal sorting tasks incoming into CC according to their security demands. Trust levels of virtual machines (VMs) in the environment are automatically set to meet the tasks security demands. Tasks are then scheduled on VMs optimizing the makespan and ensuring that their security requirements are fulfilled.

The paper also describes tests assessing the best configurations for the system components, using randomly generated batches of tasks. Results are presented and discussed. The proposed system may be used by CC service providers and CC consumers using Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) Cloud Computing models.

I. INTRODUCTION

Validation of security demands (SD) of tasks processed in Computational Clouds (CC) is a crucial part of the CC workload management process [5], [9]. Personalization of cloud services enables users to change the features of Virtual Machines (VMs) that provide the computational power for executing pools of tasks. Especially the security Trust Levels (TLs) offered by VMs may be changed and adapted to fit the SD of tasks. Another important aspect is to ensure the proper assignment of tasks to suitable VM offering the proper TL to fulfill the SD required by tasks.

This paper presents an intelligent system for management of tasks submitted into the cloud. Fig. 1 displays a representation of the proposed system, which considers three computational cloud units:

- the edge of the cloud,
- the cloud computing center, and
- the cloud storage center.

Additional units could be considered, however these three provide a sufficient level of detail for testing our system.

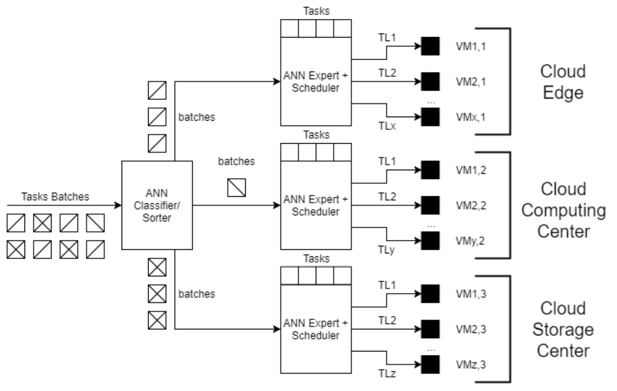

Fig. 1: Model of the proposed system

Tasks are submitted to the cloud in batches, which are classified into security classes according to their tasks SDs. According to this classification, batches are then delivered to be processed at the proper cloud units, thus allowing to determine the workload required in each cloud unit and further to decide about the configuration of the available VMs. These VMs are then customized to serve the TLs that fit the tasks SDs. Finally, tasks in each of the considered cloud units are scheduled on VMs offering proper security levels while minimizing the total processing time.

The presented system is composed of two Artificial Neural Networks (ANNs) and an Evolutionary Scheduler. One of the ANNs is a classifier/sorter ANN, which classifies and sorts batches incoming into the cloud;

Proceedings 32nd European Conference on Modelling and Simulation ©ECMS Lars Nolle, Alexandra Burger, Christoph Tholen, Jens Werner, Jens Wellhausen (Editors)
ISBN: 978-0-9932440-6-3/ ISBN: 978-0-9932440-7-0 (CD)

the other is an Expert ANN that predicts the VMs configurations. The Evolutionary Scheduler optimizes the scheduling of tasks relying on an evolutionary algorithm.

The paper is organized as follows. Section II presents the concepts underlying task processing in CC systems considering security aspects. Section III is devoted to describing the proposed intelligent system and its components. In Section IV, we describe in detail the system testing and discuss its results. The paper ends with Section V, which contains a summary, conclusions based on the experiments results, and ideas for future work.

II. SECURITY OF TASK PROCESSING IN CC SYSTEMS

Processing tasks with security requirements in CC involves many steps [10]. End users start by specifying security demands for their tasks before uploading them into the cloud. Then one has to decide in which component of the cloud infrastructure should tasks be processed. For instance, some tasks may be computed at the cloud edge, while others have to be processed inside the cloud. The collected tasks are then scheduled into available VMs which offer proper security trust levels. Finally, the results of tasks processing are returned to the end users or delivered to another service from CC.

