# Evolutionary Reinforcement Learning for Adaptively Detecting Database Intrusions

SEUL-GI CHOI*, *Department of Computer Science, Yonsei University, Seoul 03722, South Korea.*

SUNG-BAE CHO**, *Department of Computer Science, Yonsei University, Seoul 03722, South Korea.*

## Abstract

Relational database management system (RDBMS) is the most popular database system. It is important to maintain data security from information leakage and data corruption. RDBMS can be attacked by an outsider or an insider. It is difficult to detect an insider attack because its patterns are constantly changing and evolving. In this paper, we propose an adaptive database intrusion detection system that can be resistant to potential insider misuse using evolutionary reinforcement learning, which combines reinforcement learning and evolutionary learning. The model consists of two neural networks, an evaluation network and an action network. The action network detects the intrusion, and the evaluation network provides feedback to the detection of the action network. Evolutionary learning is effective for dynamic patterns and atypical patterns, and reinforcement learning enables online learning. Experimental results show that the performance for detecting abnormal queries improves as the proposed model learns the intrusion adaptively using Transaction Processing performance Council-E scenario-based virtual query data. The proposed method achieves the highest performance at 94.86%, and we demonstrate the usefulness of the proposed method by performing 5-fold cross-validation.

*Keywords*: Database intrusion detection, evolutionary learning, reinforcement learning, multi-layer perceptron, online learning, insider intrusion

## 1. Introduction

Most of a company's sensitive data are managed by a relational database management system (RDBMS), the most popular database system [24]. It is important to maintain security from data damage and information leakage [29]. When a database is hacked, the financial damage to the organization and business is estimated to be on average millions of dollars [13].

RDBMS can be attacked by an outsider or insider. It is important to detect an insider intrusion because if it remains undetected for a long time, it can cause serious damage to a database system [23]. Outsider intrusions can be defended by programming techniques, but insider intrusions are not easy to defend because they have access authority. In particular, insider intrusions are evolving constantly [21] and it is difficult to detect intrusions because they can deform patterns [17].

Most intrusion detection studies for database detect anomaly transactions that access data [11, 12, 26]. Role-based access control (RBAC) is an intrusion prevention method by allowing a user to access the system with limited roles [10]. In the RBAC system, permissions are associated with roles rather than individual users, and a profile of normal behaviour is built in each role to detect anomalous behaviour that deviates from normal behaviour (queries belonging to each role class

*E-mail: abooun@yonsei.ac.kr
**E-mail: sbcho@yonsei.ac.kr

doi:10.1093/jigpal/mjz053

are normal queries, and queries that are out of roles are abnormal queries). Providing an intrusion detection system (IDS) to the RBAC databases can protect against insider intrusion that is difficult to defend because they are provided by legitimate users of systems with access to data and resources [27].

In order to model abnormal behaviour as a method of internal intrusion detection, the information of the behaviours should be collected [3]. In a real environment, however, the amount of normal data is much larger than that of abnormal data, so it is not easy to discriminate the abnormal data [9]. Insider intrusions are also more difficult to detect than outsider intrusions due to pattern variations [17]. Hence, it is needed to detect an evolving intrusion pattern regardless of the amount of data.

In this paper, we propose an adaptive IDS based on evolutionary reinforcement learning that consists of two neural networks [1]. The action network detects the intrusion and the evaluation network provides feedback to the detection of the action network. We find the optimal model by encoding the weights of the network as the individual and producing populations of better individuals. The proposed model can adaptively deal with intrusion patterns regardless of the amount of data through continuous online learning.

We use the virtual database query data generated based on the Transaction Processing performance Council-E (TPC-E) scenario in the previous study [27] to verify the proposed model. The total number of data is 11,000, among which 1,000 data are generated for each role from 11 roles. Experimental results show that the performance for detecting intrusion is improved as the proposed model learns the intrusion patterns.

This paper is organized as follows. In Section 2, we review the studies of internal intrusion detection in the database. Section 3 describes the proposed model. Section 3.1 presents the experimental results of the data set. Finally, Section 3.2 concludes this paper.

## 2. Related work

There are two approaches to detect database intrusions. Misuse detection is judged by matching the current task with the attack pattern based on knowledge of known attacks [15]. On the other hand, anomaly detection is based on a knowledge of normal behaviour. After creating a profile for the normal behaviours, an abnormal behaviour that is not match the profile is regarded as an intrusion [2]. This approach is effective for detecting the internal database intrusion because it can respond to an unknown attack.

Table 1 shows the recent related papers. Over the two decades, the database IDS has consistently been proposed using machine learning.

