D-FICCA: A density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks

Shahaboddin Shamshirband a,b,* Amineh Amini c, Nor Badrul Anuar b, Miss Laiha Mat Kiah b, Ying Wah Teh c, Steven Furnell d

a Department of Computer Science, Chalous Branch, Islamic Azad University (IAU), 46615-397 Chalous, Mazandaran, Iran
b Department of Computer System and Technology, Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
c Department of Information Systems, Faculty of Computer Science and Information Technology, University of Malaya (UM), 50603 Kuala Lumpur, Malaysia
d Centre for Security, Communications and Network Research (CSCAN), Plymouth University, Plymouth PL4 8AA, United Kingdom

Article info
Article history:
Received 30 December 2013
Received in revised form 16 April 2014
Accepted 23 April 2014
Available online 10 May 2014

Keywords:
Imperialist competitive algorithm
Density-based clustering
Fuzzy
Intrusion
WSN

Abstract
Owing to the scattered nature of Denial-of-Service attacks, it is tremendously challenging to detect such malicious behavior using traditional intrusion detection systems in Wireless Sensor Networks (WSNs). In the current paper, a hybrid clustering method is introduced, namely a density-based fuzzy imperialist competitive clustering algorithm (D-FICCA). Hereby, the imperialist competitive algorithm (ICA) is modified with a density-based algorithm and fuzzy logic for optimum clustering in WSNs. A density-based clustering algorithm helps improve the imperialist competitive algorithm for the formation of arbitrary cluster shapes as well as handling noise. The fuzzy logic controller (FLC) assimilates to imperialistic competition by adjusting the fuzzy rules to avoid possible errors of the worst imperialist action selection strategy. The proposed method aims to enhance the accuracy of malicious detection. D-FICCA is evaluated on a publicly available dataset consisting of real measurements collected from sensors deployed at the Intel Berkeley Research Lab. Its performance is compared against existing empirical methods, such as K-MICA, K-mean, and DBSCAN. The results demonstrate that the proposed framework achieves higher detection accuracy 87% and clustering quality 0.99 compared to existing approaches.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

Wireless Sensor Networks (WSNs) provide an ideal schema for gathering data as opposed to sensor nodes and data transmission through wireless networks. This network type finds application in the military [1], health care monitoring [2] and disaster response [3] among others. The prevailing application designs for wireless sensors afford superior flexibility in establishing communications and increasing system automation but are deficient in security and privacy [4–6]. The core weakness of these sensor nodes lies in the limited-resource devices, i.e. power and processing units [7]. For this reason, vulnerability to various security threats is especially high. Meanwhile, adversaries may possess passive and active abilities to acquire secret information, such as keys stored in a compromised node by eavesdropping [8], as well as Denial-of-Services (DoS) attacks. Thus, the wireless medium becomes occupied, increasing the probability of packet collisions within the interfering signal’s range, causing, in both instances, additional consumption of sensor node energy [9].
In mitigation of security attacks, the Soft Computing (SC) approach incorporates Intelligent Intrusion Detection Systems with Preventions (IIDPSs) to detect and impede abnormal traffic patterns that diverge from the modeled, expected, normal traffic behavior [10–12]. A simple inspection packet mechanism was proposed by Tsunoda et al. [13] to avoid stateful inspection against Distributed Reflective Denial-of-Service (DRDoS) attacks. Misra et al. [14] utilized a learning automaton for the prevention of Denial-of-Service. While intrusion detection-based soft computing approaches, including misuse, anomaly and hybrid, display relatively reasonable performance regarding detection accuracy and minimal resource consumption, they fail to detect “misuse”. Hence, applying optimization techniques that shield the wireless sensor network infrastructure by maximizing detection accuracy remains a challenge [15]. As such, an attempt is made through this research to address the problem of security by applying an optimization-based clustering mechanism.

Clustering techniques are part of numerous applications [16–18]. One of the clustering approaches is a partitioning-based clustering algorithm, such as K-mean [19] and Fuzzy C-mean. For instance, Fuzzy C-mean proposed by Kumarage et al. [20] helps form a cluster founded on distance metrics to identify abnormal behavior in WSNs. However, partitioning clustering algorithms find only spherical clusters while the clustering outcome is influenced by noise.