Assuring the security of the users data and cloud infrastructure is a complex process [18], [9]. In the IaaS and PaaS models, often the biggest responsibility for ensuring security in the cloud is on the cloud providers side [3]. Moreover, as there still exist many vulnerabilities in cloud systems, international organization have produced standards providing guidance regarding cloud security [16], [8].

A. *Mapping security requirements into VMs*

Considering a given cloud unit for which there are n tasks to be processed, and that there m available VMs in that CC unit, we define an SD vector describing the security requirements of these tasks [5]:

$$SD = [sd_1, ...sd_n], \tag{1}$$

where sd_j is the security demand value of the jth task to be processed in the cloud unit. Accordingly, we define the TL vector representing the security levels offered by the VMs in that CC unit [5]:

$$TL = [tl_1, ..., tl_m], \tag{2}$$

where tl_i is the trust level offered by the ith VM in the CC unit. A task can only be scheduled on a particular VM that offers a TL equal or higher than the SD of the task. According to the NIST guidelines [2] we propose four levels of security demand/trust level, that is: $sd_j, tl_i \in \{0, 1, 2, 3\}$.

To differentiate security requirements as far as cryptography is concerned, we introduce three classes:
• Class 1, containing only small tasks, processed online using strong but fast enough cryptography algorithms. The RSA asymmetric cipher with 1024 bits key should be used for ciphering and deciphering tasks [15].
• Class 2, requiring operational cryptography [19]. This class of security demands is designed for most tasks, which may have varied workload and are processed mostly off-line, using advanced cryptography protocols without security compromises. If it is possible, these protocols should be designed to be as fast as possible. The RSA asymmetric cipher with 2048 bits key should be used for ciphering and deciphering tasks.
• Class 3, corresponding to data at rest cryptography [4]. This class contains tasks with very heavy workload, mainly ciphering data to be stored, and using the strongest cryptography methods designed for processing data at rest. The RSA asymmetric cipher with 2048 bits keys is used for ciphering and deciphering tasks. Furthermore, tasks are signed with Elliptic Curve Digital Signature Algorithm (ECDSA) based on curve defined over 521 bits field [1].

B. *Security computational overhead*

Each security operation adds a computational overhead that has to be considered during tasks processing and comprises two parts as follows.
1. Part of this overhead influences the task scheduling and includes pre and post-processing security operations, such as verification of the task integrity and ciphering results of task processing. The bias required to deliver tl_i to task j, requiring a security demand of sd_j, is assumed as an estimated time (sec.) denoted by [11]:

$$b_{i,j} = b(sd_j, wl_j, tl_i, cc_i, inputSize_j, outputSize_j) \tag{3}$$

where wl_j is the workload of task j, $inputSize_j$ and $outputSize_j$ are the sizes (in bytes) of the files characterizing the task and storing the corresponding result, respectively, and cc_i is the computational capacity of VM i, $i = 1, 2, .., m, j = 1, 2..., n$. This value can be approximated by the sum of the number of instructions to compute the cryptographic requirements and the workload.
2. The remaining part of the computational overhead is associated with complying the security protocols and performing operations that do not influence the scheduling process, such as, ciphering data stored in the data center (before sending the task back to the end user) and verifying the digital signature of the end user who wants to recover some results from the Cloud Computing system.

C. *Scheduling tasks into VMs by matching SD and TL*

Tasks scheduling is the process of assigning tasks to the available VMs assuring that each task is processed by a VM offering a TL value of at least the required SD and optimising the utilisation of VMs by minimizing the makespan. Our system uses the scheduler previously developed in [10], [11], which has as objective the minimization of the makespan - the time of conclusion of the last task. As an additional criterion, here we introduce the condition of compliance with the required SD.

III. THE INTELLIGENT SYSTEM TO SUPPORT SECURITY DECISIONS

This section describes the proposed system and its components: the sorter ANN, the expert ANN, and the scheduler. The input of the system is a stream of batches of tasks. Each batch is considered separately. The system works according to the schema illustrated in Fig. 1.:

- The first stage consists of classifying batches and sorting them to the appropriate destination in the cloud system considering their security requirements.
- The second stage concerns setting the VMs parameters according to the tasks workloads and security requirements so that all tasks can be computed with adequate security levels and without spare computing time or energy losses.
- The last stage is assigning the tasks from each batch into the created VMs ensuring the proper security level.

