Cluster Computing (2022) 25:3937–3959
https://doi.org/10.1007/s10586-022-03622-2

# Detecting malicious transactions in database using hybrid metaheuristic clustering and frequent sequential pattern mining

Rajni Jindal[1] · Indu Singh[1]

Received: 16 August 2021 / Revised: 22 February 2022 / Accepted: 3 May 2022 / Published online: 1 June 2022


**Abstract**
Database systems have become imperative for organisations around the world to store and analyse information. However, as one of the ramifications of a massive surge in cloud-based activities and interactions brought forth by the advent of the internet era, the data is exposed to an audience broader than ever and there are a variety of new challenges putting database security in jeopardy. To be able to address the problem of data security, we propose a unique method for Database Intrusion Detection System build on frequent sequential pattern mining and a modified metaheuristic hybrid clustering of Grey Wolf and Whale optimization algorithm to determine malicious transactions in Role Based Access Control and non-RBAC supervised databases. Our proposed approach extracts data dependency rules from the database logs using CM-SPADE mining algorithm to detect outsider threats. It then assigns role profiles to the users based on the previous user activities using the modified metaheuristic clustering to detect insider threats. Thereby, identifying incoming transactions as malicious by matching the role profile of the user and comparing the adherence of the transaction pattern to the generated dependency rules. To evaluate the efficiency of the model we generated a synthetic dataset including malicious and non-malicious transactions adhering to the TPC-C benchmark, and the findings were encouraging, with levels of accuracy of around 97.8 percent.

**Keywords** Database intrusion detection · Data dependency rule mining · Frequent sequential pattern mining · Hybrid metaheuristic clustering · Grey wolf optimization · Whale optimization

## 1 Introduction

In this modern era, data and information have become an indispensable part of any organization, and its importance can be seen in all phases of business activities; from product idealization to realization. According to some estimates, the total data stored across the world across all companies increases twofold in every 1.2 years [1]. This has led to the meteoric rise of database management systems as the main method to process information. With increasingly sensitive data and greater movement of operations online, the concerns for security against malicious attacks is at an all-time high.

Therefore, existing security measures like encryption, auditing, specific user level access and separation of powers have been adopted by numerous organizations to counter external threats. However, these approaches fail to account for malicious insider attacks. Hence, we must come up with novel techniques that are capable of simultaneously handling both outsider as well as insider threats [2]. A number of different measures have been implemented with reference to insider threats and cyber frauds according to the guidance offered in the Computer Emergency Response Team (CERT) Guide to Insider Threats [3]. The effectiveness of these existing systems can be enhanced through various rules, configurations and business processes.

According to the definition provided by CERT [3], an insider threat could be any employee, contractor or business partner that currently has, or in the past had authorized

✉ Indu Singh
indusingh@dtu.ac.in

Rajni Jindal
rajnijindal@dce.ac.in

[1] Delhi Technological University, Delhi, India

access to an organization's database infrastructure, and is willing to misuse this access to undermine the Confidentiality, Integrity or Availability of the organization's data [2]. Therefore, protecting data from unauthorized internal access is integral in maintaining the CIA triad.

The fundamental difficulty in identifying and preventing unauthorized internal access is that these are legitimate users with relevant access rights who are familiar with the internal architecture of the database schema as well as the security mechanisms put in place for its security. Therefore, insider threats can cause considerable damage to the system without being detected for extended periods of time. To maintain the integrity of data in such a complex environment requires high security standards in conjunction with a flexible intrusion detection system that evolves and adapts according to the threats encountered during operation.

In recent times there has been an unprecedented expansion in the usage of networked systems. This has resulted in the exposure of the database to quite a few vulnerabilities and threats. Thus, it is critical to shield the database against not only external malicious attacks but also against breaches of privileges by insiders who abuse the access rights granted to them. An intrusion can be described as any set of activities that make an effort to compromise the integrity, confidentiality or availability of a resource [4]. Although many different methods are used to protect important data under current network conditions, these methods are usually unsuccessful. The solution to recovering lost or damaged data without much difficulty in the event of a computer system getting compromised is early detection [5]. Confidentiality, integrity and availability (CIA) are the three key requirements that must be met for data protection.

One approach to reducing data vulnerability is to employ an Intrusion Detection System (IDS) in vital information systems. The main aim of intrusion detection systems is to identify/catch attacks against networks and information systems. This is because it is quite challenging to bestow information systems that are not only secure but also maintainable for their entire lifespan in such a secure and stable state [6]. Sometimes a fully secure computer system or network is not at all realized. One of the reasons for this could be operational constraints or legacy. Therefore, the job of an intrusion detection system is to track the operation of such systems and diagnose the existence of unsafe conditions. They identify active abuse and attempts by external parties or legitimate users misusing their rights or exploiting vulnerabilities present in the security system.

Preuveneers et al. [7] suggested that the use of federated learning systems for intrusion detection can improve the detection of malicious behavior by joining forces and pooling together monitoring data. Lee et al. [8] suggested Intrusion detection by using data mining techniques. Various sets of data mining algorithms like link analysis, sequential analysis and classification were considered. Barbara et al. [9] have suggested Intrusion detection by building a testbed using data mining techniques. Despite intrusion detection being a well-researched area, not much focus has been given on database intrusion detection. Besides, only two-fold judgment regarding the transaction is provided by existing database intrusion systems i.e. either the transaction is malicious or non-malicious. Although a system of the aforementioned category protects the database against threats satisfactorily, it results in unwanted system overhead and degradation of user experience [10] .

We can classify existing IDS into various categories depending on their properties [6]. The common ones are:

- Host-based Intrusion Detection Systems (HIDS): It monitors not only the incoming network packets but also the central file structure of the information system.
- Network Intrusion Detection Systems (NIDS): It consistently examines the network traffic and also the incoming transactions.

There exist other classifications which depend upon anomaly detection or signature detection and misuse.

- Misuse-based Detection: It identifies potential threats by looking for specific patterns similar to previously known threats. The main disadvantage of misuse-based detection is that it cannot identify new attacks whose patterns are not available.
- Anomaly-based Detection: It can identify and detect novel attacks which were not already known. It uses various machine learning algorithms to create a model which is then used to compare new transactions against that model. This model may suffer from false positives as the attacks are unknown.

For database security, the administrator creates preset roles with specified privileges and permissions, allowing only approved users to access the database using RBAC, or Role Based Access Control [11]. Users are given different levels of access depending on their position within the company. Users are given authority through the job processes and responsibilities that have been assigned to them, rather than directly. Instead of controlling individual user access and privileges, they are centralised across a network and allocated to a set of roles which assures database security.

This technique works by capturing sequences [2] and mining association rules [3] from transaction data, which is then used to identify an incoming transaction as legitimate or malicious. By limiting user activity to certain roles, the NIST model [11] further enhanced the security system's effectiveness. To tackle insider attacks, the RBAC

mechanism is widely being implemented [12]. We propose a novel approach (FPGWWO) which is capable of handling not only external attacks but internal attacks as well. Data dependency rules of the legitimate queries are generated using the frequent sequential pattern mining algorithm CM-SPADE that assists in inspecting the resemblance of the incoming transactions. For calculating the adherence between the incoming intra-transactional query and the set of rules mined we introduce the concept of similarity threshold. This similarity threshold known as Congruity Index is used to classify the incoming transaction as either malicious or non-malicious. To maintain the integrity of the Database Intrusion Detection System (DIDS) from insider attacks, we employ a modified hybrid of grey wolf and whale optimization metaheuristic clustering algorithm (mhGW-WOA). This algorithm takes in the user transactions as a parameter for the objective function and constructs role profiles based on previous user activities.

During the detection phase, a role matcher authenticates role clusters for the incoming transaction. If the role profile is not matched, the transaction is labelled as malicious and aborted. This module renders our approach immune to internal attacks. Subsequently, the transaction is forwarded to the rule validator if the role profile matches, which then compares the incoming query to mined rules produced during the learning phase by computing the similarity threshold, i.e., the "congruity index". Transactions are labelled malicious based on this threshold and an alarm is triggered and the database executes a rollback. Consequently, any external or internal attacks on the system that could lead to alterations will be prohibited.

The major contributions of this work are as follows:

1. *We present a comprehensive DIDS that uses a rule mining module to prevent external threats and a role clustering analyzer to assess transaction validity for preventing insider attacks and detecting intrusions in the database.*
2. *We introduce a "modified hybrid of Grey Wolf Optimizer with Whale Optimisation Algorithm (mhGW-WOA)" to cluster role profiles based on user activity for matching the roles of incoming transactions. We also employ frequent sequential pattern mining technique i.e. CM-SPADE to analyze previous user patterns in the database logs to obtain read and write rules based on the legitimate user access patterns.*
3. *We propose a novel similarity threshold, called the Congruity Index which calculates the adherence between the dependency rules and the extracted transactions. On the basis of this index, the transactions are classified into malicious or non-malicious.*

The remaining paper is organised as follows. Section 2 provides an overview of the related work. The proposed methodology is described in Sect. 3. Section 4 contains the experimental results as well as a comparison to previous methodologies. Finally, in Section 5, we offer our conclusion.

## 2 Related work

There are various Intrusion Detection System Models [13–15] which have been proposed for detecting intrusions at the network-level and OS-level. However, very few models have been implemented to identify database intrusions. There is a need for database IDSs to ensure three goals [2] :

1. Confidentiality - to protect data against unauthorized exposure.
2. Integrity - to prevent malicious data modifications
3. Availability - to protect against hardware or software failures, as well as malicious data accesses, and to recover from them.

Any set of actions which hamper these goals is termed an intrusion [4]. Since the beginning of the 21st century, concrete progress has been made in IDS[5, 16, 17] for detecting database intrusions with the help of data mining and data dependency rules. D.E. Denning [18] states that on inspection when a system's records are searched for unusual patterns, security violations can be attributed to malicious activity. Their model is based on a rule-based pattern matching system to monitor standard OS operations such as logging in, command and program executions, disk accesses, etc., looking only for deviations in usage. Despite the unique approach, at the local level, the system has great problems due to its weakness for the inadequate resolution of complex procedures. According to research, efforts to identify unauthorized usage of software applications by users [19] using an increasing time frame for user profile training and abstracting into groups of apps reduces the amount of false alarms generated considerably. Cardenas et al. [20] concluded that while control systems are pivotal, they are quite likely to be targeted by skilled and motivated hackers due to increasingly exposed vulnerabilities. In fact, the security mechanism for detecting and finding controlled data changes before hackers compromise the system was clear and in use. Different norms were depicted by Debar et.al. [6], with focus on the internal dynamics of the elements, and their corresponding behaviors and principles. The model aggregated the recommendations of many evaluation indicators to assess the accuracy of efficiency and utilise the chance variation approach in the process. Legitimate users' patterns of access are compared against those kept in a database. Bertino et al. [2] concentrated on access control systems and discussed the major access

control types. Due to the disturbing additions in security dangers and systems administration, Liao et al. [21] established that intrusion detection systems(IDS) have been conclusive and pivotal in the world of computation.

## 2.1 Data dependency mining

Data dependency mining [22] is highly effective in discovering patterns and dependencies in large data sets. It employs statistical methods to analyse the data sets and detects relationships amongst the attributes. It is immensely useful in data rectification, extracting regularities from datasets and resource filtering. This section reviews some of the related works that delve into the potential of this method in intrusion detection.

As Chen et al. [23] established, data dependency mining is given significant importance in a variety of research fields including, but not limited to, database systems, machine learning, statistics, information management and artificial intelligence. Various state of the art data mining methods are applied to gain insights into user behaviour and access patterns which has enhanced opportunities in diverse fields.

[5, 16] highlighted the use of data dependency mining in detecting database intrusions as opposed to the existing methods that used the operating system log or the application log. Before a given data item is updated, other data items are written or read from the dataset. The model proposed by Hu et al. [16] provides strategies for detecting these data dependencies and employs Petri-Nets to model normal data patterns. They further leveraged this in mining data dependencies [5] from database logs. The transactions that did not meet the criteria for the mined dependencies are flagged as malicious. While the results were promising, they did not take into account the different sensitivities of the attributes of the database.

Srivastava et al. [17] proposed an approach using weighted sequence mining to detect the dependencies that considered the varying sensitivity of the attributes. The algorithm performed better than the previous algorithms in detecting changes in sensitive attributes.

Hashemi et al. [24] furthered the existing approaches by introducing a behaviour similarity criterion to constrain the false positive rate. Besides the transactions that do not satisfy the data dependencies, the transactions which seemed identical to these invalid transactions in the log in this similarity test were also flagged as malicious. The proposed approach detecting anomalies in a time series dataset is able to identify malicious transactions even when they confirmed to data dependency rules, thereby yielding a much better true positive rate.