In dealing with the drawback of partitioning algorithms, recent approaches integrate density-based clustering methods [21]. In density-based clustering algorithms, dense areas of objects in the data space are considered clusters, which are separated by low-density areas (noise). Some examples of density-based algorithms include DBSCAN [21], FlockStream [22] and DENCLUE [23]. DBSCAN is used for anomaly detection to identify different network intrusions in which sensors capture all network traffic and the system analyzes the content of individual packets for malicious traffic [24]. They are not only able to detect arbitrarily shaped clusters but are also useful in identifying noise. In addition, having advance knowledge of the number of clusters is not required.

For issues with anomalous attack modifications, the evolutionary algorithm (EA) is considered a high-accuracy detection method to be used in constructing an intelligent detection model and to automatically identify inconsistent activities [15]. The imperialist competitive algorithm (ICA) is an evolutionary computation algorithm proposed by Atapchos-Gargari and Lucas [25]. It was inspired by the concept of imperialism, whereby powerful countries attempt to seize and colonize others. ICA has recently been successfully employed in many engineering applications [26–29]. K-MICA is introduced as a hybrid unsupervised data partitioning method that incorporates an ICA with K-means for effective data clustering [30]. Although K-MICA provides robust clustering, it fails to form arbitrary-shaped clusters or handle noise, and it necessitates the number of clusters in advance. This hybrid-based clustering has motivated us to recommend a novel hybrid clustering algorithm. In this research, ICA clustering is modified using FLC and DBSCAN. In the suggested scheme, FLC is implemented to modify the assimilation operator in the competition phases of ICA. Fuzzy min–max is utilized in the assimilation operator rather than the Boltzmann selection strategy whose main shortcoming [31] is that upon exploring for equal selection among all possible actions, it selects the worst performing action to assign colonies to imperialists. In addition, DBSCAN is employed in ICA clustering, and clustering quality is enhanced by detecting noise as low-density areas [32].

The proposed density-based fuzzy imperialist competitive clustering algorithm (D-FICCA) approach identifies data distribution anomaly profiles, such as Denial-of-Service attacks (DDoS) that affect sensor behavior during processing. The principal, real wireless sensor data sets are derived from publicly available sensor data distributions at Intel Research Laboratories (IRL) [33] and the Australian Research Council’s [34] research network on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP). In this research work, the IRL dataset was selected for experimentation performed to evaluate the detection (Utility function) accuracy for normal/abnormal data instances, false alarm rate and clustering quality (Silhouette Coefficient).

The main contributions of this paper are as follows:

(i) Use ICA clustering for attack detection in WSNs.
(ii) Present the modified ICA algorithm with FLC.
(iii) Use a density-based clustering algorithm to detect any shape clusters and effectively handle noise.
(iv) Propose a cost function of ICA clustering.
(v) Evaluate the proposed D-FICCA algorithm for anomaly detection (DDoS) attacks in terms of false alarm rate detection accuracy and clustering quality in WSNs.

The rest of the paper is organized as follows. Section 2 introduces the ICA clustering. DBSCAN is described in Section 3. In Section 4, a modified ICA is suggested based on D-FICCA for anomaly detection in WSNs. In Section 5, the experimental results are presented. Section 6 evaluates and analyses the proposed method. Finally, the overall conclusions are presented in Section 7, including highlights of recommended future directions of research and development for practical applications.

2. The imperialist competitive algorithm (ICA)

In recent years, ICA has emerged as a new, evolutionary algorithm [25], and it has been applied to obtain optimal solutions in various applications [35,36]. It assists in solving certain optimization problems [37–41]. Unlike the majority of meta-heuristic algorithms, ICA was not inspired by a nature-based evolutionary phenomenon, but it implements the socio-political process of imperialism of controlling many countries and using their sources once colonies are dominated by rules. If one empire loses its power, the others will compete to take its place. In ICA, this process is simulated by individuals known as countries [42].

The algorithm begins with a random initial population and objective function which is computed for them. The
most powerful countries are selected as imperialists and the others are colonies. Next, competition between imperialists takes place to obtain more colonies. The best imperialist has greater chances of possessing the most colonies. One imperialist with its colonies thus makes an empire [43]. Fig. 1 displays the initial population of each empire, with superior empires having greater colonies and the weaker ones, smaller. In the figure, Imperialist 1 is the most powerful and has the most colonies.