The output from the system is a vector describing the TL of all VMs and the complete schedule.

A. The sorter ANN

The system starts by automatically classifying the incoming traffic into security classes, which define the cryptography level required by tasks in a particular batch, without any additional knowledge about them.

The classified batches are then sorted into pools corresponding to their security classes and defining where the tasks should be processed in the CC infrastructure:

- Class 1: tasks that may be processed in the cloud edge,
- Class 2: tasks to be executed by fast VMs inside the cloud, and
- Class 3: tasks that have to be stored in the cloud longer and then processed inside cloud storage centers (when enough computational power is available).

To define the training set for the pattern recognition problem, a set of batches was previously classified. The numbers of tasks requiring each SD value l were counted separately in each batch:

$$N(t)_l = card\{j : sd_j = l\}, l = 0, 1, 2, 3 \tag{4}$$

where t indexes batches entering the CC system, $t = 1, 2, ..., T$ and $card$ represents the set cardinality.

The SD statistics for the t-th batch are stored in vector $input^{recog}(t)$, which is considered as the input for the ANN classifier/sorter.

$$input^{recog}(t) = [N(t)_l]. \tag{5}$$

The security class of each batch t is declared by specifying its target value

$$target^{recog}(t) = k. \tag{6}$$

The training and testing sets were created from data obtained from different batches coming during system operations:

$$\begin{aligned} &TSET^{recog} = \\ &\{(input^{recog}(t), target^{recog}(t))^{t=1,2...,T}\} \end{aligned} \tag{7}$$

The original set was split into three parts: the training $TSET^{recog}_{train}$, containing information from 70% of all batches; the validation set $TSET^{recog}_{valid}$, and testing set $TSET^{recog}_{test}$, each containing 15% of randomly chosen batches (not used for training).

A shallow feed-forward Neural Network was then trained to classify inputs according to the target classes defined before. We used a two-layer feed-forward network with sigmoid hidden and softmax output neurons [12], see Fig. 2.

Additionally, the sorter may help to detect anomalies in batches, for instance, by allowing the identification of differences in security demands from past patterns, therefore supporting detection of hostile tasks or users abnormal behavior.

B. Expert system for setting Trust Levels

The aim of the expert system is to decide the proper TL values for the VMs based on the SD levels required by the tasks. For each cloud unit, an expert ANN was designed to get knowledge about the computational capacities cc of all VMs, as represented in Fig.1.

We examined several strategies of assigning TL values to VMs, for example, by using arbitrary human decisions or Stackelberg Game solutions [12]. Then we trained the ANN to mimic these decisions. To formulate the input for this expert system, we analyzed the individual tasks workload; wl_j denotes the workload of task j. The total workload of tasks requiring the SD level l in batch t is represented by $W(t)_l$

$$W(t)_l = \sum_j \{wl_j : sd_j = l\}, l = 0, ..., 3, t = 1, 2, ..., T \tag{8}$$

.

Vector

$$input^{expert}(t) = [W(t)_l] \tag{9}$$

represents the workload for each SD value l in batch t, and is given as the input to the ANN expert system.

Target values were defined to indicate the trust levels tl for all the virtual machines in the system, in ascending computer capacity order.

Vector

$$target^{expert}(t) = [tl_1, tl_2, ..., tl_m] \tag{10}$$

represents the decisions of an expert for a particular batch t. This vector assigns the trust level tl_i to the i-th VM according to it's computing capacity cc_i and the amount of work that has to be done using trust level tl_i, $i = 1, ..., m$.

The training, validation and testing sets were also formulated considering data obtained from different batches entering the system, through the definition of the set

$$TSET^{expert} = \{(input^{expert}(t), target^{expert}(t))^{t=1,2...,T}\} \tag{11}$$

which was split into training set $TSET^{expert}_{train}$, validation set $TSET^{expert}_{valid}$ and $TSET^{expert}_{test}$, analogously to the process described in Section III-A for the sorter ANN.