Rahman et al. [25] worked on an algorithm centred around pattern mining. They used weighted uncertain sequence mining to detect sequences or patterns in uncertain databases. It was effective and efficient in in mining frequent weighted sequences. It is also suitable for real life databases because real world data tends to be uncertain. The main obstacle they encountered was recognising the more significant valid patterns according to their relevance.

Kundu et al. [26] tried to tackle the problem of identification appropriate support and confidence values with attribute dependency mining. They suggested an intrusion detection system that leveraged sequence alignment which allows for a wide range of transactional and profile granularity levels to be selected. It gave more efficient results than the existing methods.

Subudhi et al. [27] introduced an intrusion detection system (IDS) that used OPTICS clustering in conjunction with Ensemble Learning, which included numerous aggregation methods such as Boosting, Bagging, and Stacking, to identify malicious users in a database. They divided intrusion detection into 2 phases, namely, training and testing. In the training phase, the input dataset attributes were preprocessed. Subsequent to that, OPTICS clustering was applied to it to develop behavioural profiles. During the testing phase, a new transaction was reviewed against these profiles to check for conformity. If it turned out to be deviant from the profiles, it was checked for maliciousness by an ensemble learner.

Sallam et al. [28] used an anomaly detection system to discover data aggregation and data updation. Their method involved using a standard table reference rate and retrieving tuples from previous database access logs. In addition, the incoming user requests were evaluated to see if any of them were likely to exceed the regular data access rates.

## 2.2 Sequence pattern mining

First coined by Agrawal et al. [29], a data mining subdomain, namely, Sequential Pattern Mining (SPM) has extensive applications in the field of database security to detect recurrently occurring sequences, intriguing features and patterns in sequential databases.

Agrawal and Srikant [29] introduced AprioriAll and AprioriSome algorithms. These algorithms can be utilized to extract sequential patterns from a database. Both algorithms grow linearly as the size of the database increases but face the same challenge of determining the support and confidence worths. To overcome these issues, Agrawal and Srikant [30] put-forth a new algorithm which generalizes the SPM algorithm and provides more efficient performance than the Apriori algorithms. In this procedure, the cost or profit of the item that is being used is taken as the weight while also taking into account the unique or distinct sign of all the items in real-time application.

Zaki et al. [31] proposed a novel algorithm called SPADE which divides the original problem into smaller sub-problems using equivalence classes on recurrent sequences. A major advantage is that each equivalence can be analyzed individually and processed in the main-memory. Also, unlike other approaches which take multiple scans, SPADE uses at most three database scans and only one scan if 2-sequence instruction is supported. It outperforms the AprioriAll algorithm by at-least a factor of 2 with 2-sequence instruction support. Similarly, Pei et al. introduced a prefixspan approach [32] with the motive of searching for necessary sequences by looking for the prefix sub-sequences and projecting just their corresponding postfix sub-sequences into projected databases. It also outperforms the Apriori algorithms. However, for large datasets the SPAM algorithm proposed by Ayres et al. [33], which uses a depth-first traversal process coupled with a vertical bit-map representation to store every sequence allowing noteworthy bitmap compression along with an efficient support counting, is considered more efficient. Although, the SPAM algorithm has a major limitation as it consumes more space in comparison to PrefixSpan or SPADE algorithm.

Gomariz et al. [34] developed a new close constraint algorithm called ClaSP which was tested of the gazelle dataset, can be used to mine frequent close sequential pattern utilizing various search-space pruning methods combined with a vertical database layout. Similarly, the CMAP algorithm introduced by Viger et al.[35] is based on the CMAP structure that can be built within a single database scan and utilizes co occurrence-based pruning. It outperforms algorithms like Apriori, SPADE, PrefixSpan, SPAN as well as close-sequential patterns like ClaSP by 8 times post-integration with them. Upon testing on six real-life datasets, it is found that integration with ClaSP and SPADE provides best results.

Lan et al. [36] devised a different method in order to retrieve weighted sequential patterns. Here, the maximum weighted upper bound model was taken into account which resulted in higher accuracy and efficiency for the subsequences. The major disadvantage of this model was its failure to handle the updating or removal of sequences and dynamic addition.

Rahman et al. [25] proposed the concept of pattern mining where sequences are selected, via an algorithm, from uncertain databases with weight restrictions. The main challenge is to distinguish between important and valid patterns based on significance. In 2019 Rahman et al. [37], devised a method to overcome this downside by utilizing weight and support constraints in order to build patterns in uncertain databases.

## 2.3 Anomaly detection

Chung et al. [38] developed a DEMIDS abuse detection system, customized to relational database systems. DEMIDS creates profiles based on audit records that reflect normal user behavior while dealing with the DBS. Although DEMIDS may be used to identify both intrusion and insider abuse, it focuses on detecting anomalous activity by authorised individuals who misuse their rights. As a result, the system is especially beneficial for internal control.

Spalka et al. [39] presented a framework for database expansion and user interaction with a DBMS, as well as a database misuse detection system. They examine two techniques to handle with database extension, first relying on measured value and another based upon $\Delta$-relations, in a detailed examination, and show that previously conventional statistical functions produce acceptable detection results. These reference data can assist discover a number of abnormalities in primarily developing relationships. Anomaly detection in user interactions relies heavily on reference data.

The Collaborative Adversarial Network (CAN) model, proposed by Alzubi et al. [40], is a collaborative network of generators that is pitted against a discriminator adversarial network.It entails the segmentation of frequent highlights between two phrases and the use of community-oriented learning to traditional ill-disposed and adversarial learning for the extraction of common features which can be used for the calculation of similarity between rules and queries to efficiently detect anomalous transactions.

In order to identify intrusions in databases, Hashemi et al. [24] suggested a data mining technique. This technique is based on (1) mining dependencies, (2) identifying aberrant update trends, and (3) comparing normal transactions. The first and second characteristics are intended to improve the rate of genuine positive detection. The third feature is designed to lower the number of false positives. The proposed approach's main advantages are (1) its ability to identify interruption transactions, (2) its capacity to spot malicious activities, and (3) its potential to classify transactions as regular also when they violate the addiction rules.

Alzubi et. al. [41], proposed a CNN and LSTM based two-layer architecture for image captioning, a custom form of which can be used for the captioning of queries in the form of signals as anomalous or non-anomalous queries.

Kamra et al. [42], suggested a method for identifying unusual access patterns in database management systems. They have created three models, each with a different level of granularity, to describe SQL queries seen in database log files. The FP rate for clustering methods based on k-means

and k-centers is exceptionally low. The FN ratio for k-means was significantly higher than other algorithms. Either of the clustering techniques, the findings for the outlier identification methodology was not particularly outstanding.

For threat detection, Panigrahi et al. [43] developed architecture that employs both intra-transactional and inter-transactional methods. For assessing the transaction provided by a customer, sequence alignment and spatio-temporal outlier identification were used. Transaction was categorised as regular, unusual, or suspicious. Analyses were conducted on a vast number of simulated databases and the simulation produced a TP of approximately 98% and an FP just under 10%. Dempster-Shafer theory assist in True positive and Bayesian learning can aid to minimize the frequency of false alarms even further.

When an application makes a query, The relevant profile and restrictions were compared to the platform's current scenario in the DetAnom identification process[44]. The query was tagged as unusual if there was a difference. The major benefit in this technique was that it doesn't require any prior information of the system nor examples of prospective attacks in order to create system profiles.

Sallam et al. [45] proposed a new technique called DBSAFE for anomaly detection in databases that use the RBAC(role-based access control model). The procedure requires recording training information during either normal operation or by using the audit log. Query features issued by each role are then used to denote the access profiles. Analyzing the syntactic features of queries provides the access patterns. It assumed only one role activated per user. A major extension of the technique was proposed by Sallam et al.[46] which considered common access patterns between the roles. It uses NBC and MLC classifiers for RBAC models and unsupervised anomaly detection when role information is unavailable. However, these alone prove to be insufficient. Therefore, Sallam et al. [47] proposed another extension of his previous work. It uses a three-stage process for collecting profile information. It uses a standard periodicity detection algorithm to detect the frequency corresponding to which queries in logs are regularly issued after which the expected time-interval in which periodic queries are expected to be issued is determined. The final stage recognizes the relationships among periodic queries that are issued together. A query is marked anomalous if it is received at an unexpected time. However, a major limitation of this method is that it is still unable to process aperiodic queries.

Ronao et al. [48] proposed random forest with weighted voting (WRF) and principal component analysis (PCA) as a feature selection approach for detecting database access anomalies, provided that the database employs a role-based access control (RBAC) model. PCA generates uncorrelated and meaningful characteristics while also reducing dimensionality for easier integration with big databases. RF takes advantage of SQL queries' inherent tree-structure syntax, and its weighted voting mechanism reduces false alarms even more.

They suggest utilizing CNN-LSTM neural networks to classify a device's status and authorization by obtaining characteristics from SQL server [49]. CNN derives significant characteristics from queries autonomously, while LSTM analyzes the SQL sequence's temporal features from TPC- E benchmark.The authors also used statistical analysis to determine the features of misclassification data.

### 2.4 Metaheuristic clustering

Mahalingam et al. [50] suggested a dynamic clustering technique by breaking up the moving object Optimal Background separation using Optimal Weighted Centroid (OWC) - Modified Kernel Fuzzy C Means Algorithm (MKFCM) for information clustering utilizing the OWC (Optimal Weighted Centroid) [50] .The effectiveness of the suggested procedure is inspected with going before procedures k-NN just as MLP concerning precision, f-measure, ROC just as review.The effectiveness of the suggested procedure is inspected with going before procedures k-NN just as MLP concerning precision, f-measure, ROC just as review.

Rathore et al. [51] proposed a hybrid whale and grey wolf optimisation (WGWO)- based bunching component for energy gathering remote sensor organizations (EH-WSNs). The exploitation and investigation abilities of the proposed mixture WGWO approach are a lot higher than the conventional different existing metaheuristic calculations during the assessment of the algorithm.

Ali Akbar et al. [52], proposed a novel artificial neural network(ANN) training algorithm which utilizes the power of invasive weed optimization and differential evolutionary model to find the optimal weights of the ANN. This proposed training mechanism is proven to be more efficient and less error-prone and thus can be utilized as a classifier to detect intrusive transactions in databases.

Rahnema et al. [53] used a hybrid of improved artificial bee colony and whale optimization algorithm for data clustering. This hybrid solves the issue of late convergence and exploration of ABC by using Random Memory and Elite Memory. The former in the ABCWOA algorithm has utilized the search stage as a bait in the whale optimization algorithm whereas the latter is used to increase convergence. This combination performs better than other metaheuristic algorithms.

Aljarah et al. [54], proposed a novel grey-wolf inspired clustering approach and enhanced it using the Tabu Search(TS) algorithm whereas Ghany et al.[55] proposed a

modified Hybrid Whale algorithm with Tabu Search for Data Clustering. In [54], the GWO is used to deal with problems related to clustering while the TS algorithms behaves like a local search hill-climbging algorithm. However, TS accepts non-upgrading solutions to escape from local optima. The main intent behind the hybrid GWOTS is to use the process of adaptive memory in TS for exploring the neighbourhood of each leader before extracting the updated positions of the remaining pack. GWOTS tries to find the optimal solution centroid for each group in such a manner that each member wolf in GWOTS represents a solution containing clusters / groups of centroids. This hybrid approach was run on 15 datasets out of which it showed significantly great results on 8 datasets. Contingent on these outputs, it can be concluded that the effectiveness of the GWOTS is highly competitive and comparatively better than usual methods in most cases. The primary rationale is that it can establish an equilibrium between main exploitative and exploratory trends and in the scenario of finding any high-quality solution; it is locally-informed and aggravates the exploitation around that position, which drives to increase the level of the quality. While in [55], WOA uses optimal solutions to relocate swarm members. In order to overcome this problem, WOA needs to retain multiple optimal solutions within the scope of the solution. While the Tabu Search(TS) algorithm being a meta-heuristic method, uses memory components to exploit and explore search space. They presented WOATS for data clustering in [55], which uses an objective function inspired by partitional clustering to keep clustering solutions of high quality. The efficiency of WOATS is measured using publicly available real-world data sets. These data sets vary in size, from small data sets to large data sets, demonstrating their advantages over a variety of primitive and hybrid swarm intelligence technologies. The results obtained by WOATS are significantly better than other methods, ensuring its effectiveness. Each whale in the group symbolizes a complete solution in WOATS (Group Center). These centers are selected based on the best value of the objective function. WOATS always achieves the best objective function value with the least number of iterations-for all data sets, compared with other methods, WOATS achieves the smallest objective function value with the least number of iterations, and the number of iterations is less than 50. WOATS is designed to pool huge biological data sets based on big data technology.