In the 2000s, Valeur *et al.* [34] proposed an anomaly detection system to learn the profile of normal access to the database by the web application. Mathew *et al.* [19] modelled user access patterns by profiling data elements accessed by users. They proposed a feature extraction method that summarizes and clusters the query's result tuples and proves its performance using a machine learning method. Chagarmudi *et al.* [7] presented a model to prevent malicious insider activities of an authenticated user in the database application level. User's tasks were mapped to several transactions and analysed through simulation. In the previous study, Ronao and Cho [27] proposed an anomaly detection system that uses principal component analysis (PCA) and a weighted random forest using TPC-E virtual query data. They use PCA to extract meaningful features from the query, and reduce the dimensionality. Random forest with weighted voting minimizes false-positive and false-negative errors.

The above studies were learned with restricted data. Hence, they cannot adaptively cope with intrusions with various patterns. We propose a detection model for online learning.

TABLE 1. Related work in database intrusion detection for insider intrusion.

| **Year** | **Authors** | **Model** | **Description** |
|---|---|---|---|
| 2017 | Mazzawi *et al.* [20] | *k*-means clustring | An algorithm for detecting malicious user activity in database is proposed. The algorithm relies on the consistency of the individual in past and a similar user group. |
| 2016 | Ronao & Cho [27] | Random forest | The extracted features from SQL queries are highly sparse, and PCA is used for dimension reduction. Random forest classified either normal or abnormal query. |
| 2014 | Sheykhkanloo [30] | Neural network | The problem is set to preventing SQL injection. A neural network model detects a given SQL query is a malicious or not. |
| 2014 | Makiou [18] | Naïve Bayes | A hybrid approach for web-based SQL injection is proposed. Extract features from SQL queries and classify using a naïve Bayesian model. |
| 2012 | Salama *et al.* [28] | Rule-based algorithm | A framework for misuse and anomaly detection techniques is proposed. A behaviour profile and rule are created and saved on XML file. A rule-based detection algorithm is used for detecting anomalies. |
| 2010 | Kundu *et al.* [16] | Sequence alignment | An IDS using inter-transactional and intra-transactional features is proposed. Use the sequence alignment for comparing database access patterns of genuine users and intruders. |
| 2010 | Skaruz *et al.* [31] | Recurrent neural network | The time-series characteristic of SQL attacks is considered. Recurrent neural network and gene expression programming are used for detecting intruders. |
| 2010 | Pinzon *et al.* [25] | Ensemble of neural networks and support vector machine | A combination technique of neural network and SVM is proposed for the detection of SQL injection attacks. Agent-based learning and adapting are enabled. |
| 2009 | Bockermann *et al.* [5] | Self-organizing map (SOM) | In order to detect malicious behaviour in database transactions, SOM and SQL-based feature extraction methods are used. |

## 3. The proposed adaptive database intrusion detection model

Figure 1 shows the structure of the proposed model. The proposed model consists of two-phase: evaluation network and action network. The key idea is that the evaluation network guides the direction in which the action network learns, and the action network continues to evolve based on feedback from the evaluation network to detect abnormal queries. Two networks are evolved and reinforced, making it easier to adaptively detect abnormal queries.

The evaluation network and the action network are multi-layer perceptron (MLP) as follows. Let $f_Q = [f_1, f_2, \cdots, f_{277}]$ be the feature set of the preprocessed query. $f_Q$ is fed into evaluation network