Following colony distribution among imperialists, the colonies approach their associated imperialist countries. This movement is shown in Fig. 2, which demonstrates that a colony moves towards its related imperialist by \( x \) units, where \( x \) has a uniform distribution described by Eq. (1):

\[
X \sim U(0, \beta d)
\]  

(1)

where \( \beta \) is a number greater than 1 and is the distance between the colony and the imperialist. Moreover, there is a deviation of \( \theta \) in the motion of the colony toward the imperialist, defined in Eq. (2):
where $\gamma$ is a parameter that adjusts the deviation from the original direction (Fig. 3).

If, after this movement, one of the colonies has more power than its relevant imperialist, they will switch positions. For competition among empires to initiate, the total objective function of each empire is calculated, depending on the objective function of both the imperialist and its colonies. After competition starts, the weakest empire loses its control and other powerful ones attempt to gain it. An empire that has lost all its colonies will collapse. Finally, the most powerful empire will take possession of the other empires and win the competition. Fig. 3 illustrates the imperialistic competition.

$$\theta \sim U(-\gamma, \gamma)$$

(2)

To apply ICA for clustering, Niknam et al. [30] introduced K-MICA as a hybrid method that incorporates ICA and K-means for effective data clustering. Despite K-MICA being able to provide robust clustering, it cannot form an arbitrary cluster shape or handle noise, and it requires the number of clusters in advance, especially in wireless sensor network environments [30]. However, hybrid-based ICA is adapted to clustering techniques to improve system convergence as well as the time complexity. Thus, the proposed ICA-based clustering methods have motivated us to modify the ICA-based clustering by incorporating the DBSCAN algorithm with Fuzzy-based ICA. The aim is to exploit the benefits from DBSCAN and fuzzy in ICA clustering in order to reduce the weaknesses of previous hybrid-based ICA methods in terms of

![Fig. 3. Imperialistic competition.](image)

![Fig. 4. Comparison between DBSCAN and K-means on a small synthetic data set.](image)
detection accuracy and false alarm rate as well as clustering quality.

3. Density-based clustering algorithm

Clustering is a prominent task in data mining techniques, and different clustering algorithms have been developed. Clustering methods are categorized as distance-based [44] and density methods [45]. Distance-based methods determine the similarity between two objects in a cluster by the distance, e.g., Euclidean, whereas in density-based methods, the similarity may be defined by connectivity according to density or contiguity, and may not necessarily rely on the absolute distance between two objects. Similarity measures play a fundamental role in the design of clustering methods. Clustering algorithms, which are founded on distance metrics like K-means, have the following limitations. First, they are aimed at identifying spherical clusters but are incapable of revealing clusters with random shapes. However, non-convex and interwoven clusters are seen in many applications. Second, the K-means algorithm cannot detect noise and outliers.

Density-based methods scan for high-point density regions that are separated by areas of lower-point density. Not only do they find arbitrary shape clusters but they also handle noise and do not necessitate prior knowledge of the number of clusters \((k)\) as the K-means algorithm does. Fig. 4 shows a comparison between K-means and DBSCAN on a small synthetic data set.

Density-based spatial clustering of applications with noise (DBSCAN) is the most successful density-based clustering method due to accuracy. It can effectively discover clusters with different local density and DBSCAN has been applied in a multitude of applications including earth environments for a long time [46]. As of late, it has found use for medical purposes, including serving as a pre-processing phase for the prediction of Alzheimer’s disease [47] and for skin cancer [48]. DBSCAN is used to identify unknown intrusions, whereby sensors capture all network traffic and the system analyzes the content of individual packets for malicious traffic [49]. In this research work, the DBSCAN algorithm works to cluster data, with the aim to identify and report abnormal information. It has the ability to form arbitrary shape clusters and deal with noise effectively. The distribution zone of normal values is relatively small with high density, while the abnormal values form a low-density distribution zone.

Fig. 5 shows that the DBSCAN algorithm manages potential noise and detects random shape clusters. The
K-means algorithm assigns noise to clusters as blue and green data points. Thus, DBSCAN detects noise indicated as red data points. Furthermore, DBSCAN detects arbitrary shape clusters concisely, while K-means shapes clusters incorrectly. The pseudo code of DBSCAN is given in Algorithm 1 in parallel to Fig. 5.