## 3 Proposed methodology

We propose a Database Intrusion Detection System (DIDS) that uses rule mining and clustering in the initial training phase and then followed by the classification in the detection phase. The proposed approach FPGWWO mines the read and write dependency rule subsequently followed by the modified Hybrid of GreyWolf Optimizer with Whale Optimisation Algorithm (mhGW-WOA) clustering using the current database logs containing transactional details. The pertinent patterns generated by the algorithm prevents undesirable rules from being created. In the first part of the learning phase, we make use of the CM-Spade [35] algorithm to generate sequential patterns succeeded by generating data dependencies that occur from the preprocessed transaction logs. Classification rules above certain confidence values are generated pertaining to the mined data dependencies. In the second part of the learning phase, role profiles (clusters) are generated from defined parameters after analysis of database logs using (mhGW-WOA) clustering. It is assumed that only legitimate transaction logs and database logs are provided in this phase.

### 3.1 System architecture

The proposed DIDS functions as a separate unit, enhancing system security without interfering with the database management system (DBMS). The system is designed to identify and abort malicious transactions sent by attackers to the DBMS, which managed to bypass the initial security of the database system. The scenario may arise due to a compromised legitimate account, web portal susceptible to SQL injection, abuse of user privileges [56]. In all these cases, an attacker can access the database by submitting transactions manually or through an application. The architecture of the proposed system (FPGWWO) consists of two modules:

- Learning Phase
- Detection and Classification Phase

### 3.2 Learning phase

The architecture of the learning phase is represented in Fig. 1. In this phase, the rule generator uses transaction logs to extract data dependencies, and then stores them for resemblance calculations in subsequent stages. While assessing resemblance, database records are also used to analyze user access patterns, and then these patterns are grouped into role profiles. These role profile groups and generated rules are then passed to the discovery phase. The rules and role clusters formed are then used in the detection phase. The learning module has two vital units:

- Rule mining
- mhGW-WOA Clustering

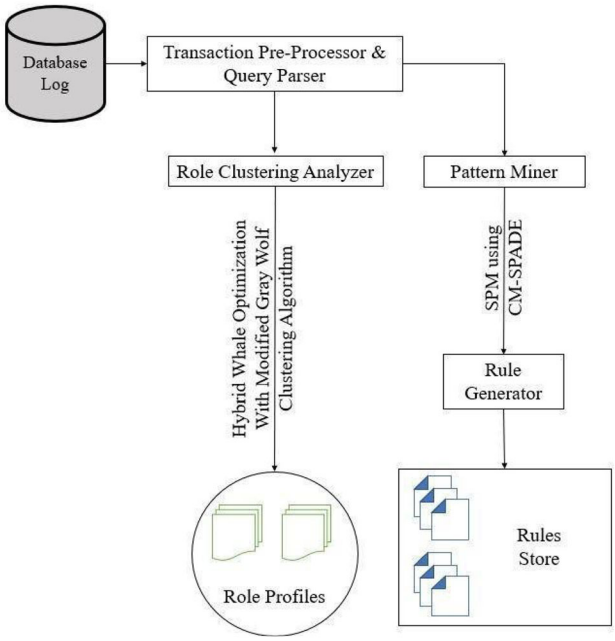


**Fig. 1** Learning phase architecture

### 3.2.1 Rule mining

Rule mining refers to the discovery of patterns that indicate the basic and important characteristics of the database, then preprocess the appropriate pattern derived from the algorithm in a way that excludes the generation of undesired rules. In the first part of the learning phase, we make use of the CM-Spade [35] algorithm to generate sequential patterns succeeded by generating data dependencies that occur from the preprocessed transaction logs. The classification rules that are higher than the defined confidence threshold are generated with respect to the mined data dependencies.

**Query parser** The SQL queries from the transaction table cannot be categorized as malicious or non-malicious as such. In order for our model to classify the transaction, we use a query parser which takes the SQL queries as input, parses and analyses them to convert to appropriate read and write transaction sequences. Each transaction is assigned a unique transaction ID. The commands are executed for tokens taken from the source token list. If no error occurs when parsing the input into a read-write stream, a parse tree is generated. The generation of the parse tree proceeds recursively until no error occurs while parsing the input into read and write rules.

For example,

Consider a SQL query as:

Account_id = select ID_no from Account where name= 'Alice' and IFSC='ASCR4112';

update Loan set amount=amount + 1500 where Loan_id = Account_id;

Output generated by query parser:

Tid = <R(name), R(IFSC), R(ID_no), W(Account_id), R(Loan_id), R(Account_id), R(amount), W(amount)>

**Transaction pre-processor** The module specifies the identified transactions in an ideal way. Each transaction can be expressed as a set of queries (select, insert, delete or update). Then, each query can be expressed as a series of operations (read or write) on the item set. The items in the item set are arranged in lexicographic order to avoid multiple representations of the same query.

- Each transaction starts with the beginning transaction and ends with commit/rollback.
- We denote a transaction T as $< q_1, q_2, q_3, \cdots, q_n >$ where $q_i \in Q(s).Q(s)$ is the set of queries.
- We denote a query $q_i$ as $< op_1(D_1), op_2(D_2), op_3(D_3), \cdots, op_n(D_n) >$. Where $op_i \in$ [R,W] and $D_i$ represent an item-set $< d_1, d_2, \cdots, d_k >$ lexicographically sorted.
- Transactions are expressed as queries in order. For each of these queries, the sub-query (if any) is expressed first. Clauses like while, order by, etc. they are evaluated before the SQL statement type (select, update, insert, or delete).

**Sequential patterns** A sequence is an ordered list of operations. Sequential patterns are those elements that appear multiple times in the same order simultaneously in the data. Data correlation is symbolized as a read stream and a write stream.

**Definition 1** (*Transaction*): A transaction is an orderly arrangement of operations performed on a data-base instance, with the user operation denoted by Op ∈ Read, Write. For the transaction to be completed, the operations must be executed in a discrete manner. TIDs are issued to transactions as a unique identifier.

**Definition 2** (*Operation O(a)*): O(a) represents the relationship with the action of a data item a. It comprises action to the set Read,Write.

**Definition 3** (*Sequence*): It is an ordered set of read, write operations with respect to time, which is defined as: $Seq = \{O_1(a_1), O_2(a_2), \cdots, O_n(a_n)\}$ where Oi ∈ R,W and $a_i$ represent an attribute of a data item. There are two categories of data dependency sequences extracted from transaction logs:

**Definition 4** (*Read Sequence RdSq(a)*): For a particular attribute a, read sequence is defined in the format: $RdSq(a) = \{R(a_1), R(a_2), \in, R(a_n), O(a)\}$ where $a_1, a_2, \cdots, an$ are the attributes that are to be read before a transaction performs any operation O ∈ R, W on the attribute a and R($a_j$ ) denotes a quantum unit of read operation performed on $a_j$.

**Table 1** Transactions for Mining Sequential Patterns

| Transaction ID | Transaction |
|---|---|
| 20 | $<r10, r15, r11, w13, r16, r12, w15, r10, r15, w11, r12, r14, r11, w14>$ |
| 21 | $<r15, r10, w12, r10, w15, r11, r16, r13, w11>$ |
| 22 | $<r16, r10, r15, w14, r10, w13>$ |
| 23 | $<r11, w11, r13, r16, w12, r15, w14, r10, w13>$ |
| 24 | $<r11, r14, w15>$ |
| 25 | $<r10, r14, w10, r13, r14, w13>$ |
| 26 | $<r13, w15>$ |
| 27 | $<r15, r14, w14, r12, r13, w13, r12, w16>$ |
| 28 | $<r14, r12, r15, w16>$ |
| 29 | $<r14, r15, w14, r14, w13>$ |

**Definition 5** (*Write Sequence WrSq*(*a*)): For a particular attribute a, write sequence is defined in the format: $WrSq(a) = \{O(a), W(a_1), W(a_2), \cdots, W(a_n)\}$ where $a_1, a_2, \cdots, a_n$ are the attributes that are to be updated after a transaction performs operations O ∈R, W on the attribute a and W($a_j$ ) denotes a quantum unit of write operation performed on $a_j$ .

**Definition 6** (*Sensitive elements*): They are defined as those elements which are either updated in the transaction or belong to restricted access groups. Sensitivity of an element is determined by its weight in the transaction.

**Definition 7** (*Confidence*) The confidence of a given sequence (read or write) is the ratio between the weighted count of occurrences of the given sensitive element $a_i$ and the weighted count of occurrences of sensitive element $a_i$. We have given a formal definition of confidence for a read sequence below. The confidence level for the script sequence is similar.

$$Confidence(RdSq(a_i)) = \frac{CountRdSq(a_i)}{Count(a_i)} \quad (1)$$

**Definition 8** (*Read Rules* (*RR*)): For every read sequence $\{R(a_1), R(a_2), \cdots, R(a_n), Op(a)\}$, that is generated using sequential pattern mining, we generate a read rule of the format $<R(a_1), R(a_2), \cdots, R(a_n)> \rightarrow O(a)$, that implies, before a, we must read $a_1, a_2, \cdots, a_n$.

A rule is added to the Read Rules set only if the rule that is generated from the read sequence has a confidence value greater than the minimum confidence provided.

**Definition 9** (*Write Rules* (*WR*)): For every write sequence $\{Op(a), W(a_1), W(a_2), \cdots, W(a_n)\}$, that is generated using sequential pattern mining, we generate a write rule of the format $O(a) \rightarrow <W(a_1), W(a_2), \cdots, W(a_n)>$, that implies, after updating a, data items $a_1, a_2, \cdots, a_n$ must be updated during the transactions. A rule is added to the Write Rules set only if the rule that is generated from the write sequence has a confidence value greater than the minimum confidence provided.

Table 1. displays the read and write items in a transaction. Each of these transactions are assigned a unique transaction ID.

**Definition 10** (*Prefix*): Given two transactional sequences $a = \{Op_1(it_1), Op_2(it_2), \cdots, Op_n(it_n)\} and b = \{Op_1(it_1), Op_2(it_2), \cdots, Op_m(it_m)\} (m < n)$, sequence b is called a prefix of sequence a $\iff Op_i(it_i) = Op_i o(it_i o) \forall i \in \{1, 2, \cdots, m\}.T$

**Rule generator** The process of extracting relevant rules from the data mainly involves the extraction of frequent sequence patterns through successive algorithms, because the transactions have previously been processed in an ordered set of queries. For this, we use the CM-SPADE algorithm to extract sequence patterns from the transaction log. CM-SPADE was chosen because it outperforms the most advanced algorithms in mining sequential patterns (GSP, PrefixSpan, SPADE, and SPAM) and closed sequential patterns (ClaSP and CloSpan)[35].

**Definition 11** (*Sequential Pattern Mining*):
Consider SDB a sequence database and minsup as threshold. A sequence s is considered a sequential pattern iff supSDB(s) ≥ minsup.

Table 2 displays the number of frequent sequential patterns mined using transaction table (Table 1). Patterns with support value higher than the specified minimum support are added to the set of Sequential Patterns. We have taken minsup = 5.

**Table 2** Mined sequential patterns

| Sequential patterns |
|---|
| *r*15, *w*14, *w*13 |
| *r*16, *r*15, *r*10 |
| *r*16, *r*15, *w*13 |
| *r*16, *r*15, *w*14 |

**Definition 12** (*Vertical Database Format*): Vertical Format Sequence Database SDB is a database, where each entry represents an item and indicates the sequence list in which the item appears and where it appears [31]. Table 3 shows the vertical representation of the database of Table1.

**Definition 13** (*i-extension*): An item q is said to succeed by i-extension to an item j in a sequence $I_1, I_2, \cdots, I_n$ iff j, q $\in I_x$ for an integer x such that $1 \leq$ x $\leq$ p and q $>$ lex j.

**Definition 14** (*s-extension*): An item q is said to succeed by s-extension to an item j in a sequence $I_1, I_2, \cdots, I_n$ iff j$\in I_n$ and q$\in I_m$ for some integers n and m such that $1 \leq$ n$<$m $\leq$p.

**Definition 15** (*CMAP*): A Co-occurrence MAP (CMAP) is a structure mapping each item q$\in$ I to a set of items succeeding it. We define two $CMAP_s$ named $CMAP_i$ and $CMAP_s$. $CMAP_i$ maps each item q to the set cmi(q) of all items j $\in$ I succeeding q by i-extension in no less than minsup sequences of SDB. $CMAP_s$ maps each item q to the set $cm_s(q)$ of all items j $\in$ I succeedings q by s-extension in no less than minsup sequences of SDB.