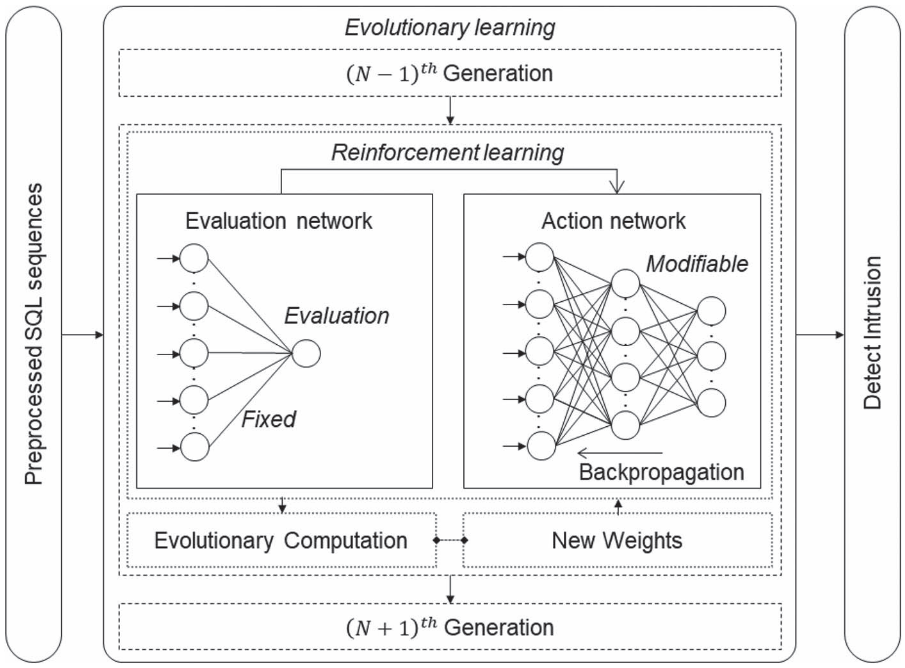

FIGURE 1. The architecture of adaptive database intrusion detection model. It consists of two neural networks: evaluation network and action network.

and action network. The evaluation network outputs the learning rate of the query $\eta$, and action network outputs the detection result $D_Q$. The evaluation network guides the learning direction of the action network by giving the learning rate to be applied to the error backpropagation for the detection result. By encoding the weights of the networks into individuals, we produce better populations over generations.

## 3.1. *Evolutionary learning*

The action network and evaluation network are encoded into one individual. For a neural network as phenotype, the structure of nodes and edges is fixed, and the weights of the network are encoded as genotype. One individual represents the initial weights of the action network and the evaluation network. The initial weights of the action network perform reinforcement learning by applying a learning rate given as the feedback of the evaluation network after the judgment of the action network. The population, which is a set of individuals, evolves over generations to find the optimal set of weights for detecting abnormal queries. We set a population as 50 individuals. Figure 2 shows an example of encoding and evolution operators.

***Encoding.*** If the individual is too lengthy, convergence to the optimal solution may slow down [14]. When the weights are encoded as real values, we need to be considerate on the number of the weights [22]. An individual has the representation with the maximum number of neurons. Appropriate number of neurons is decided in the process of evolution, and the weights are determined accordingly.

$$Individual_i = < w_1, w_2, \ldots, w_x > \text{ where } w_k \in \Re \tag{1}$$

TABLE 2. Reinforcement learning algorithm.

```
#Evaluation Network
Input: feature vector for a query to the evaluation network
    Execute the activation function
Output: Learning rate $\eta$ of the query as the impact of features

#Action Network
    1. Input the feature vector of a query to the action network
    2. Execute the activation function
    3. Output the highest probability of nodes in output layer
    4. If classification is correct
           $\eta^* = \begin{cases} 1-\eta & if\ \eta \geq 0.5 \\ \eta & otherwise \end{cases}$
       else
           $\eta^* = \begin{cases} \eta & if\ \eta \geq 0.5 \\ 1-\eta & otherwise \end{cases}$
    5. Reflect new learning rate $\eta^*$ on loss function
    6. Backpropagate the error
```

***Fitness.*** The fitness of the individual is evaluated by the detection accuracy of the model for one epoch.

$$fitness = \frac{no.\ of\ correct\ classifications}{no.\ of\ all\ validation\ data}$$

***Selection.*** Depending on how the chromosome is selected, it is sometimes slower to approach the optimal solution, or it converges towards local optima. We employ roulette-wheel selection which is the most typical selection method. This is to select a good parent individual by giving the probability of selection as proportionate to the fitness of the individual [4]. We raise the average fit as much as possible to keep the diversity of the genes and prevent it from falling into the local optima.

$$Prob_i = \frac{fitness_i}{\sum_j finess_j}\ where\ j = 1 \ldots pop.size \tag{2}$$

***Crossover.*** Uniform crossover is employed to inherit the gene of the selected parents with threshold probability $P_C$. When two parent individuals are $S_1$ and $S_2$, a random number is generated at each position of the individual. If the number exceeds the threshold probability, the weight of $S_1$ is selected. Otherwise, the weight of $S_2$ is selected [32].