The overall architecture of the DBSCAN algorithm is outlined as Algorithm 1 in five steps by adjusting the input and outputs parameters. The input parameters are D, MinPts, and epsilon, where D is a dataset, MinPts is the minimum number of data points in a predefine radius (epsilon). The output contains high and low-density regions. The high-density regions form clusters with arbitrary shapes, while the low density areas indicate noise present in the dataset. Thus, DBSCAN performs the following steps for each data point (p) in the data set:

Step 1: Start with an arbitrary starting point in dataset (D) that has not been visited.
Step 2: Calculate the distance of each data point (q) that exists in (ε) threshold neighborhood of the data point (P). The aim is to find the number of data points present in the neighborhood radius (Nε(p)).

\[ N_ε(p) = \{ q \in D : \text{dist}(p, q) \leq \varepsilon \} \]  

Step 3: Compare the number of neighborhood radius (Nε(P)) with MinPts. The comparison is meant to clarify the status of the data points. In other words, if Nε(P) is less than MinPts, it is considered as a noise, otherwise, it is assigned to a new cluster.

\[ |N_ε(p)| \geq \text{MinPts} \]  

Step 4: If data point (P) is assigned to be part of the cluster, for all ε neighborhoods of (P), repeat the procedure from Step 2 until all points in the cluster are determined.
Step 5: A new unvisited point is retrieved and processed, leading to the discovery of a new cluster or noise.
Step 6: This process continues until all points are marked as having been visited.

Using density-based clustering technique for intrusion detection avoids general clustering problems such as sensitivity to initialization, specification of number of clusters, or detection of particular cluster shapes [50–53]. The main idea of intrusion detection based on DBSCAN is that most of the data is normal, and normal data are gathered together into a high-density cluster, while the invasion data is vary few, and very different with normal data. Therefore, thorough clustering, invasion data would be a low-density cluster. As a result, the largest cluster of data is normal network data, while the small cluster of data that is considered to be the invasion data.

4. Modified ICA

4.1. Modified cost function based on DBSCAN

Most cost functions in ICA clustering measure the distance of each data point from their related clusters [30]. However, DBSCAN clustering results in random shapes and requires a cost function that is not dependent on distance from the cluster center. In our scheme, cost function measures the compactness of the cluster to which the data point belongs. Compactness is calculated based on the average distance between the data point and all other objects in the cluster to which the data point belongs. The smaller cost function value represents the cluster’s compactness.

4.2. Fuzzy imperialist competition clustering algorithm

To improve the convergence velocity and precision of ICA in the assimilation step, the standard imperialist competition algorithm was modified as follows: ICA was incorporated to the fuzzy logic controller (FLC) and is now called FICCA. This was done to revise the assimilation operator. Fuzzy min–max was applied rather than the Boltzmann action selection strategy. The main drawback of the Boltzmann strategy [31] is that during exploration to equally select among all possible actions, it may choose the worst performing action to assign colonies to imperialists. For this reason, the fuzzy set adapts to the action selection strategy to adjust its rules in order to avoid possible faults of the worst imperialist selection. Subsequently, the Boltzman action selection strategy for conventional ICA is initially discussed, followed by the FICCA details. To assign
colonies to the imperialist, ICA performs the assigning procedure based on the Boltzmann selection strategy \( (P_i) \) [31] as shown in Eqs. (5) and (6).

\[
P_i' = \exp^{-\eta \text{ImpCost}_i(t)/\max(\text{ImpCost}_i(t))}
\]

\[
P_i = \frac{P_i'}{\sum_{j=1}^{n\text{Emp}} P_j'}
\]

where \( \eta \) is the pressure coefficient. The total power of an empire is mainly affected by the imperialist country’s power. However, the power of an empire’s colonies has an effect, albeit negligible, on the total power of that empire. This fact is modelled by defining the total cost of an empire in Eq. (7).

\[
T \cdot C_n = \text{Cost(imprialist}_n) + \xi \text{mean(Cost(colonies of empire}_n))
\]

where \( T \cdot C_n \) is the total cost of the \( n \)th empire and \( \xi \) is a positive small number. A small value for \( \xi \) causes the total power of the empire to be determined by only the imperialist, and increasing it will boost the colonies’ role in determining the empire’s total power.

In order to identify the optimal action of colonies in assigning an imperialist, the fuzzy reward signal \((f_j)\) states are considered in the Boltzmann selection strategy to model the manner of the fuzzy selection agent in Eq (8).