The algorithm generates data dependency rules after the generation of frequent sequences which are transformed to read and write sequence. Read rules are denoted in the format: $<R(a_1), R(a_2), \cdots, R(a_n) > \rightarrow O(a)$ which gives the attributes that are to be read before any operation is performed on an attribute a. Write rules are denoted in the format: $O(a) \rightarrow <W(a_1), W(a_2), \cdots, W(a_n) >$ which gives the attributes that are to be updated after any operation is performed on an attribute a. If the confidence value of any read or write rule generated is greater than the minimum confidence provided, it is added to the rule set.

Table 4 shows the number of data dependency rules generated in the set $\in$ Read, Write in accordance to the sequential pattern generated from Table 2. Confidence value for each rule is shown which is maintained above the specified minimum confidence.

**Algorithm 1:** Dependency Rule Miner

```
Input : DB: Initial Database
        minsup: Minimum Support
Result: R: Set of Sequential Patterns
1  Begin:
2    Scan DB to create Vertical(DB) and identify F
3    the list of frequent items
     Enumerate(F) for each pattern A_i ∈ F do
4      Output A_i.
5      T_i ← φ for each pattern A_j ∈ F, with j ≥ i
       do
6        R ← MergePatterns(A_i, A_j)
7        for each pattern r ∈ R do
8          a ← last element in r
9          if r is i-ext then
10           if sup(CMAP_i ≥ minsup) then
11             T_i ← T_i ∪ {R};
12           else if r is s-ext then
13             if sup(CMAP_s ≥ minsup
                then
14             | T_i ← T_i ∪ {R};
15             else if ∄x : x ∈ cm_i(a) AND
16             ∄x : x ∈ cm_s(a) then
17             | continue;
18             end
19           end
20         end
21       end
22     end
23   end
24 End
```

In line 2 of the algorithm 1 the initial database is scanned to create a vertical representation of the database and identify the list of frequent items F. From line 3 the enumerate function is initialized. From line 3 the loop iterates over each element $A_j$ from frequent list F. In step 6, the $T_i$ is initialized to an empty set. From line 7 a nested for loop iterates over each element $A_i$ from frequent list F with j greater than equal to i. Inside the nested loop pattern, $A_i$ and $A_j$ are merged and saved to variable R in line 8. From line 9 to line 18, the for loop iterates over the merged patterns R and checks if the pattern is either i-ext or s-ext and its support is greater than or equal to the minimum support and adds the respective pattern to $T_i$. In line 19 the Enumerate function is called recursively with frequent list $T_i$.

### 3.2.2 Role clustering analyzer

We use a role clustering method to detect the insider threats which would be otherwise left undetected. In our model (FPGWWO) we use a hybrid meta heuristic clustering algorithm to achieve the same. This hybrid module has two components namely Grey Wolf Optimizer (GWO) and Whale Optimizer (WO).These are defined as follows:

**Grey wolf optimizer (GWO)** Mirjalili et al. [12] introduced a new method based on swarm intelligence

**Table 3** Vertical representation of Table 1

(a) Read rules

| *r*10 | | *r*11 | | *r*12 | | *r*13 | | *r*14 | | *r*15 | | *r*16 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TID | Item sets | TID | Item sets | TID | Item sets | TID | Item sets | TID | Item sets | TID | Item sets | TID | Item sets |
| 20 | 1,8 | 20 | 3,13 | 20 | 6,11 | 20 | | 20 | 12 | 20 | 2,9 | 20 | 5 |
| 21 | 2,4 | 21 | 6 | 21 | | 21 | 8 | 21 | | 21 | 1 | 21 | 7 |
| 22 | 2,5 | 22 | | 22 | | 22 | | 22 | | 22 | 3 | 22 | 1 |
| 23 | 8 | 23 | 1 | 23 | | 23 | 3 | 23 | | 23 | 6 | 23 | 4 |
| 24 | | 24 | 1 | 24 | | 24 | | 24 | 2 | 24 | | 24 | |
| 25 | 1 | 25 | | 25 | | 25 | 4 | 25 | 2,5 | 25 | | 25 | |
| 26 | | 26 | | 26 | | 26 | 1 | 26 | | 26 | | 26 | |
| 27 | | 27 | | 27 | 4,7 | 27 | | 27 | 2 | 27 | 1 | 27 | |
| 28 | | 28 | | 28 | 2 | 28 | 5 | 28 | 1 | 28 | 3 | 28 | |
| 29 | | 29 | | 29 | | 29 | | 29 | 1,4 | 29 | 2 | 29 | |

(b) Write rules

| *w*10 | | *w*11 | | *w*12 | | *w*13 | | *w*14 | | *w*15 | | *w*16 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TID | Item sets | TID | Item sets | TID | Item sets | TID | Item sets | TID | Item sets | TID | Item sets | TID | Item sets |
| 20 | | 20 | 10 | 20 | | 20 | 4 | 20 | 14 | 20 | 7 | 20 | |
| 21 | | 21 | 9 | 21 | 3 | 21 | | 21 | | 21 | 5 | 21 | |
| 22 | | 22 | | 22 | | 22 | 6 | 22 | 4 | 22 | | 22 | |
| 23 | | 23 | 2 | 23 | 5 | 23 | 9 | 23 | 7 | 23 | | 23 | |
| 24 | | 24 | | 24 | | 24 | | 24 | | 24 | 3 | 24 | |
| 25 | 3 | 25 | | 25 | | 25 | 6 | 25 | | 25 | | 25 | |
| 26 | | 26 | | 26 | | 26 | | 26 | | 26 | 2 | 26 | |
| 27 | | 27 | | 27 | | 27 | 6 | 27 | 3 | 27 | | 27 | 8 |
| 28 | | 28 | | 28 | | 28 | | 28 | | 28 | | 28 | 4 |
| 29 | | 29 | | 29 | | 29 | 5 | 29 | 3 | 29 | | 29 | |

**Table 4** Data dependency Rules

| Sequential Pattern | Data Dependency Rule | Confidence |
|---|---|---|
| *r*16, *r*15, *w*14 | *r*15, *w*14 → *r*16 | 60% |
| *r*16, *r*15, *w*13 | *r*15, *w*13 → *r*16 | 60% |
| *r*16, *r*15, *w*14 | *r*16, *r*15 → *w*14 | 75% |
| *r*16, *r*15, *w*13 | *r*16, *r*15 → *w*13 | 75% |
| *r*16, *r*15, *r*10 | *r*10 → *r*16, *r*15 | 80% |
| *r*15, *w*14, *w*13 | *w*13 → *w*14, *r*15 | 83% |
| *r*15, *w*14, *w*13 | *w*14, *w*13 → *r*15 | 100% |
| *r*16, *r*15, *w*14 | *r*16, *w*14 → *r*15 | 100% |
| *r*16, *r*15, *r*10 | *r*16, *r*15 → *r*10 | 100% |

technology, called the Grey Wolf Optimizer (GWO), inspired from Grey Wolf, and studied its techniques for solving standard and real-life problems. The GWO variants imitate the hunting methods and leadership skills inherited by grey wolves in the wild. In GWO, the population is divided into four groups such as alpha, beta, delta and omega, which are used to mimic the management hierarchy and simulate the leadership hierarchy.

In this section the numerical models of the social chain of command, following, encompassing, and assaulting prey are given. At that point the GWO calculation is laid out.

To numerically display the social order of wolves when planning GWO, people can consider the most suitable arrangement as alpha ($\alpha$). Therefore, the second and third best arrays are named beta ($\beta$) and delta ($\delta$), respectively. The remaining competitor deals are considered Omega ($\gamma$). In the GWO calculation, the chasing (improvement) is guided by $\alpha$, $\beta$ and $\delta$. x wolves follow these three wolves.

**Encircling prey** As referenced above, grey wolves encircle prey during the chase.To numerically demonstrate surrounding conduct, the accompanying conditions are proposed:

$$\mathbf{d} = |\mathbf{c}.\mathbf{x}_{p(t)} - \mathbf{x}(t)| \quad (2)$$

$$\mathbf{x}(t+1) = \mathbf{x}_{p(t)} - \mathbf{a}.\mathbf{d} \quad (3)$$

where t represents the current iteration, **a** and **c** are coefficient vectors, $x_p$ is the position vector of the prey and x is the position vector of the grey wolf. The vectors a and c are calculated as follows:

$$\mathbf{a} = 2l.\mathbf{r}_1 \quad (4)$$

$$\mathbf{c} = 2.\mathbf{r}_2 \quad (5)$$

where components of a are linearly decreased from 2 to 0 over the course of iterations and $r_1$, $r_2$ are random vectors in [0, 1].

**Hunting** In order to simulate hunting behavior mathematically, we assume that alpha ($\alpha$), beta ($\beta$), and delta ($\delta$) have a better understanding of the potential location of the prey.The following mathematical equations are developed in this regard: Equation of hunting Cums

$$\mathbf{d}_\alpha = |\mathbf{c}_1.\mathbf{X}_\alpha - \mathbf{x}|, \mathbf{d}_\beta = |\mathbf{c}_1.\mathbf{X}_\beta - \mathbf{x}|, \mathbf{d}_\delta = |\mathbf{c}_1.\mathbf{X}_\delta - \mathbf{x}| \quad (6)$$

$$\mathbf{x}_1 = \mathbf{X}_\alpha - \mathbf{a}_1.(\mathbf{d}_\alpha), \mathbf{x}_2 = \mathbf{X}_\beta - \mathbf{x}_2.(\mathbf{d}_\beta), \mathbf{x}_3 = \mathbf{X}_\delta - \mathbf{a}.(\mathbf{d}_\delta) \quad (7)$$

$$\mathbf{x}(t+1) = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3}{3} \quad (8)$$

**Searching for prey and attacking prey** A is random value in gap$[-2a, 2a]$.When random value $|A| < 1$, the wolves are forced to attack the prey. Searching for prey is the exploration ability and attacking the prey is the exploitation ability. The arbitrary values of A are utilized to force the search to move away from the prey. When $|A| > 1$, the members of the population are enforced to diverge from the prey.

**Attacking prey (exploitation)** As referenced above, the grey wolves finish the chase by assaulting the prey when it quits moving. To numerically display moving toward the prey we decline the estimation of $\sim a$. Note that the vacillation scope of $\sim A$ is likewise diminished by $\sim a$. In other words $\sim A$ is an arbitrary incentive in the stretch [2a, 2a] where an is diminished from 2 to 0 throughout emphasess. At the point when irregular estimations of $\sim A$ are in [1, 1], the following situation of a pursuit specialist can be in any position between its present position and the situation of the prey. It shows that $|A| < 1$ powers the wolves to assault towards the prey.

**Search for prey (exploration)** Grey wolves mostly search as indicated by the situation of the alpha, beta, and delta. They wander from one another to look for prey and unite to assault prey. To numerically show disparity, we use $\sim A$ with random values greater than 1 or less than -1 to oblige the pursuit specialist to wander from the prey.- This emphasizes exploration and permits the GWO algorithm to look universally. It additionally shows that $|A| > 1$ powers the grey wolves to separate from the prey to ideally locate a fitter prey.

**Whale optimization algorithm(WOA)** The WOA is presented by Mirjalili et al. [12] to take care of the global optimization problem by imitating the conduct of humpback whales. WOA investigations show us two methods through which WOA can be adapted even more . First being the fact that its gradient free mechanism because it has capacity to dodge nearby optima and get global optimal solution. Second, that WOA doesn't require underlying changes in calculations for addressing distinctive improvement issues because it just has two fundamental parameters to be adjusted

- First, it presents all the investigations and explorations for WOA, in which a meta-heuristic hybrid model is used to integrate WOA with different methods to update the presentation of subsequent calculations.
- Second, this work has zeroed out all modification techniques that have been applied to WOA to improve its ability to find the best arrangement.
- Third, we have collected most of the scouting work identified by various applications used in the WOA.

The focus of this research includes several aspects: First, it presents all the investigations and explorations for WOA, in which a meta-heuristic hybrid model is used to integrate WOA with different methods to update the presentation of subsequent calculations. Second, this work has zeroed out all modification techniques that have been applied to WOA to improve its ability to find the best arrangement. Third, we collected most of the scouting work identified by the different applications used in the WOA.

In this section the mathematical model of encircling prey, spiral bubble-net feeding maneuver, and search for prey is first provided. The WOA algorithm is then proposed.