***Mutation.*** We allow mutation to avoid local optima. If the random number generated at each position of the child individual is smaller than the threshold probability $P_M$, the corresponding weight is randomly mutated. Otherwise, the weight is maintained.

$$< w_1, w_2, \ldots, w_k > \rightarrow < w'_1, w'_2, \ldots, w'_n >\ where\ w_i, w'_i \in [0,1] \tag{3}$$

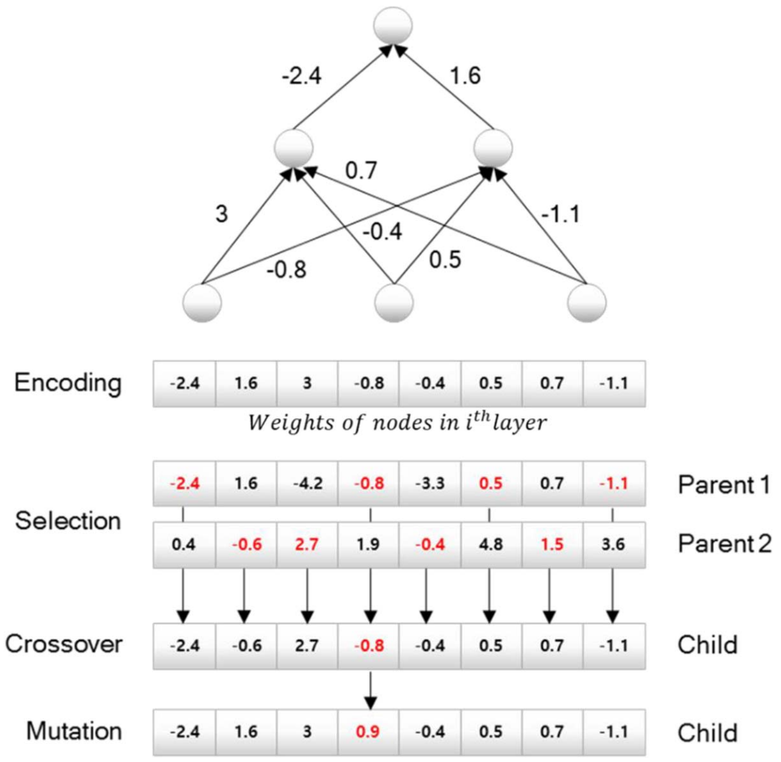

FIGURE 2. Example of encoding and evolution operators.

### 3.2. *Reinforcement learning*

Reinforcement learning is employed to the weights of action networks for online learning. It occurs whenever an individual judges a query within a generation. The evaluation network assumes that the weights assigned by evolution are ideal, and guides the direction of the action network to learn [9].

The initial weights of the evaluation networks are fixed within a generation through evolution. When the action network performs backpropagation, the learning rate $\eta$ is provided by the evaluation network. If a query is classified as abnormal in the evaluation network, the decreased learning rate is provided in backpropagation of action network; otherwise, the increased learning rate is provided. The reinforcement learning process of a query is as shown in Table 2.

## 4. Experiments

### 4.1. *Experimental setup*

***Dataset***. In this paper, we use the virtual query data based on TPC-E database scenarios collected in the previous study, because the query log of the actual database has restrictions on access regarding security. TPC-E scenarios are on-line transaction processing workload of a brokerage company provided by the TPC [33].

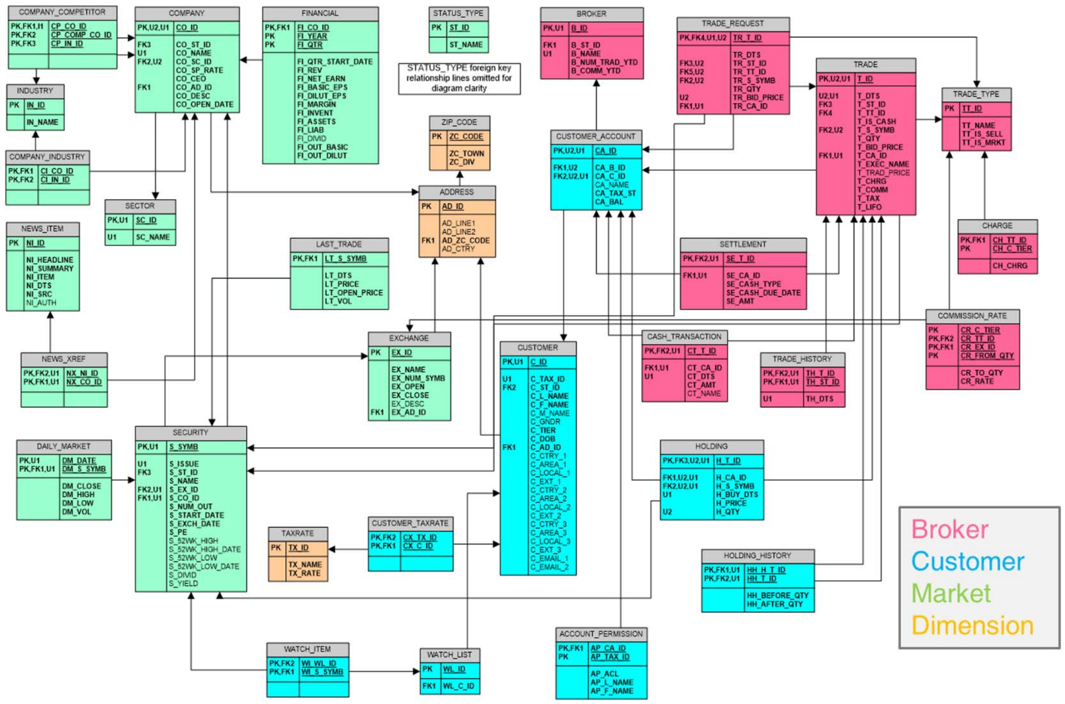
Broker
Customer
Market
Dimension
STATUS_TYPE foreign key relationship lines omitted for diagram clarity