A backpropagation feed-forward NN was then trained with the defined inputs and used for targets prediction. We used a two-layer ANN, with sigmoid hidden and linear output neurons [7], as represented in Fig.4.

The quality of prediction was assessed through the coefficient of determination R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}, \tag{12}$$

. where:

- y is the given set of data,
- $\hat{y}$ represents the calculated values of y, and
- $\bar{y}$ represents mean value of y.

A properly trained expert system may be an automatic alternative for human security decisions made by CC administrators. Once the ANN is trained, it can also deal with unknown situations, without requiring additional customization.

C. *Evolutionary scheduler*

Our system uses the Independent Batch Scheduler [13] as the main method of mapping tasks into VMs. The Security Biased Expected Time to Compute (SBETC) matrix with security biases is computed using Eq. (13):

$$SBETC = [ETC[j][i] + SB(SD, TL)(i, j)]^{i=1,...,m}_{j=1,...,n} \tag{13}$$

where

$$ETC[j][i] = wl_j / cc_i \tag{14}$$

in which cc_i is the computational capacity of the i-th VM in Giga Flops per Second (GFLOPS) and wl_j is the workload of j-th task in Flops (FLO); n and m are the number of tasks and the number of VMs, respectively.

The Security Bias Matrix (SB) is obtained by aggregation of security biases in a matrix form:

$$SB(SD, TL)(i, j) = [b(sd_j, wl_j, tl_i, cc_i, inputSize_j, outputSize_j)]^{j=1,2...,n}_{i=1,2...,m}. \tag{15}$$

The full description of this model can be found in [11]. The main scheduling objective is the minimization of the makespan, which can be defined as follows:

$$C_{\max} = \min_{S \in Schedules} \left\{ \max_{j \in Tasks} C_j \right\}, \tag{16}$$

where C_j is the conclusion time of the j-th task, $Tasks$ is the set of tasks in the batch, and $Schedules$ is the set of all possible schedules which can be generated for the tasks from that batch. The scheduler is based on the evolutionary algorithm solution proposed in [10] and [11], using only a particular subset of its features.

IV. NUMERICAL EVALUATION OF THE SYSTEM PERFORMANCE

Cloud Sim test bed (www.cloudbus.org) was used as a testing tool. All security algorithms were implemented in Java. Processing of different dimensions pictures was considered as a tasks set. The examined workload was based on proposed day and night pattern (to represent the load of a CC system). All designed ANN were implemented in MATLAB 2017 (www.mathworks.com). We examined different ANN sizes together with different learning algorithms, to assess the quality of the solutions.

A. *Trust Levels*

The following trust levels of VMs were considered:

- Trust Level 0 (tl=0) - bare tasks are processed without any cryptographic computational overhead (no security required).
- Trust Level 1 (tl=1) - corresponds to the TL required by tasks of Class 1 as defined in Section III.
- Trust Level 2 (tl=2) - corresponds to the TL required by tasks of Class 2 as defined in Section III.
- Trust Level 3 (tl=3) - corresponds to the TL required by tasks of Class 3 as defined in Section III.

B. *Tasks and their security demands*

The security demand of each task was defined according to the trust level values defined in Section III. Tab. 1 presents the sizes of ten pictures that were used for the tests. For each picture, the size is presented in pixels and with a qualitative classification (Small, Medium and Big).

TABLE 1: Pixel size of blurred JPG pictures

Picture Number [PN]	Picture size [pixels]	Size
PN1	200x200	Small
PN2	400x400	Small
PN3	600x600	Medium
PN4	800x800	Medium
PN5	1000x1000	Medium
PN6	1200x1200	Medium
PN7	1400x1400	Medium
PN8	1600x1600	Medium
PN9	1800x1800	Medium
PN10	2000x2000	Big

As bare task we considered a Gaussian Blur operation with 5x5 mask made on each of these pictures. A task resulted from combining a picture size [PN1-PN10] with an SD value. Tab. 2 presents the characteristics of these tasks. For each task, column (1) is the task ID, columns (2) and (4) present the pair (PN, sd), column (3) is the workload in terms of the number of instructions (without security), column (5) is the number of instructions required to process the SD (bias), and column (6) is the total number of instructions to process the task (size).