**Encircling prey phase** Humpback whales can identify the location of their prey and surround them. Since the position of the optimal design in the search space is not known a priori, the WOA algorithm assumes that the current best candidate solution is the target prey or is close to the optimal solution. Once the best search agent is defined, other search agents will try to upgrade their ranking to the best search agent. This behavior is represented by the following equations:

$$\mathbf{D} = |\mathbf{C}.\mathbf{X}_{p(t)} - \mathbf{X}(t)| \quad (9)$$

$$\mathbf{X}(t+1) = \mathbf{X}_{p(t)} - \mathbf{A}.\mathbf{D} \quad (10)$$

where t indicates the current iteration, **A** and **D** are coefficient vectors, $X_p$ is the position vector of the prey, and X indicates the position vector of a whale.

The vectors **A** and **D** are calculated as follows:

$$\mathbf{A} = 2\mathbf{a}.\mathbf{r}_1 - \mathbf{a} \tag{11}$$

$$\mathbf{C} = 2.\mathbf{r}_2 \tag{12}$$

where the components of **a** are linearly decreased from 2 to 0 over the course of iterations and **r1**, **r2** are random vectors in [0,1].]

**Bubble-net attacking method (Exploitation Phase)** In order to mathematically model the bubble-net behavior of humpback whales, two approaches are designed as follows:

1. **Shrinking encircling mechanism** This behavior is achieved by reducing the value of **a**. Note that the fluctuation range of **A** is also reduced by **a**. In other words, **A** is a random value on the interval [a, a], where a decreases from 2 to 0 during the iteration. Set a random value for **A** to [1,1] and you can define a new location for the search agent at any position between the original agent location and the current best agent location.
2. **Spiral updating position** This approach first calculates the distance between the whale located at (X,Y) and prey located at (X*,Y*). A spiral equation is then created between the position of whale and prey to mimic the helix-shaped movement of humpback whales as follows:

$$\mathbf{X}(t+1) = \mathbf{D}'e^{bt}cos(2\Pi t) + \mathbf{X}^*(t) \tag{13}$$

where $\mathbf{D}' = |\mathbf{X}^*(t) - X(t)|$ and indicates the distance of the $i_{th}$ whale the prey (best solution obtained so far), b is a constant for defining the shape of the logarithmic spiral, and t is a random number in [− 1,1].

Note that the humpback whale swims around its prey in a tight circle while following a spiral path. To simulate this simultaneous behavior, we assumed a 50% probability of choosing between the shrink wrap mechanism or the spiral model to update the position of the whale during the optimization process. The mathematical model is as follows:

$$\mathbf{X}(t+1) = \begin{cases} \mathbf{X}^*(t) + \mathbf{A}.\mathbf{D}, & \text{if } p<0.5 \\ \mathbf{D}'e^{bt}cos(2\Pi t) + \mathbf{X}^*(t), & \text{if } p \geq 0.5 \end{cases} \tag{14}$$

where $p$ is a random number in [0,1].

In addition to the bubble-net method, the humpback whales search for prey randomly. The mathematical model of the search is as follows:

**Search for prey(Exploration Phase)** The same method based on the change of the vector **A** can be used to search for prey . In fact, humpback whales randomly search based on each other's location. Therefore, we use **A** with a random values greater than 1 or less than -1 to force the search agent to stay away from the reference whale. Unlike the exploitation phase, we update the position of the search agent in the exploration phase based on the randomly selected search agent instead of the best search agent. This mechanism and $|A| > 1$ emphasize browsing and allow the WOA algorithm to perform a global search. The mathematical model is as follows:

$$\mathbf{D} = |\mathbf{C}.\mathbf{X}_{rand} - \mathbf{X}(t)| \tag{15}$$

$$\mathbf{X}(t+1) = \mathbf{X}_{rand} - \mathbf{AD} \tag{16}$$

where $\mathbf{X}_{rand}$ is a random position vector (a random whale).

**Algorithm 2:** mhGW-WOA Clustering

```
   Input : Load Database Logs
   Result: Cluster centroids
 1 Begin:
 2   Initialize number of search agents
 3   Initialize each search agent to contain K
       randomly cluster centers
 4   while t < no. of total iterations do
 5     for search agent do
 6       Calculate fitness value for the agent
 7       Update best search agent
 8     end
 9     for search agent do
10       Update a, A, r, c, l, p, b
11     end
12     if p<0.5 then
13       if |a| <1 then
14         D⃗ = |C⃗.X⃗*(t) − X⃗(t)|
15         X⃗(t+1) = X⃗*(t) − A⃗.D⃗
16       else
17         D⃗ = |C⃗.X⃗_rand − X⃗|
18         X⃗(t+1) = X⃗_rand − A⃗.D⃗
19       end
20     else
21       if |a| <1 then
22         D⃗′ = |X⃗*(t) − X⃗(t)|
23         X⃗(t+1) = D⃗′.e^bl. cos(2πl) + X⃗*(t)
24       else
25         Get best 3 positions
26         x[i] = pscores[0][1] + pscores[1][1] +
             pscores[2][1]
27         x[i]/ = 3
28       end
29     end
30   end
31 End
```

Steps 2 and 3 consist of the following tasks: initialising search agent variables. The next five steps entail searching for the most qualified potential agent available.

In step 9, the variables are updated to reflect the new values that have been determined. Using the Whale Optimization Algorithm, we will next update our agents at stages 10 through 20 of the process.

Then, in steps 21 to 24, we use the Modified Grey Wolf Optimization Algorithm to update our agents' information. When compared to previous techniques, this step broadens the scope of the algorithms' learning capabilities.

The algorithm produces the best cluster centres, which will be utilised as the basis for role profiles in the following section. Our clustering method is optimised via hybrid of the Whale Optimization Algorithm and the Grey Wolf Optimization Algorithm, as shown in Fig. 2.

## 3.3 Detection and classification phase

The learning phase of the algorithm provides the read rules, write rules and the role profiles. Now, these rules and role profile clusters are used to detect any anomalous transaction via the detection phase architecture.

The architecture of the detection phase is shown in Fig. 3. In this detection phase, the current user transaction is first passed through the transaction pre-processor and query parser. Then the anomalies are detected in the current user transactions for each transaction by checking them against the user role profiles through role authenticator and against the sequential rules generated in the learning phase segment, via the rule validator. If a query is found to be malicious, an alarm is triggered. Otherwise the transaction is committed to the database.

The detection phase architecture has 2 levels namely Role Authenticator and Rule Validator, which are described below.

### 3.3.1 Role authenticator

The role profile clusters generated by the clustering algorithm in the learning phase are matched against the incoming transaction in the database. The role authenticator individually checks all the queries to determine whether they are justified by the role of the user or not, thereby making the intrusion detection model immune against the insider threats as well. If the role profile of an incoming transaction is not matched to the existing profile clusters, the transaction is declared as malicious and aborted, instantaneously. However, if the role profile match is found, the transaction sequence is further checked by the rule validator for classification.

### 3.3.2 Rule validator

The rule validator identifies whether queries in a single configuration file are executed in an order similar to the order of previously validated queries. The incoming transaction query sequence is compared with read and write rules that are derived from data dependency mining using the CMSPADE algorithm in the learning phase. The input sequence list generated at run time is compared to the data-dependent rules generated for each sensitive element present in the list. Therefore, compliance with each rule is calculated. Now, we need to quantify the rule-based similarity of the two sequence datasets to draw conclusions about the credibility of the incoming transactions. This is done by calculating the similarity, as described in the next section.

### 3.3.3 Calculating similarity

A 100 point similarity system has been introduced to measure the extent of adherence between transaction and a set of rules mined using the rule mining algorithm. We have four grades of similarity called resemblance (R). Each grade has been allocated points. We try to contain the values of overlap measurement between 0.1 and 0.9 for minimum and maximum adherence respectively.

We define:

$$\mathbf{l} = 0.1 \; and \; \mathbf{h} = 0.9 \tag{17}$$

I ***Grade D resemblance:*** **(10 Points assigned out of 100)**

Grade D resemblance denoted by R4 is the measure of the number of mined rules in our database that have the same element accused in the transaction as in the rules. Let, a : the number of rules that follow this condition. b : the number of rules that do not follow this condition.

$$R_4 = \left[1 + \frac{(a-1)\cdot 1b}{(a+b)} * (h-1)\right] * 10 \tag{18}$$

*(Higher weight is assigned to rules that do not follow the required conditions)*

II ***Grade C resemblance:*** **(20 Points assigned out of 100)**

This stage considers a subset of rules that belong to set "a" in stage i.e the rules that have common elements as incoming transactions. Let, x : denotes the number of rules that follow the same order as in transactions. *Eg* : TR : R(a)R(b)W(c) *Rule* : R(d)R(a)R(b)W(a)W(c) y : denotes the number of rules that do not follow that condition.

$$R_3 = \left[1 + \frac{(x-1)\cdot 2y}{(x+y)} * (h-1)\right] * 20 \tag{19}$$

III ***Grade B resemblance:*** **(30 Points assigned out of 100)**

This stage considers subsets of stage 2.The set of rules is used i.e those rules that follow the same order as transactions. Let, p : the number of rules that

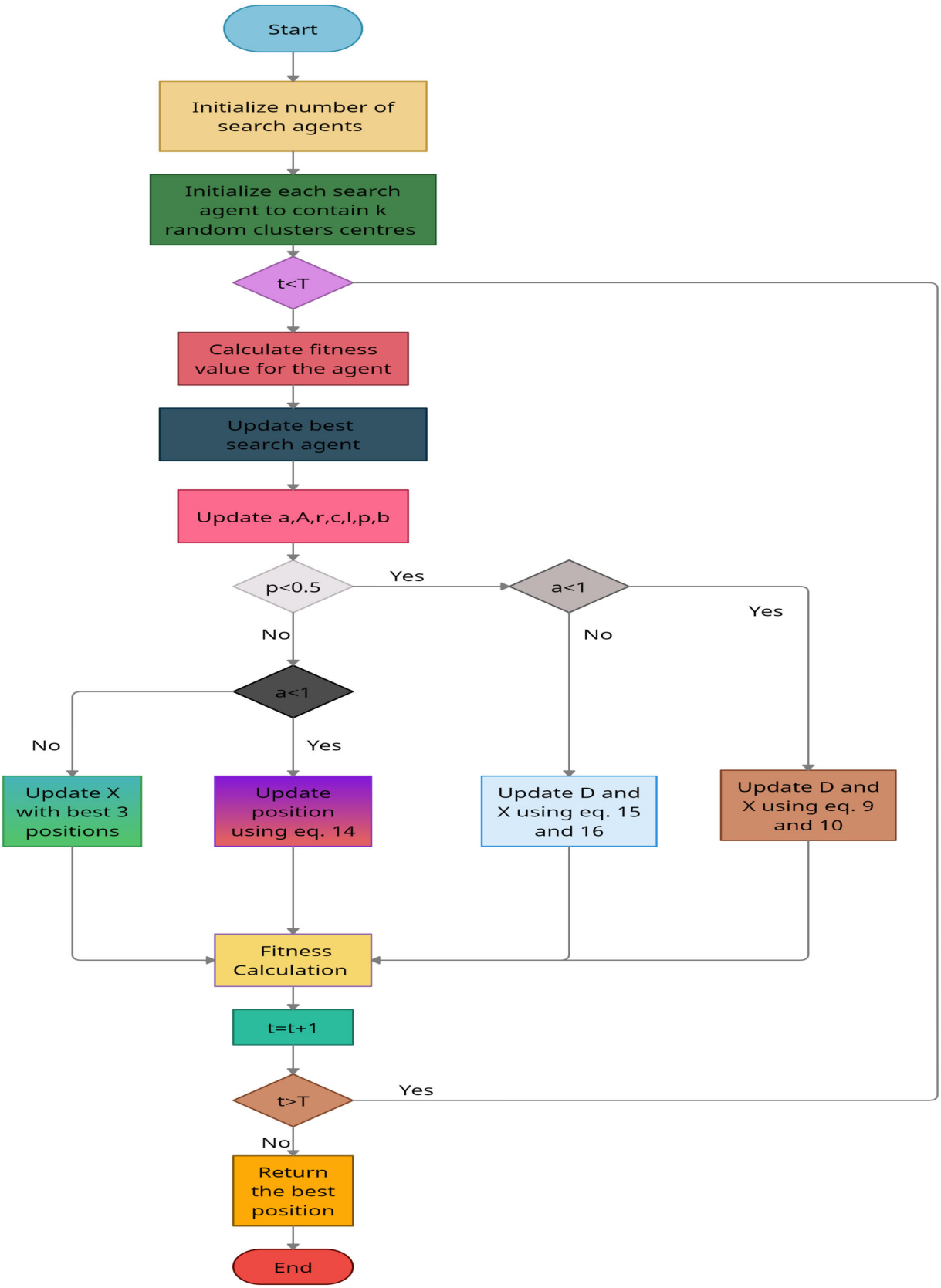


**Fig. 2** Flowchart of the mhGW-WOA clustering

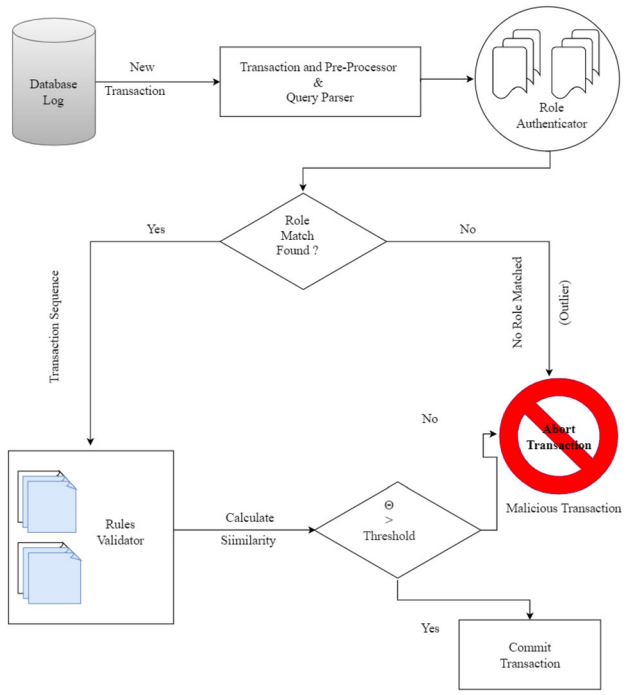


**Fig. 3** Detection Phase architecture

contain transactions as a subset i.e all common elements are together as in the transaction. q : the number of elements that do not follow this rule.