FIGURE 3. TPC-E schema.

TABLE 3. Eleven transactions for TPC-E scenarios.

| No. | Transaction | Query | Authority |
|---|---|---|---|
| 0 | Broker-volume transaction | SELECT | Read-only |
| 1 | Customer-position transaction | SELECT | |
| 2 | Market-watch transaction | SELECT | |
| 3 | Security-detail transaction | SELECT | |
| 4 | Trade-status transaction | SELECT | |
| 5 | Trade-lookup transaction | SELECT | |
| 6 | Trade-order transaction | SELECT/INSERT | Read/Write |
| 7 | Trade-update transaction | SELECT/UPDATE | |
| 8 | Data-maintenance transaction | SELECT/UPDATE | |
| 9 | Market-feed transaction | SELECT/INSERT/UPDATE/DELETE | |
| 10 | Trade-result transaction | SELECT/INSERT/UPDATE/DELETE | |

The TPC-E schema consists of 11 read-only and read/write transactions, with 33 tables and 191 attributes categorized into four categories: customer, broker, market, and dimension. TPC-E schema structures correspond to roles as obtained from the RBAC model. Figure 3 shows the TPC-E schema and Table 3 shows 11 transactions based on the TPC-E scenario.

We refer the transaction database footprint and pseudocode found in [33] for the role implementations. Each role has a set of specific tables T (and its corresponding attributes A) that it is allowed to

TABLE 4. Feature set by field.

| **Field** | **Description** | **Feature** | **#** |
|---|---|---|---|
| SQL-CMD[] | Command features | Query mode, query length | 2 |
| PROJ-REL-DEC[] | Projection relation features | #projected relations, position of the relations | 3 |
| PROJ-ATTR-DEC[] | Projection attribute features | #attributes, #attributes per table, position of the attribtues | 66 |
| SEL-ATTR-DEC[] | Selection attribute features | #attributes, #attributes per table, position of the attribtues | 67 |
| ORDBY-ATTR-DEC[] | ORDER BY clause features | #attributes, #attributes per table, position of the attribtues | 67 |
| GRPBY-ATTR-DEC[] | GROUP BY clause features | #attributes, #attributes per table, position of the attribtues | 67 |
| VALUE-CTR[] | Value counter features | #string values, length of string values, #numeric value, #funtions, #JOIN, #AND/OR | 5 |

*#attributes (in a clause), #attributes per table (in a clause per table), position of the attributes (in a clause).

TABLE 5. Parameters for the evolutionary computation.

| **Parameter** | **Value** |
|---|---|
| The number of generations | 1,000 |
| The number of populations | 50 |
| Crossover rate | 0.95 |
| Mutation rate | 0.001 |
| Selection method | roulette wheel |
| Crossover method | uniform crossover |

access and a set of commands C that it is allowed to execute. 1,000 queries were generated for each role class. Anomalous queries were generated with the role annotation negated; if a certain normal generated query is under role class 1, then it is effectively transformed into an anomalous one if the query's role is changed to any other role besides role 1. Five-fold cross-validation was performed and 30% of the total dataset was transformed into anomalous queries for every run.

The queries are collected using a random query generator with a total of 11,000 queries for each transaction. The queries are parsed into a query clause and extracted into the query attribute represented as a vector from the parsed query. The vector has 277 attributes in total, as shown in Table 4. We have extracted the subsets of attributes that are highly correlated with the class while having low inter-correlation are preferred. We have selected 19 attributes as features.= {SQL-CMD[], PROJ-REL-DEC[], PROJ-ATTR-DEC[], SEL-ATTR-DEC[],, GRPBY-ATTR-DEC[], VALUE-CTR[]}.