Fuzzy rules assign a weight to all possible next states. Linked to the threshold value, optimal cost may be achieved. Thus, the fuzzy actions that lead to Optimal Confidence \((OC)\) less than \((OC)_{th}\) should be rewarded with a positive value, while those actions producing \(OC\) higher than \(OC_{th}\) should be penalized with a negative value. The reinforcement signal used in this work is formally defined by:

\[
f(t+1) = \begin{cases} 
100, & \text{if } OC_{\text{measured}}^x(t) < OC_{\text{th}} \\
-100, & \text{otherwise}
\end{cases}
\]

where \( f(t+1) \) is the reinforcement signal for the \( k \)th colony in iteration \( t+1 \). The value of \( OC_{\text{measured}}^x(t) \) is calculated as the fuzzy min–max weighted average as per Eqs. (7) and (8):

\[
\text{Optimal Confidence} = \text{output}(C_j) = \left( \sum_{j=1}^{N} x_j C_j \right) / \left( \sum_{j=1}^{N} x_j \right)
\]

where \( N \) is the number of rules, \( x_j \) is the degree of truth for rule \( j \), and \( C_j \) is the selected output’s constant value for the same rule. The FLC inputs, given by the Time response \((Tr)\), Buffer size \((Bs)\), and Count \((Co)\) correspond to the fuzzy state of the DoS attack detection \(S(t)\),

\[
S(t) = [\text{Time response. Buffer size. Count}] = [Tr. Bs. Co]
\]

The linguistic variables of Time response \((Tr)\), Buffer size \((Bs)\), and Count \((Co)\) act as inputs and the Detect Confidence \((DC)\) acts as an output are used in the experiments.

Three fuzzy sets are identified in the current Buffer size \((Bs)\), whose linguistic terms are ‘Low’ \((L)\), ‘Medium’ \((M)\), and ‘High’ \((H)\). These three fuzzy sets discriminate the cases when \((Bs)\) is less than \((3 \text{ k})\), which has been defined the length of packet received from source during specified time window. The output linguistic variable represents the system’s Detect Confidence \((DC)\) in the presence of abnormal behavior. To illustrate, if the confidence value is higher than \(80\), then the system is more than \(80\%\) certain that there is an abnormal entity, if the detection confidence is smaller than \(40\), it is more likely that there is no abnormality. However, input and output variables give us a notion of how traffic connection is changing. The membership functions are triangular or trapezoidal. Table 1 demonstrates the results of applying one of the possible min–max action selections.

Fuzzy min–max action selection mechanisms can reason with precise information, explain decisions for a single colony agent well, and the actions performed in decision making acquired are feasible even when the complexity of anomaly variables is high. However, the action decision policy accuracy adapts to each colony so as to mitigate the problem of Boltzmann action selection through fuzzy weight strategy policy whereby the subject colony gets assigned to an object imperialist.

4.3. Proposed density-based fuzzy imperialist competitive clustering algorithm (D-FICCA)

The proposed ICA clustering-based intrusion detection strategy is primarily a combination of the density method and fuzzy logic controller. The modified ICA-based detection system operates to sense DDOS attacks, where the sink node selects the best strategy of detecting an immediate attack and reports it to the base station. Regardless of whether the attacks are carried out on a regular or irregular basis, the sink node in terms of IDS can adjust its optimization density-based clustering parameters through fuzzy rules to identify future attacks. In our scheme, the proposed D-FICCA steps are described below:

Step 1: Generate an initial population

An initial population of input sensor data is generated by DBSCAN [21] initialization as follows:

\[
P = \begin{bmatrix} 
X_1 \\
X_2 \\
\vdots \\
X_{N_{\text{initial}}}
\end{bmatrix}
\]

\[
x_i = \text{colony}_i = [x_{i1}, x_{i2}, \ldots, x_{im}] 1 < i < n \]

\[
C_j = \text{Cluster}_j = [c_1, c_2, \ldots, c_m] 1 < j < m
\]

\[
N_{\text{new}} = N_{\text{initial}} - N_{\text{noise}}
\]

\[
P = \text{the population; } X_i \text{ is one of the colonies; } N_{\text{initial}} \text{ is the number of the population and } d \text{ is the number of dimensions of each colony; } m \text{ is the number of arbitrary shape clusters } (C_j); \text{ and } N_{\text{new}} \text{ is calculated based on the deduction of } N_{\text{initial}} \text{ from the number of noise } (N_{\text{noise}}).
Step 2: Calculate cost function value

The cost function is evaluated for each colony as follows:

Step 2.1: For each data point \( (x) \), the following distance is calculated.