$$R_2 = \left[1 + \frac{(p-1)\cdot 25q}{(p+q)} * (h-1)\right] * 30 \quad (20)$$

IV ***Intra-Transaction rules:*** These rules consider a group of transactions that are executed together in a sequential manner. The algorithm used will be similar to the mentioned above. The rule transaction will be considered sequentially in the batches of k $(2 \leq K \leq 5)$. Let, s : number of rules that adhere to the transaction set in exact order. t : number of rules that do not adhere to the transaction set.

(a) ***Grade $A_2$ resemblance***

This follows the above methods given in the above sections. The batch of size 2 is assigned a high weight and the order should look like: **k=2 12 Points assigned out of 100**

$$R_{12} = \left[1 + \frac{(s-1)\cdot 2t}{(s+t)} * (h-1)\right] * 12 \quad (21)$$

(b) ***Grade $A_3$ resemblance***

The batch size of 3 will follow the same order as the above subset. It is assigned a lower weight than the $A_2$ resemblance. But it will have the same multiple of $A_2$ resemblance as the points assigned to both these resemblances are the same. **k=3 12 Points assigned out of 100**

$$R_{13} = \left[1 + \frac{(s-1)\cdot 1t}{(s+t)} * (h-1)\right] * 12 \quad (22)$$

(c) ***Grade $A_4$ resemblance***

$A_4$ resemblance represents a batch with size 4. The order for the $A_4$ resemblance will be exactly half of that of $A_3$ resemblance as the points given to the $A_4$ resemblance is half of it and also the weight assigned to it is the same as that of the previous subset. **k=4 6 Points assigned out of 100**

$$R_{14} = \left[1 + \frac{(s-1)\cdot 1t}{(s+t)} * (h-1)\right] * 6 \quad (23)$$

(d) ***Grade $A_5$ resemblance***

The $A_5$ resemblance is for the final batch of size 5. As the $A_5$ resemblance contains the same points as that of the previous subset, i.e. $A_4$ resemblance so it will have the same multiple as that of $A_4$. But the two equations should differ as the weights assigned to $A_5$ resemblance is the lowest of all the Grade A resemblances. **k=5 6 Points assigned out of 100**

$$R_{15} = \left[1 + \frac{(s-t)}{(s+t)} * (h-1)\right] * 6 \quad (24)$$

**Definition 16** (*Congruity Index*) A Congruity Index $(\Theta)$ is a similarity threshold defined as the sum of all the grades of similarity (resemblances).

$$\Theta = \sum_{i=1}^{n} R_i$$

where $n$ is the number of grades.

In our case we have four grades namely A,B,C,D with four subgrades for A $(A_1, A_2, A_3, A_4)$ Therefore,

$$\Theta = R_{12} + R_{13} + R_{14} + R_{15} + R_2 + R_3 + R_4 \quad (25)$$

### 3.3.4 Detection algorithm

In our proposed algorithm 3 we take in the generated rule set, number of clusters, similarity threshold $(\delta)$ and the transactions itself. Our main aim is to classify the transaction as either malicious or non-malicious. The algorithm starts with converting the transaction into sequence using the sequence generator. Then in steps 4- 9 we calculate the role similarity of each cluster and store the highest role similarity cluster along with its similarity. We compare that

similarity with that of the threshold, if it's less than the threshold we classify the transaction as malicious and raise an alarm. However if the similarity is greater than the threshold, we pass in the respective rule into the rule authenticator module. This novel module simply calculates the different resemblances (B, C, D, $A_1$, $A_2$, $A_3$, $A_4$) from step 13-20. We define congruity index ($\Theta$) which would be used to classify the transaction. In steps 22-25 we compare the value of $\Theta$ with the $\delta$ . If $\Theta$ is less than $\delta$ then the transaction is declared malicious and if $\Theta$ is greater than $\delta$ then the transaction is declared non-malicious. Also following this in steps 27–30 if the transaction was malicious the database executes a rollback along with raising an alarm and if the transaction was non-malicious the database commits all the data.

**Algorithm 3:** Detection Phase

**Data :** R: Rule Set
K: Number of Clusters
$\delta$: Similarity Threshold
T: Transaction
**Result:** C:Class (malicious or non-malicious)

1 **During Transaction:**
2 **for** *each of the Query q in TRN* **do**
3 Seq ← SequenceGenerator(q)
4 Max_sim ← $\phi$
5 for all $C_k \forall$ Cluster_Centres
6 Temp ← cal_role_sim(q,$C_k$)
7 **if** *Temp* > *Max_sim* **then**
8 Max_sim ← Temp
9 Max_cluster ← $C_k$
10 **End If**
11 **end**
12 if($Max\_sim < \delta$)
13 **Raise Alarm**
14 **for** *each* ***rule*** *in R* **do**
15 $R_4$ = GradeD_resemblance(rule, Seq)
16 $R_2$ = GradeC_resemblance(rule, Seq)
17 $R_3$ = GradeB_resemblance(rule, Seq)
18 $R_{12}$ = Grade$A_2$_resemblance(rule, Seq)
19 $R_{13}$ = Grade$A_3$_resemblance(rule, Seq)
20 $R_{14}$ = Grade$A_4$_resemblance(rule, Seq)
21 $R_{15}$ = Grade$A_5$_resemblance(rule, Seq)
22 $\Theta = R_4 + R_3 + R_2 + R_{12} + R_{13} + R_{14} + R_{15}$
23 **if** $\Theta < \delta$ **then**
24 C← malicious
25 **else**
26 C← non-malicious
27 **end**
28 **if** *(C== malicious)* **then**
29 **Rollback – Raise Alarm**
30 **else**
31 commit
32 **end**
33 **end**
34 **End**

## 4 Experimentation and results

In order to analyze the performance of the proposed methodology (FPGWWO), multiple assessments were performed on a synthetically generated banking databases adhering to the TPC-C benchmark [57]. The database consisted of two types of logs - one with the normal user transactions of various user roles and other which comprised a mixture of normal and malicious user transactions. These two types of logs in the data-base were used to evaluate the performance of our system. In total, 5000 transactions were generated. Precision, recall, and F1-score were the performance measures utilised in the assessment. Precision is defined as the ratio of properly identified malicious transactions, referred to as True Positives (TP), to the total number of transactions classified as malicious from database logs, which includes both False Positives (FP) and True Positives (TP). The ratio of correctly identified malicious transactions, known as True Positives (TP), to the total number of malicious transactions in the database logs, a combination of False Negatives (FN) and True Positives (TP), is defined as recall. High precision and low recall, as well as low precision and high recall, indicate a poor performing model in the situation of an unbalanced dataset. For performance evaluation, the F1-score is employed, which considers both precision and recall. The harmonic mean of precision and recall is defined as the F1-score.

### 4.1 Generation of synthetic transactions

FPGWWO took into account the fact that no datasets that conformed to our methodology existed, prompting us to employ synthetically created datasets effectively. This dataset comprises two separate modules for the two sorts of transactions that we worked on: malicious transactions generation module and regular transactions generation module, both of which are described and worked on below.

### 4.2 Generation of malicious transactions

We created two sets of fraudulent transactions to test our technique against a wide range of threats. The attacker may be ignorant of regular database requirements, thus the first batch of transactions was produced at random. This collection is created by randomly changing characteristics of genuine queries that make up normal behaviour, as well as legitimate transactions of each. The second set of transactions were created in such a way that they do not correspond to typical user behaviour and are signs of people attempting transactions that are outside of their scope and authorization.

### 4.3 Generation of normal transactions

Normal transactions are ones that, at the moment of execution, fulfil data dependencies and user-access patterns. We define a predefined set of SQL queries that comply to the TPC-C benchmark and the schema used to create regular transactions. Individual characteristics are allocated to each user that are within the scope of their access and permissions, and transactions are created based on those attributes, meeting the user-access patterns.

### 4.4 Evaluation of our approach

The Figs. 4, 5, 6 above depict the variation in precision, recall and F1 Score of the system for threshold value of 50. With change in the number of transactions in the database. It can be inferred from the graph that the value of Precision first increases from 0.785 to 0.855. With an increase in the number of transactions, the value of Recall rises from 0.912 to 0.950. The number of rules created for each transaction grows as the number of transactions grows. As a result, rules will be more matched to the database, and role clustering will be more refined. As a result, the number of false negatives decreases while the number of true positives rises, resulting in an increase in the recall value. Because the F1-score is a harmonic mean of Precision and Recall, it reflects a good balance of precision and recall by continuing to rise between the two values, ranging from 0.845 to 0.882 (Table 5).

Using a threshold value of 60, the Figs. 7, 8, 9 illustrate the variance in accuracy, recall, and F1 Score of the system as a function of the number of transactions in the database. As shown in the graph, the value of Precision begins to rise from 0.772 to 0.818 at the beginning of the study period. Increase in the number of transactions result in an increase in the value of Recall, which rises from 0.921 to 0.938. Increasing the number of transactions leads to an increase in the number of rules that are created for each transaction. As a result, the rules are more matched to the database, and