***System Setup.*** There are 1,000 query data per a role. The training set and test set were split by 8:2. Table 5 shows the parameters of the evolutionary computation. In the experiment, the action network and evaluation network consist of 3 layers: input layer, hidden layer and output layer.

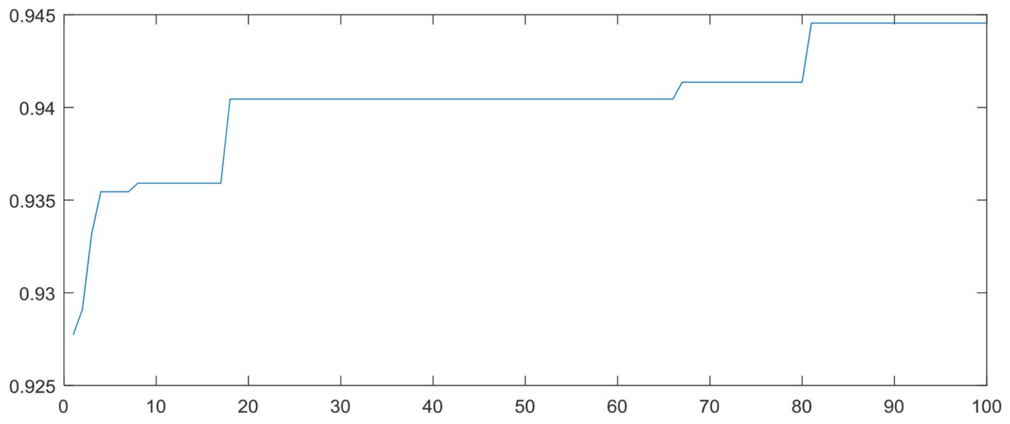

FIGURE 4. Performance per generation.

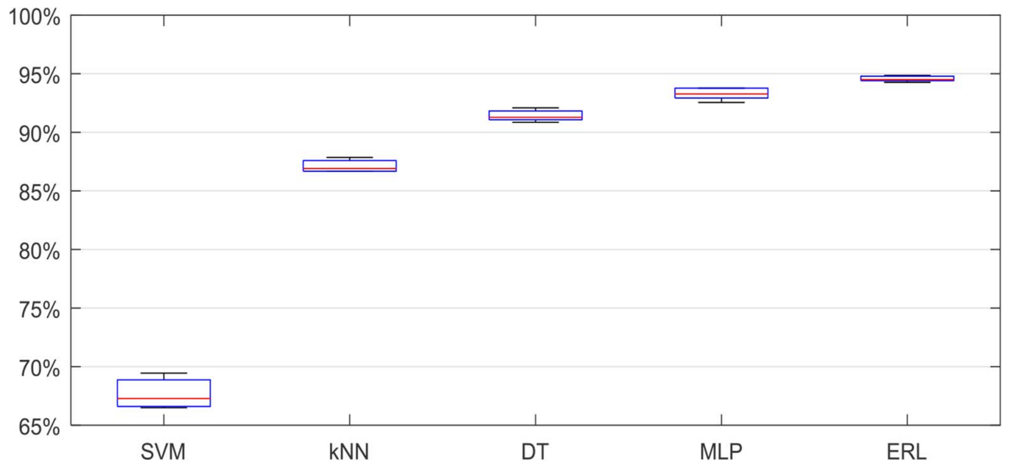

FIGURE 5. Comparison of the performance for 5-fold cross validation.

### 4.2. *Result and analysis*

Experimental results showed that the highest accuracy of the proposed model was 94.86%. As the generation increased, the performance improved steadily. After the 80th generation, the average performance was converged with 94.5% accuracy. Figure 4 shows the performance per generation for the proposed model. We verified that performance was steadily improved as reinforcement learning even after convergence through evolutionary learning.

We performed 5-fold cross-validation with other machine learning techniques to show the usefulness of the proposed method. The proposed model showed the highest performance in comparison with other techniques, followed by MLP, decision tree and *k*-nearest neighbours. Figure 5 shows a boxplot for the accuracy of 5-fold cross-validation.