TABLE 2: All possible tasks and their characteristics in terms of workload without bias, SD value, bias workload and workload with bias

task ID (1)	PN (2)	Instr. bare task wl (3)	sd (4)	Instr. bias b (5)	Instr. TOTAL wl $wl + b$ (6)
1			0	0	180281484
2	PN1	180281484	1	58248804234	58429085718
3			2	189148318241	189328599725
4			3	193351681261	193531962745
5			0	0	366694660
6	PN2	366694660	1	233237558475	233604253135
7			2	762857236471	763223931131
8			3	767753680777	768120375437
9			0	0	582192590
10	PN3	582192590	1	549063149835	549645342425
11			2	1723705647330	1724287839920
12			3	1729619478414	1730201671004
13			0	0	935976091
14	PN4	935976091	1	984022696944	984958673035
15			2	3071839716809	3072775692900
16			3	3079432624626	3080368600717
17			0	0	1381754383
18	PN5	1381754383	1	1467324300905	1468706055288
19			2	4807412731944	4808794486327
20			3	4817016916348	4818398670731
21			0	0	1907429016
22	PN6	1907429016	1	2218939482253	2220846911269
23			2	6930461839807	6932369268823
24			3	6942582060238	6944489489254
25			0	0	2541112486
26	PN7	2541112486	1	2877893318993	2880434431479
27			2	9441396183684	9443937296170
28			3	9456608456473	9459149568959
29			0	0	2943581287
30	PN8	2943581287	1	3761249543741	3764193125028
31			2	12339533813479	12342477394766
32			3	12357819370145	12360762951432
33			0	0	3672489994
34	PN9	3672489994	1	4821170159062	4824842649056
35			2	15973646688412	15977319178406
36			3	15995539895451	15999212385445
37			0	0	5027151620
38	PN10	5027151620	1	5668347564768	5673374716388
39			2	19726744097608	19731771249228
40			3	19752079020993	19757106172613

C. ***Classifier/sorter ANN tests***

The batches were classified as follows:

• Batch Type from Class 1 [BT1] - containing only small tasks which demand on-line and fast cryptography, which can be calculated e.g. on the edge of the cloud.

• Batch Type from Class 2 [BT2] - containing mixed big, medium and small tasks. Each task has to be considered separately.

• Batch Type from Class 3 [BT3] - comprising only big tasks that have to be sent to the cloud and stored until there is enough computational capacity available for the cryptographic bias.

The classifier/sorter ANN was designed to classify each batch into the proper type (BT1, BT2 or BT3). The batches workload in all three classes was generated according to following the day-night pattern function:

$$f(x) = \begin{cases} 25\sin(\frac{x\pi}{12} + 75) & \text{when } \sin(\frac{x\pi}{12}) > 0 \\ 75\sin(\frac{x\pi}{12} + 75) & \text{when } \sin(\frac{x\pi}{12}) \leq 0 \end{cases} \quad (17)$$

The classifier/sorter ANN uses the Scaled Conjugate Gradient (SCG) backpropagation learning method, which is appropriate for classification [6]. Neural Networks were tested with different numbers of neurons: 5, 10, 15, 20, 25, 30, 50, 100. Our focus is on assessing the True Positive Rate (TPR) for BT3 because this is the most critical classification. In our system, the worst situation occurs when a BT3 batch is classified as BT2 or BT1, meaning that a batch requiring high computational power could be delivered to a VM not offering enough computational power.

For each NN configuration, we made 10 measurements and computed the TPR of BT3 mean and standard deviation values. The corresponding results are shown in Tab. 3.