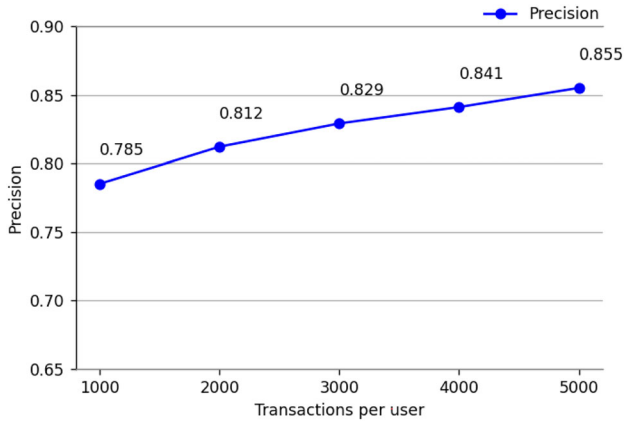


**Fig. 4** Variation of precision with number of transactions per user at Threshold 50

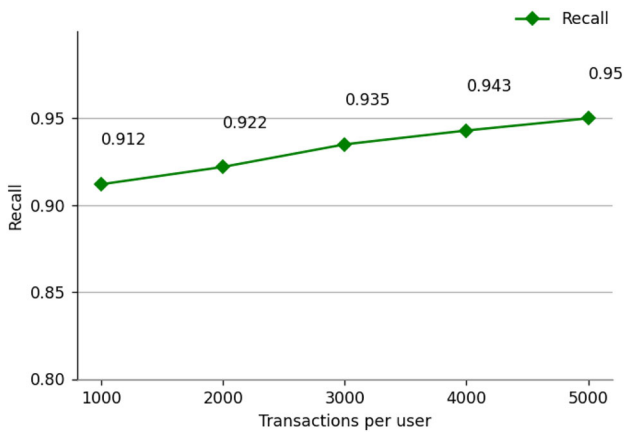


**Fig. 5** Variation of recall with number of transactions per user at Threshold 50

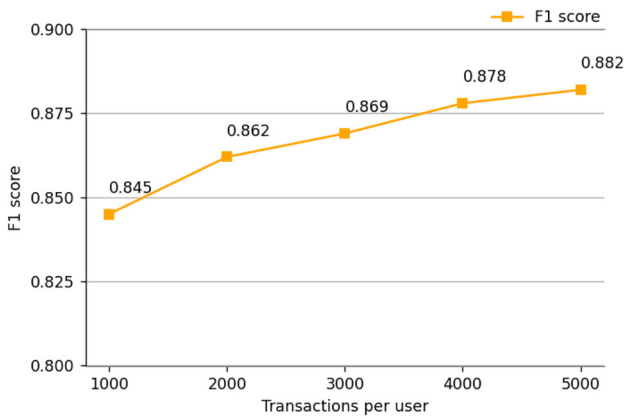


**Fig. 6** Variation of F1 score with number of transactions at Threshold 50

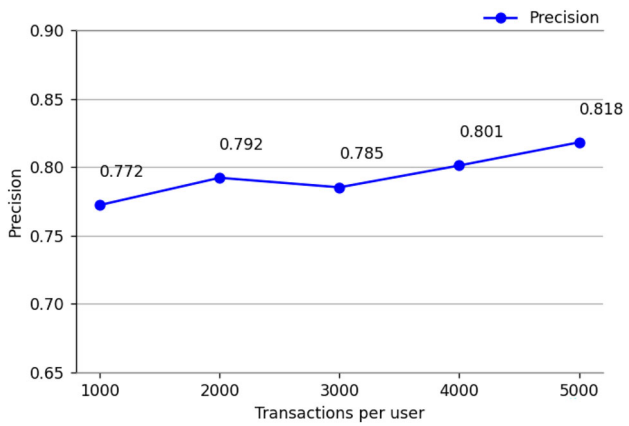


**Fig. 7** Variation of Precision with number of transactions per user at Threshold 60

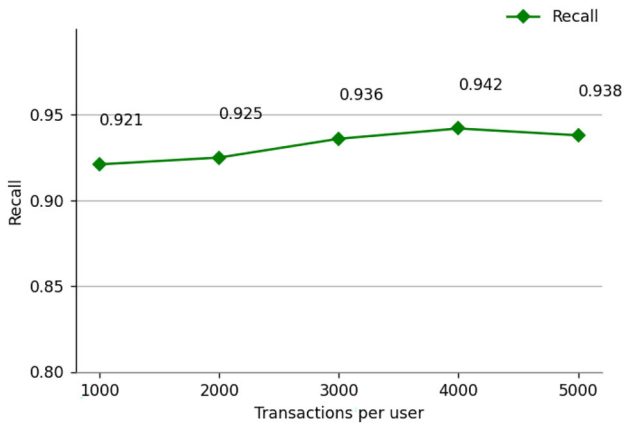


**Fig. 8** Variation of Recall with number of transactions per user at Threshold 60

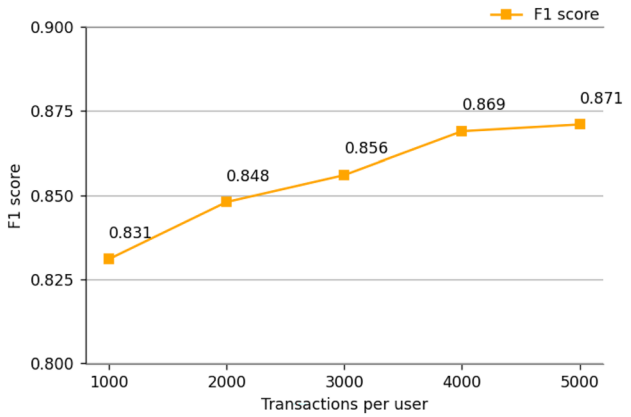


**Fig. 9** Variation of F1 score with number of transactions at Threshold 60

the role clustering is more refined as a result of these improvements. Thus, there is a drop in the number of false negatives and an increase in the number of true positives, which leads to an increase in the recall value due to the decrease in false negatives. Here also, the value of F1 Score shows a slight increase from 0.831 to 0.871 (Table 6).

We also assess how well our role clustering algorithm performs. Figure 10 shows the variation of the loss of the clustering algorithm. As observed from the graph, the loss steadily decreases initially and then reaches a stable value and sustains it. Loss value becomes less than half of its initial value which is an impressive improvement and shows the adaptability of the algorithm on the dataset. This behaviour validates that the algorithm is performing good for the study.

Comparative performance of our technique with other eminent authors of this domain has been tabulated below. We have incorporated an approach that examines the intra-transactional queries to deal with the insider threats as well

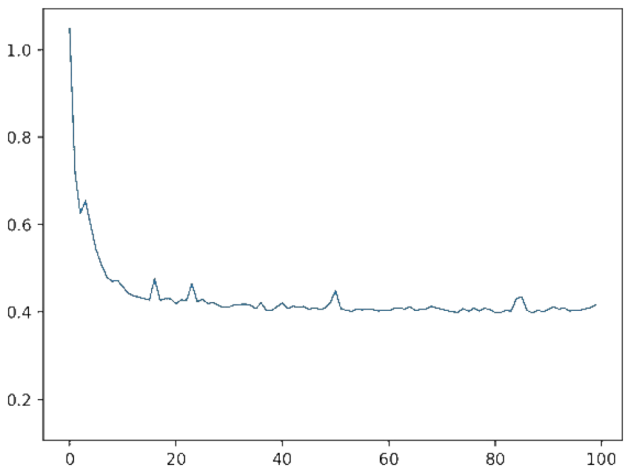

**Fig. 10** Loss of mhGW-WOA clustering

**Table 5** Precision, recall, F1-score at Threshold 50

| Parameter | Threshold = 50 | | | | |
|---|---|---|---|---|---|
| | Number of transactions | | | | |
| | 1000 | 2000 | 3000 | 4000 | 5000 |
| Precision | 0.785 | 0.812 | 0.829 | 0.841 | 0.855 |
| Recall | 0.912 | 0.922 | 0.935 | 0.943 | 0.95 |
| F1 Score | 0.845 | 0.862 | 0.869 | 0.878 | 0.882 |

**Table 6** Precision, recall, F1-score at Threshold 60

| Parameter | Threshold = 60 | | | | |
|---|---|---|---|---|---|
| | Number of transactions | | | | |
| | 1000 | 2000 | 3000 | 4000 | 5000 |
| Precision | 0.772 | 0.792 | 0.785 | 0.801 | 0.818 |
| Recall | 0.921 | 0.925 | 0.936 | 0.942 | 0.938 |
| F1 Score | 0.831 | 0.848 | 0.856 | 0.869 | 0.871 |

as external threats. We have used algorithms like CM-SPADE and Hybridization of Whale Optimization Algorithm with Modified Grey Wolf clustering Algorithm and a novel approach to similarity calculation. These algorithms have enhanced the performance of our approach as compared to the other state-of-the-art techniques.

Table 7 contrasts the various techniques based on the algorithm used, including but not limited to its ability to avoid intrusion and higher performance on metrics such as precision and recall. On comparing the metrics across different support values for each technique, the maximum values were considered. We observed better performance for our algorithm compared to the alternatives. The critical factor behind the improved performance is the sensitivity

**Table 7** Performance Comparison of FPGWWO with state-of-the-art techniques

| Author Name | Proposed technique | Use of RBAC | Intrusion detection capabilities | Approach for query evaluation | Detection level | Detection of threats | Detection paradigm | Performance |
|---|---|---|---|---|---|---|---|---|
| Hu et al. [5] | Integrated dependency with sequence alignment analysis | No | Partial | Syntax Centric | Transaction level | External and user threats only | Anomaly | Recall = 0.90 Precision = 0.64 |
| Srivastava et al. [17] | Integrated dependency with sequence alignment analysis | No | Partial | Syntax Centric | Transaction level | External and user threats only | Anomaly | Recall = 0.77 Precision = 0. 80 |
| Kamra et al. [42] | Dependency And Relation Analysis | Yes | Yes | Syntax Centric | Transaction level | Insider using RBAC, External using clustering | Anomaly | Recall = 0.63 Precision = 0.77 |
| Hashemi et al. [24] | Temporal Mining | No | Yes | Syntax Centric | Transaction Level | Both external and insider threats | Anomaly analysis of the time series | Recall = 0.90 Precision = 0.75 |
| Kundu et al. [26] | Sequence alignment analysis | No | Partial | Syntax Centric | Transaction level | External and user threats | Anomaly | Precision = 0.945 |
| Panigrahi et al. [43] | User Behavior Mining | Yes | Partial | Syntax Centric | Transaction level | Both External and Insider threats | Both Anomaly detection and misuse detection | Recall = 0.93 Precision = 0.91 |
| Doroudian et al. [58] | Mining dependencies | No | Yes | Syntax Centric | Transaction level | External and user threats | Anomaly and specification based | Recall = 0.89 Precision = 0.91 |
| Ronao and Cho [48] | Weighted Random Forest | Yes | Yes | Syntax Centric | Root level | External and user threats | Anomaly | Recall = 0.95 Precision = 0.89 |
| Sallam et al. [46] | Anomaly Detection using Bayesian Classifier | Yes | Yes | Data and Syntax Centric | RDBMS level | Insider threats | Anomaly | Recall = 0.97 Precision = 0.90 |
| Kim and Cho [49] | CNN - LSTM Neural Network | Yes | Yes | Data Centric | RDBMS level | User threats and Insider threats | Anomaly | Recall = 0.887 Accuracy of 0.933 |
| Seok-Jun et al. [59] | Convolutional Neural-based Learning Classifier | Yes | Yes | Data-Centric | RDBMS level | Insider Threats | Anomaly | Recall = 0.93 Precision = 0.92 |
| **FPGWWO** (our approach) | Frequent Sequential Pattern Mining and metaheuristic clustering | Yes | Yes | Data and Syntax centric | Transaction level | External and insider threats | Anomaly | Recall = 0.938 Precision = 0.91 |

of the algorithm to change in user patterns since we consider the relative adherence to data dependencies. This behaviour is expressed by real-world transactions where they never fully comply with data dependencies.

Since our algorithm combines the benefits of anomaly-based and statistical detection methods, we are able to reduce the number of false positives and false negatives errors.

## 5 Conclusion and future scope

In this study, we have proposed a Database Intrusion Detection System (DIDS) to prevent unauthorised access from changing critical user details. Our method (FPGWWO) can identify threats from both within and outside the system by incorporating a frequent sequential pattern mining algorithm and a hybrid metaheuristic clustering using modified Grey Wolf optimization and Whale Optimization Algorithm (mhGW-WOA).

CM-SPADE is used to mine frequent sequential patterns from database logs to construct read and write rules, which are based on legitimate access patterns. Parallely, the hybrid swarm clustering algorithm (mhGW-WOA) clusters out users based on their role profiles to ensure that each user's role is matched in the current role profiles and determine congruity index.

In terms of computational complexity, storage efficiency, and faster convergence rate, the hybrid of grey wolf and whale optimizers appears to be advantageous, as well as provides increased accuracy in clustering role profiles for user transactions. As a result, when compared to existing state-of-the-art methodologies in the literature, our proposed methodology has a higher overall efficiency. Our future research will focus on improvising ways for integrating user behaviour and other elements of transaction pattern mining utilising more efficient mining algorithms. It would also take into account the limits of the current technique in terms of adding new features and calculating adherence to improve the model's performance.