Table 6 shows the confusion matrix of the best detection model. The prediction of class 0 is perfectly correct, and the performances of roles with read-only access are almost 100% except

TABLE 6. Confusion matrix of the proposed model.

| | | Predicted class | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **C0** | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | **C7** | **C8** | **C9** | **C10** | **Accuracy** |
| True class | **C0** | **208** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100.00% |
| | **C1** | 0 | **195** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 99.49% |
| | **C2** | 0 | 0 | **202** | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 99.02% |
| | **C3** | 0 | 0 | 1 | **221** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 99.10% |
| | **C4** | 0 | 0 | 1 | 0 | **159** | 4 | 0 | 13 | 0 | 1 | 1 | 88.83% |
| | **C5** | 0 | 1 | 0 | 0 | 0 | **200** | 0 | 0 | 0 | 0 | 0 | 99.50% |
| | **C6** | 0 | 0 | 1 | 0 | 0 | 0 | **174** | 0 | 0 | 0 | 34 | 83.25% |
| | **C7** | 0 | 0 | 2 | 0 | 0 | 23 | 0 | **171** | 0 | 5 | 1 | 84.65% |
| | **C8** | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | **188** | 0 | 0 | 98.95% |
| | **C9** | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | **177** | 0 | 98.88% |
| | **C10** | 0 | 0 | 0 | 0 | 0 | 0 | 14 | 0 | 0 | 3 | **192** | 91.87% |
| Total accuracy | | | | | | | | | | | | | **94.86%** |

for role 4. The role 4 is incorrectly classified into role 7 (13 cases) because two roles have the same access for three tables. The role 6 is misclassified into role 10 (34 cases), and these have access to the same ten tables. Also, role 7 (trade-update transaction) and role 5 (trade-lookup transaction) have almost same accessible table except for only one as shown in Figure 3. These kinds of misclassifications can be improved by controlling property access, which did not restrict in the query generation process, not to the similar queries by accessing the same table.

## 5. Conclusions

In this paper, we have proposed an adaptive intrusion detection model that can deal with a potential insider intrusion patterns even with a restricted amount of data. Experiment results have demonstrated that the detection performance of the proposed model is improved as the model learns the intrusion, especially in RBAC database security [6, 8]. The proposed model is very effective for intrusion detection compared to other machine learning, and verified that it can detect dynamically abnormal queries more easily by combining evolutionary learning and reinforcement learning. It allows our proposed model to adaptively detect intrusions. It is necessary to investigate the evolution process of the detection model such as the crossover and mutation operators by analysing the individual with the highest performance by generation, which is the future work to be addressed.

## Acknowledgements

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract (UD190016ED).

## References

[1] D. Ackley and M. Littman. Interactions between learning and evolution. *Artificial Life II*, **10**, 487–509, 1991.

[2] E. Bertino, E. Terzi, A. Kamra and A. Vakali. Intrusion detection in RBAC-administered databases. In *Computer Security Applications Conference*, pp. 173–182, 2005.

[3] E. Bertino, A. Kamra and J. P. Early. Profiling database application to detect SQL injection attacks. *Performance, Computing, and Communications Conference*, IEEE, New Orleans, LA, USA, 449–458, 2007.

[4] T. Blickle and L. Thiele. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation*, **4**, 361–394, 1996.

[5] C. Bockermann, M. Apel and M. Meier. Learning SQL for database intrusion detection using context-sensitive modeling. *International Conference on Detect Intrusions Malware Vulnerability*, Springer, Berlin, Heidelberg, 196–205, 2009.

[6] M. A. de Carvalho and P. Bandiera-Paiva. Health information system role-based access control current security trends and challenges. *Journal of Healthcare Engineering*, **2018**, 8 pages, 2018, Article ID 6510249 10.1155/2018/6510249.

[7] M. Chagarlamudi, B. Panda and Y. Hu. Insider threat in database systems: preventing malicious users' activities in databases. *Information Technology: New Generations*, IEEE, Las Vegas, NV, USA, 1616–1620, 2009.

[8] J. P. Cruz, Y. Kaji and N. Yanai. RBAC-SC: role-based access control using smart contract. *IEEE Access*, **6**, 12240–12251, 2018.

[9] L. Ertoz, E. Eilertson, A. Lazarevic, P. N. Tan, V. Kumar, J. Srivastava and P. Dokas. Minds-Minnesota intrusion detection system. *Next Generation Data Mining*, Chapman and Hall/CRC, 199–218, 2004.

[10] D. F. Ferraiolo and D. R. Kuhn. Role-based access control. *The National Computer Security Conference*, Baltimore, Maryland, USA, 554–563, 1992.

[11] S. R. Hussain, A. Sallam and E. Bertino. Detanom: detecting anomalous database transactions by insiders. *ACM Conference on Data and Application Security and Privacy*, 25–35, 2015.

[12] M. M. Javidi, M. Sohrabi and M. K. Rafsanjani. Intrusion detection in database systems. *Communication and Networking*, Springer, Berlin, Heidelberg, 93–101, 2010.