TABLE 3: Classifier/sorter ANN: TPR of BT3 average, standard deviation for each tested number of neurons in the hidden layer

Avr % ± St. dev.%	No. neurons
86,8 % ± 4,7%	5
85,8% ± 9,0%	10
86,9% ± 6,5%	15
90,9% ± 4,2%	20
86,5% ± 8,9%	25
88,7 % ± 6,1%	30
86,9 % ± 9,0%	50
86,4 % ± 8,9%	100

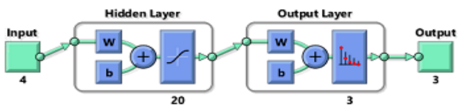

Fig. 2: Sorter ANN, a two-layer network with 20 sigmoid hidden neurons and 3 softmax output neurons.

The highest TPR for BT3 appeared when the SCG algorithm was applied for the ANN with 20 neurons (90,9% ± 4,2%). So this ANN was chosen as the most accurate one (Fig. 2).

The NNs results were also validated using confusion matrices for the training and testing process. Fig. 3 displays an example of the confusion matrix for the sorter ANN with 20 neurons. Green cells show the numbers of correct classifications (for each class separately) and the corresponding percentage of all data. Red cells represent these figures for incorrect predictions (for each class separately). Gray cells in the last column indicate the percentage of correct predictions for each class, while gray cells in the last row present the percentage of correctly classified cases. The blue cell shows the percentage of overall correct and incorrect predictions. For validation, the gray cell in the red frame was used, as it indicates the percentage of properly classified class 3 tasks. In this example the TPR for BT3 is 93.8%.

D. ***Expert ANN tests***

The expert ANN has to allocate proper security levels to virtual machines (a machine with a given security level can compute only tasks with the same or lower security level). This network was created and tested with three learning methods appropriate for prediction [6]:

• SCG backpropagation algorithm,
• Bayesian Regularization (BR), and
• Levenberg-Marquardt (LM).

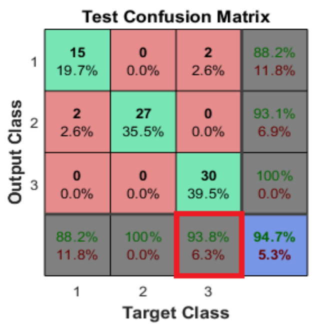

Fig. 3: Confusion Matrix for the Sorter ANN with 20 Neurons

For each of these methods, an ANN with different hidden layer size (5, 10, 15, 20, 25, 30, 50, 100) was ran ten times. To evaluate the Expert ANN performance we used the coefficient of determination (Eq. (12)). We computed it on the testing set (15% of all data) to determine which configuration of the method and the number of neurons present the highest R^2 value. Tab. 4 displays the average and standard deviation R^2 values in the ten runs of each configuration.

TABLE 4: Expert ANN: Average and standard deviation values of R^2 for learning methods LM, BR and SCG with different number of neurons

LM Avr % ± St. dev.	BR Avr % ± St. dev.	SCG Avr % ± St. dev.	Neurons No. of neurons
95% ± 2%	96,3% ± 2%	91,8% ± 2%	5
92% ± 3%	87,5% ± 10%	90% ± 4%	10
92% ± 3%	87,6% ± 14%	87,9% ± 4%	15
91% ± 3%	91,7% ± 3%	87,5% ± 5%	20
90% ± 2%	88,1% ± 8%	88,1% ± 5%	25
89% ± 4%	87% ± 3%	88,2% ± 3%	30
88% ± 5%	85,2% ± 5%	87,6% ± 4%	50
81% ± 3%	85,1% ± 5%	73,9% ± 7%	100

The best value of the coefficient of determination was obtained for the BR algorithm and an the ANN with 5 neurons (96,3% ± 2%). This ANN was chosen as the final solution, as illustrated by Fig. 4.

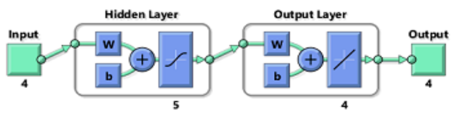

Fig. 4: Expert ANN, a two-layer network with 5 sigmoid hidden neurons and 4 linear output neurons.