## References


1. Fernández-García, A.J., Iribarne, L., Corral, A., Criado, J., Wang, J.Z.: A flexible data acquisition system for storing the interactions on mashup user interfaces. Comput. Standards Interfaces **59**, 10–34 (2018)
2. Bertino, E., Sandhu, R.: Database security-concepts, approaches, and challenges. IEEE Trans. Depend. Secure Comput. **2**(1), 2–19 (2005)
3. Cappelli, D.M., Moore, A.P., Trzeciak, R.F.: The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes (Theft, Sabotage, Fraud). Addison-Wesley (2012)
4. Heady, R., Luger, G., Maccabe, A., Servilla, M.: The architecture of a network level intrusion detection system. Tech. rep., Los Alamos National Lab., NM (United States); New Mexico Univ., Albuquerque (1990)
5. Hu, Y., Panda, B.: A data mining approach for database intrusion detection. In: Proceedings of the 2004 ACM symposium on Applied computing, pp. 711–716 (2004)
6. Debar, H., Dacier, M., Wespi, A.: Towards a taxonomy of intrusion-detection systems. Comput. Netw. **31**(8), 805–822 (1999)
7. Preuveneers, D., Rimmer, V., Tsingenopoulos, I., Spooren, J., Joosen, W., Ilie-Zudor, E.: Chained anomaly detection models for federated learning: An intrusion detection case study. Appl. Sci. **8**(12), 2663 (2018). https://doi.org/10.3390/app8122663
8. Lee, W., Stolfo, S.: Data mining approaches for intrusion detection (1998)
9. Barbará, D., Couto, J., Jajodia, S., Wu, N.: Adam: a testbed for exploring the use of data mining in intrusion detection. ACM Sigmod Record **30**(4), 15–24 (2001)
10. Kanoun, W., Cuppens-Boulahia, N., Cuppens, F., Autrel, F.: Advanced reaction using risk assessment in intrusion detection systems. In: International Workshop on Critical Information Infrastructures Security, pp. 58–70. Springer (2007)
11. Sandhu, R., Ferraiolo, D., Kuhn, R., et al.: The nist model for role-based access control: towards a unified standard. In: ACM workshop on Role-based access control, vol. 10 (2000)
12. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
13. Hoglund, A.J., Hatonen, K., Sorvari, A.S.: A computer host-based user anomaly detection system using the self-organizing map. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, vol. 5, pp. 411–416. IEEE (2000)
14. Lunt, T.F., Tamaru, A., Gillham, F.: A real-time intrusion-detection expert system (IDES). SRI International, Computer Science Laboratory (1992)
15. Talpade, R., Kim, G., Khurana, S.: Nomad: Traffic-based network monitoring framework for anomaly detection. In: Proceedings IEEE International Symposium on Computers and Communications (Cat. No. PR00250), pp. 442–451. IEEE (1999)
16. Hu, Y., Panda, B.: Identification of malicious transactions in database systems. In: Seventh International Database Engineering and Applications Symposium, 2003. Proceedings., pp. 329–335. IEEE (2003)
17. Srivastava, A., Sural, S., Majumdar, A.K.: Database intrusion detection using weighted sequence mining. J. Comput. **1**(4), 8–17 (2006)
18. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. **2**, 222–232 (1987)
19. Corney, M., Mohay, G., Clark, A.: Detection of anomalies from user profiles generated from system logs. In: Proceedings of the Ninth Australasian Information Security Conference, pp. 23–31. Australian Computer Society (2011)
20. Cárdenas, A.A., Amin, S., Lin, Z.S., Huang, Y.L., Huang, C.Y., Sastry, S.: Attacks against process control systems: risk assessment, detection, and response. In: Proceedings of the 6th ACM symposium on information, computer and communications security, pp. 355–366 (2011)
21. Liao, H.J., Lin, C.H.R., Lin, Y.C., Tung, K.Y.: Intrusion detection system: a comprehensive review. J. Netw. Comput. Appl. **36**(1), 16–24 (2013)
22. Hastie, T., Tibshirani, R., Friedman, J.: Unsupervised learning. In: The elements of statistical learning, pp. 485–585. Springer (2009)
23. Chen, M.S., Han, J., Yu, P.S.: Data mining: an overview from a database perspective. IEEE Trans. Knowledge Data Eng. **8**(6), 866–883 (1996)
24. Hashemi, S., Yang, Y., Zabihzadeh, D., Kangavari, M.: Detecting intrusion transactions in databases using data item dependencies and anomaly analysis. Expert Syst. **25**(5), 460–473 (2008)
25. Rahman, M.M., Ahmed, C.F., Leung, C.K., Pazdor, A.G.: Frequent sequence mining with weight constraints in uncertain databases. In: Proceedings of the 12th international conference on ubiquitous information management and communication, pp. 1–8 (2018)

26. Kundu, A., Sural, S., Majumdar, A.K.: Database intrusion detection using sequence alignment. Int. J. Inform. Security **9**(3), 179–191 (2010)
27. Subudhi, S., Panigrahi, S.: Application of optics and ensemble learning for database intrusion detection. J. King Saud University-Comput. Inform. Sci. (2019)
28. Sallam, A., Bertino, E.: Result-based detection of insider threats to relational databases. In: Proceedings of the ninth ACM conference on data and application security and privacy, pp. 133–143 (2019)
29. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the eleventh international conference on data engineering, pp. 3–14. IEEE (1995)
30. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: International conference on extending database technology, pp. 1–17. Springer (1996)
31. Zaki, M.J.: Spade: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1), 31–60 (2001)
32. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining sequential patterns by pattern-growth: the prefixspan approach. IEEE Trans. Knowledge Data Eng. **16**(11), 1424–1440 (2004)
33. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 429–435 (2002)
34. Gomariz, A., Campos, M., Marin, R., Goethals, B.: Clasp: An efficient algorithm for mining frequent closed sequences. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 50–61. Springer (2013)
35. Fournier-Viger, P., Gomariz, A., Campos, M., Thomas, R.: Fast vertical mining of sequential patterns using co-occurrence information. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 40–52. Springer (2014)
36. Lan, G.C., Hong, T.P., Lee, H.Y.: An efficient approach for finding weighted sequential patterns from sequence databases. Appl. Intell. **41**(2), 439–452 (2014)
37. Rahman, M.M., Ahmed, C.F., Leung, C.K.S.: Mining weighted frequent sequences in uncertain databases. Inform. Sci. **479**, 76–100 (2019)
38. Chung, C.Y., Gertz, M., Levitt, K.: Demids: A misuse detection system for database systems. In: Working Conference on Integrity and Internal Control in Information Systems, pp. 159–178. Springer (1999)
39. Spalka, A., Lehnhardt, J.: A comprehensive approach to anomaly detection in relational databases. In: IFIP Annual Conference on Data and Applications Security and Privacy, pp. 207–221. Springer (2005)
40. Alzubi, J.A., Jain, R., Kathuria, A., Khandelwal, A., Saxena, A., Singh, A.: Paraphrase identification using collaborative adversarial networks. J. Intell. Fuzzy Syst. **39**(1), 1021–1032 (2020). https://doi.org/10.3233/JIFS-191933
41. Alzubi, J.A., Jain, R., Nagrath, P., Satapathy, S., Taneja, S., Gupta, P.: Deep image captioning using an ensemble of cnn and lstm based deep neural networks. J. Intell. Fuzzy Syst. **40**(4), 5761–5769 (2021). https://doi.org/10.3233/JIFS-189415
42. Kamra, A., Terzi, E., Bertino, E.: Detecting anomalous access patterns in relational databases. VLDB J. **17**(5), 1063–1077 (2008)
43. Panigrahi, S., Sural, S., Majumdar, A.K.: Two-stage database intrusion detection by combining multiple evidence and belief update. Inform. Syst. Front. **15**(1), 35–53 (2013)
44. Hussain, S.R., Sallam, A.M., Bertino, E.: Detanom: Detecting anomalous database transactions by insiders. In: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy, pp. 25–35 (2015)
45. Sallam, A., Bertino, E., Hussain, S.R., Landers, D., Lefler, R.M., Steiner, D.: Dbsafe-an anomaly detection system to protect databases from exfiltration attempts. IEEE Syst. J. **11**(2), 483–493 (2015)
46. Sallam, A., Fadolalkarim, D., Bertino, E., Xiao, Q.: Data and syntax centric anomaly detection for relational databases. Wiley interdisciplinary reviews: data mining and knowledge discovery **6**(6), 231–239 (2016)
47. Sallam, A., Bertino, E.: Detection of temporal insider threats to relational databases. In: 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), pp. 406–415. IEEE (2017)
48. Ronao, C.A., Cho, S.B.: Anomalous query access detection in rbac-administered databases with random forest and pca. Inform. Sci. **369**, 238–250 (2016)
49. Kim, T.Y., Cho, S.B.: Cnn-lstm neural networks for anomalous database intrusion detection in rbac-administered model. In: International Conference on Neural Information Processing, pp. 131–139. Springer (2019)
50. Mahalingam, T., Subramoniam, M.: A hybrid gray wolf and genetic whale optimization algorithm for efficient moving object analysis. Multim Tools Appl. **78**(18), 26633–26659 (2019)
51. Rathore, R.S., Sangwan, S., Prakash, S., Adhikari, K., Kharel, R., Cao, Y.: Hybrid wgwo: whale grey wolf optimization-based novel energy-efficient clustering for eh-wsns. EURASIP J. Wireless Commun. Netw. **2020**(1), 1–28 (2020)
52. Movassagh, A.A., Alzubi, J.A., Gheisari, M., Rahimi, M., Mohan, S., Abbasi, A.A., Nabipour, N.: Artificial neural networks training algorithm integrating invasive weed optimization with differential evolutionary model. J. Ambient Intell. Human. Comput. (2021). https://doi.org/10.1007/s12652-020-02623-6
53. Rahnema, N., Gharehchopogh, F.S.: An improved artificial bee colony algorithm based on whale optimization algorithm for data clustering. Multim. Tools Appl. **79**(43), 32169–32194 (2020)
54. Aljarah, I., Mafarja, M., Heidari, A.A., Faris, H., Mirjalili, S.: Clustering analysis using a novel locality-informed grey wolf-inspired clustering approach. Knowledge Inform. Syst. **62**(2), 507–539 (2020)
55. Ghany, K.K.A., AbdelAziz, A.M., Soliman, T.H.A., Sewisy, A.A.E.M.: A hybrid modified step whale optimization algorithm with tabu search for data clustering. Journal of King Saud University-Computer and Information Sciences (2020)
56. Viet, K., Panda, B., Hu, Y.: Detecting collaborative insider attacks in information systems. In: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 502–507. IEEE (2012)
57. Benchmark, T.C.: (2020). http://www.tpc.org/tpcc/default5.asp
58. Doroudian, M., Shahriari, H.R.: A hybrid approach for database intrusion detection at transaction and inter-transaction levels. In: 2014 6th Conference on Information and Knowledge Technology (IKT), pp. 1–6. IEEE (2014)
59. Bu, S.J., Cho, S.B.: A convolutional neural-based learning classifier system for detecting database intrusion via insider attack. Inform. Sci. **512**, 123–136 (2020)

**Rajni Jindal** is currently Professor in Department of Computer Science & Engineering at Delhi Technological University (erstwhile Delhi College of Engineering). She is working here as faculty for more than 25 years. She completed her PhD (Computer Engineering) from Faculty of Technology, Delhi University in the area of Data Mining. She received her M.E. (Computer Technology & Applications) degree from - Delhi college of Engineering and her Master of Computer Applications (MCA) degree from Jawaharlal Nehru University (JNU), Delhi. Prior to joining teaching she has also worked with Tata Consultancy Services (TCS). She also worked as Professor, Head (IT) and Dean (Research & Collaboration) at IGDTUW for almost three years. She possesses a work experience of more than 25 years in research and academics. Her major areas of interest are Database Systems, Data Mining and Operating systems. She has supervised many M.Tech and 8 Ph.D thesis. She has authored/co-authored more than 160 research papers and articles for various national and international journals/conferences. She has visited and presented papers abroad at places like San Francisco, Las Vegas and Spain. She has completed AICTE sponsored project in the area of education data mining as Co-Principal Investigator. She has authored books on "Data Structures using C" and "Compiler-Construction and Design". Other than organizing various workshops and lectures she successfully organized an IEEE International Conference on Data mining and Intelligent Computing (ICDMIC-2014) held at IGDTUW and another IEEE International Conference on information processing (IICIP 2016) at DTU. She is a life member of professional bodies like CSI, ISTE and Senior member of IEEE, USA

**Indu Singh** is currently working as Assistant Professor in Computer Science Engineering Department at Delhi Technological University, Delhi, India. Singh has received her B.Tech in 2010 in Computer Science Engineering and M.Tech degree in Information Security from Ambedkar Institute of Advanced Communication Technologies & Research, Govt. of NCT Delhi in 2012. She is currently pursuing her Ph.D in Computer Science Engineering from Delhi Technological University, Delhi with specialisation in Data Mining and Information Security. Her research interests include Database Systems, Data Mining, Information Security, Machine Learning, Fuzzy systems and Swarm Intelligence. She has several papers published in International conferences and Journals of IEEE, Elsevier, Springer and ACM. She has received IEEE Best Paper Award in ICACCI-2016. Singh is also serving as a reviewer for Computers in Biology and Medicine (Elsevier), Soft Computing Journal (Springer), Journal of Ambient Intelligence and humanized computing (Springer) and various International Conferences of Elsevier, Springer and IEEE. She is also a member of IEEE.