[13] A. Kamra, E. Bertino and G. Lebanon. Mechanisms for database intrusion detection and response. In *Proceedings of the 2nd SIGMOD PhD Workshop on Innovative Database ReSearch*, ACM New York, NY, USA, pp. 31–36, 2008.

[14] H. Kitano. Designing neural networks using genetic algorithms with graph generation system. *Complex Systems*, **4**, 461–476, 1990.

[15] S. Kumar and E. H. Spafford. *A pattern matching model for misuse intrusion detection*. Purdue University, Computer Science Technical Reports, Report Number: 94-071, 1994.

[16] A. Kundu, S. Sural and A. K. Majumdar. Database intrusion detection using sequence alignment. *International Journal of Information Security*, **9**, 179–191, 2010.

[17] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. *SIAM International Conference on Data Mining Society for Industrial and Applied Mathematics*, San Francisco, California, USA, 25–36, 2003.

[18] A. Makiou. Improving web application firewalls to detect advanced SQL injection attacks. *International Conference on Information Assurance and Security*, IEEE, Okinawa, Japan, 35–40, 2014.

[19] S. Mathew, M. Petropoulos, H. Q. Ngo and S. Upadhyaya. A data-centric approach to insider attack detection in database systems. *International Symposium on Research in Attacks, Intrusions and Defenses*, Ottawa, Canada, 382–401, 2010.

[20] H. Mazzawi, G. Dalal, D. Rozenblat, L. Ein-dor, M. Ninio and O. Lavi. Anomaly detection in large databases using behavioral patterning. *IEEE International Conference on Data Engineering*, San Diego, California, USA, 1140–1149, 2017.

[21] R. Mohan, V. Vaidehi, A. Krishna, M. Mahalakshmi and S. Ckakkaravarthy. Complex event processing based hybrid intrusion detection system. *IEEE International Conference on Signal Processing, Communication and Networking*, Chengdu, China, 1–6, 2015.

[22] D. J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. *International Joint Conference on Artificial Intelligence*, **89**, 762–767, 1989.

[23] S. Panigrahi, S. Sural and A. K. Majumdar. Two-stage database intrusion detection by combining multiple evidence and belief update. *Information Systems Frontiers*, **15**, 35–53, 2013.

[24] J. Parmar. A classification based approach to create database policy for intrusion detection and respond anomaly requests. *IEEE Conference on IT in Business, Industry and Government*, Indore, India, 1–7, 2014.

[25] C. Pinzon, A. Herrero, J. F. De, E. Corchado and J. Bajo. CBRid4SQL: a cbr intrusion detector for SQL injection attacks. *Hybrid Artificial Intelligent Systems*, **6077**, 510–519, 2010.

[26] Y. A. Rathod, M. B. Chaudhari and G. B. Jethava. Database intrusion detection by transaction signature. *IEEE International Conference on Computing Communication & Networking Technologies*, Karur, Tamilnadu, India, 1–5, 2012.

[27] C. A. Ronao and S. B. Cho. Anomalous query access detection in RBAC-administered databases with random forest and PCA. *Information Sciences*, **369**, 238–250, 2016.

[28] S. E. Salama, M. I. Marie, L. M. E. Fangary and Y. K. Helmy. Web anomaly misuse intrusion detection framework for SQL injection detection. *International Journal of Advanced Computer Science and Applications*, **3**, 123–129, 2012.

[29] R. J. Santos, J. Bernardino and M. Vieira. Approaches and challenges in database intrusion detection. *ACM Conference on Special Interest Group on Management of Data Record*, **43**, 36–47, 2014.

[30] N. M. Sheykhkanloo. Employing neural networks for the detection of SQL injection attack. *International Conference on Security of Information and Networks*, Sochi, Russia, 318–323, 2014.

[31] J. Skarus, J. P. Nowacki, A. Drabik, F. Serefynski and P. Bouvry. Soft computing techniques for intrusion detection of SQL-based attacks. *Asian Conference on Intelligent Information and Database Systems*, Yogyakarta, Indonesia, 33–42, 2010.

[32] G. Syswerda. Uniform crossover in genetic algorithms. *International Conference of Genetic Algorithms*, Morgan Kaufmann, Fairfax, Virginia, USA, 2–9, 1989.

[33] Transaction Processing Performance Council (TPC), TPC benchmark E Standard specification. *Version*, **1**, 0, 2014.

[34] F. Valeur, D. Mutz and G. Vigna. A learning-based approach to the detection of SQL attacks. *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Gothenburg, Sweden, 123–140, 2005.

Received 4 July 2018