*E. **Evolutionary scheduler tests***

The aim of these tests was to examine how much time we may gain by scheduling tasks for generated VMs instead of submit them randomly.
For this purpose we have simulated a multi-cloud environment consisting of three types of cloud architectures, namely: cloud edge, cloud computing center and cloud storage center (see: Fig. 1). This environment is based on a real cloud characteristics from public (cloud storage center) and private, academic (cloud edge and cloud computing center) infrastructure. Tab. 5) presents the characteristics of the simulated environment: the VMs simulated in each CC unit, their types, their computational capacity (in GFLOPS) and their range of TLs.
Batches of tasks were generated and classified by the Classifier/sorter ANN and allocated to one of the simulated environment parts. For each of these components, the Expert ANN was determining the proper TLs for the corresponding VMs. Finally, the evolutionary scheduler was assigning tasks into particular VMs of each of the three parts of the tested system.

TABLE 5: Characteristics of the simulated Cloud, see Fig.1, and of the VMs used for simulation

Cloud Edge	cc	tl$\in 0, 1, 2, 3$
VM number, type	[GFLOPS]	min:max
1, m1.tiny	0.293799	0
2, m1.tiny	0.293799	0
3, m1.tiny	0.293799	0
4, m1.tiny	0.293799	0
Cloud Computing Center	cc	tl$\in 0, 1, 2, 3$
VM number, type	[GFLOPS]	min:max
5, m1.tiny	0.293799	0:1
6, m1.small	1.227791	0:2
7, m1.medium	1.856781	0:2
8, m1.large	3.444585	0:3
Cloud Storage Center	cc	tl$\in 0, 1, 2, 3$
VM number, type	[GFLOPS]	min:max
9, m3.medium	97,6	0:1
10, m3.large	396,8	0:2
11, m3.xlarge	787,2	0:2
12, m3.2xlarge	1523,2	0:3

Tab. 6 presents the value of gain as far as makespan Eq. (16) is concerned. The presented results were obtained for 20 runs of the implemented evolutionary algorithm after 300 epochs.The results reveal that the maximum gain achieved by the application of the evolutionary scheduler was reached for the cloud storage center, being greater than twice the gain for the cloud edge.

TABLE 6: Results of applying the evolutionary scheduler: mean values of makespan gain in percentage for the Random task sender (Rand) and the Evolutionary scheduler (Evol)

CC part	Gain Rand	Gain Evol
Edge	0%	15%
Computing Center	0%	35%
Storage Center	0%	42%

V. SUMMARY

In this paper we presented an intelligent system for supporting security services in Computational Clouds (CC), which can improve the quality of security cloud services. The system is composed of two different kinds of Artificial Neural Networks (ANN) and an evolutionary algorithm. The first stage of system operation concerns sorting incoming batches of tasks according to their security demands. This allows one to divide the traffic into streams that can be processed by different parts of the CC environment. Our system uses the classifier/sorter ANN to perform this stage. The second stage, which is performed by the expert ANN, consists of fitting the security services offered by the VMs into the security demands of tasks in each of the CC components. Finally, the scheduler, based on the evolutionary algorithm, maps tasks into VMs, minimizing the makespan of tasks processed in the corresponding CC component. This scheduler considers the tasks characteristics in terms of size and security requirements, as well as the particular VM security services. Additional possible feature of our system is the possibility of detection of deviations in traffic incoming into CC. Therefore, in the future, it can be used to support the detection of security threats, like tasks injection or malicious workload.

The experimental results presented in this paper demonstrate and confirm the effectiveness of the system. The system is designed for CC service providers and CC consumers using Infrastructure as a Service or Platform as a Service Cloud Computing models.

In the future, we would like to introduce genetic algorithms for supporting the automatic detection of security threats.

ACKNOWLEDGEMENT

This article is based upon work from COST Action IC1406 "High-Performance Modelling and Simulation for Big Data Applications" (cHiPSet), supported by COST (European Cooperation in Science and Technology).

References

[1] W. A. Al-Hamdani. Elliptic curve for data protection. In *Proceedings of the 2011 Information Security Curriculum Development Conference*, InfoSecCD '11, pages 1–14, New York, NY, USA, 2011. ACM.

[2] P. Bowen, J. Hash, and M. Wilson. NIST Special Publication 800-100, Information Security Handbook: A Guide for Managers, 2006.

[3] CSA. *Security Guidance for Critical Areas of Focus in Cloud Computing V3.0.* Cloud Security Alliance, 2011.

[4] Google. Encryption at rest in google cloud platform, 2016.

[5] D. Grzonka, A. Jakóbik, J. Kołodziej, and S. Pllana. Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security. *Future Generation Computer Systems*, 2017.

[6] M. T. Hagan, H. B. Demuth, and M. Beale. *Neural Network Design.* PWS Publishing Co., Boston, MA, USA, 1996.

[7] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.

[8] ISO/IEC. ISO/IEC 27002:2005 – Information Technology - Security Techniques - code of practice for information security management, 2005.

[9] A. Jakóbik. Big data security. In F. Pop, J. Kołodziej, and B. Di Martino, editors, *Resource Management for Big Data Platforms: Algorithms, Modelling, and High-Performance Computing Techniques*, pages 241–261, Cham, 2016. Springer International Publishing.

[10] A. Jakóbik, D. Grzonka, J. Kołodziej, and H. González-Vélez. Towards secure non-deterministic meta-scheduling for clouds. In *30th European Conference on Modelling and Simulation, ECMS 2016, Regensburg, Germany, May 31 - June 3, 2016, Proceedings.*, pages 596–602, 2016.

[11] A. Jakóbik, D. Grzonka, and F. Palmieri. Non-deterministic security driven meta scheduler for distributed cloud organizations. *Simulation Modelling Practice and Theory*, 76:67 – 81, 2017.

[12] A. Jakóbik and A. Wilczynski. Using polymatrix extensive stackelberg games in security-aware resource allocation and task scheduling in computational clouds. *Journal of Telecommunications and Information Technology (JTIT)*, pages 1–71, 2017.

[13] J. Kołodziej. *Evolutionary Hierarchical Multi-Criteria Metaheuristics for Scheduling in Large-Scale Grid Systems.* Springer Publishing Company, Incorporated, 2012.

[14] S. Matsuda and S. Moriai. Lightweight cryptography for the cloud: Exploit the power of bitslice implementation. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES 2012*, pages 408–425, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[15] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[16] T. E. Network and I. S. Agency. *Cloud Computing Benefits, risks and recommendations for information security.* ENISA, 2009.

[17] NIST. Publication 500-291, Version 2, NIST Cloud Computing Standards Roadmap. Technical report, 2013.

[18] P. K. S. Gupta. *Taxonomy of cloud security*, volume 3. 2013.

[19] B. Schneier. *Applied Cryptography (2Nd Ed.): Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, Inc., New York, NY, USA, 1995.

AUTHOR BIOGRAPHIES

JACEK TCHÓRZEWSKI received his B.Sc. and M.Sc. degrees with distinctions in Computer Science at Cracow University of Technology, Poland, in 2016 and 2017, respectively. Currently he is Research and Teaching Assistant at Cracow University of Technology and Ph.D. student at AGH Cracow Univeristy of Science and Technology. His e-mail address is: jacek.tchorzewski@onet.pl

ANA RESPÍCIO is Assistant Professor at the Informatics Department, Faculty of Science, University of Lisboa, and Senior Researcher of CMAFCIO. She is vice-chair of the IFIP WG 8.3: Decision Support. Her e-mail address is: alrespicio@fc.ul.pt

JOANNA KOŁODZIEJ is an associate professor in Research and Academic Computer Network (NASK) Institute and Department of Computer Science of Cracow University of Technology. She serves also as the President of the Polish Chapter of IEEE Computational Intelligence Society. Her e-mail address is: joanna.kolodziej68@gmail.com