- \( \text{cost}(x) \): the average distance between \( (x) \) and all other objects in its colony

\[
\text{cost}(x) = \frac{\sum_{x' \in C_i, x' \neq x} \text{distance}(x, x')}{|C_i| - 1}
\] (13)

Step 3: Sort the initial population according to objective function values.

The initial population ascends based on the value of its objective function.

Step 4: Select the imperialist states.

Colonies with the maximum objective function are the selected imperialist states and the remaining ones become these imperialists’ countries.

Step 5: Divide colonies among imperialists.

To proportionally divide the colonies among imperialists, the normalized cost of an imperialist is defined by Eq. (12).

\[
\text{Cost}_n = \text{cost}_n - \max_i \{\text{cost}_i\}
\] (14)

where \( \text{cost}_n \) is the cost of the \( n \)th imperialist and \( \text{Cost}_n \) is its normalized cost. Having the normalized cost of all imperialists, the normalized power of each imperialist is defined by Eq. (15).

\[
P_n = \frac{\text{Cost}_n}{\sum_{i=1}^{N_{imp}} \text{Cost}_i}
\] (15)

The initial colonies are divided among empires in order of their power. The initial number of colonies of the \( n \)th empire is given in Eq. (16).

\[
N \cdot \text{Cost}_n = \text{round}(P_n \cdot N_{col})
\] (16)

where \( N \cdot \text{Cost}_n \) is the initial number of colonies of the \( n \)th empire and \( N_{col} \) is the total number of initial colonies. To divide the colonies, \( N \cdot \text{Cost}_n \) of the colonies are randomly chosen and given to the \( n \)th imperialist. These colonies, along with the \( n \)th imperialist, form the \( n \)th empire.

Step 6: Perform the DBSCAN algorithm for each empire.

Step 7: Move colonies toward their imperialist states based on fuzzy min–max (as described in Section 4.2).

Step 8: Check the cost of all colonies in each empire.

Step 9: Run imperialistic competition.

Step 10: Remove the weakest empire.

Step 11: Inspect the number of empires; if it is 1, then go to Step 7.

The pseudo code of the proposed D-FICCA is shown in Algorithm 2.

**Algorithm 2 D-FICCA**

1: Generate an initial population using DBSCAN
2: Calculate the cost function for initial population
3: Sort initial population based on the cost function values
4: Select Imperialist states
5: Divide colonies among imperialist
6: Use DBSCAN algorithm for each empire
7: while there is one empire do
8: \( \text{while} \) all empires are selected do
9: \( \text{Select} \) \( i \)th empire
10: \( \text{while} \) all colonies are selected do
11: \( \text{Select} \) \( j \)th colony
12: Move the colony toward the imperialist state using fuzzy min–max
13: Use mutation to change the direction of colony
14: Calculate the cost function value for two new population
15: Compare and select the best one
16: Replace \( j \)th colony with the new one
17: end while
18: Sort all colonies based on their cost function
19: Check cost of all colonies in each empire
20: \( \text{if} \) there is a colony which has lower cost than its imperialist \( \text{then} \)
21: Exchange the position of the colony and the imperialist
22: \( \text{end if} \)
23: Update the position of the \( i \)th empire
24: end while
25: Calculate total cost of empires
26: Find the weakest empire
27: Give one of its colony to the winner empire
28: Check the number of colony in each empire
29: \( \text{if} \) there is an empire without colony \( \text{then} \)
30: Remove the empire and gives its imperialist to the best empire
31: \( \text{end if} \)
32: end while

5. Experimental results

5.1. WSN model

Considering the limitations in existing methods, a novel anomaly detection framework is required that deals with the specific constraints of sensor nodes and challenges pertaining to the dynamic nature of sensed data itself. The proposed approach takes these aspects into consideration to identify distributed attacks through local data correlations in a distributed environment (Fig. 6).

Fig. 6 shows that 10 sensors (motes) were selected from the corridor (sensors 1–10) in the proposed method. In this case, the sensors read the data. Basically, an epoch is a monotonically increasing sequence of numbers from each mote. The dataset consists of Moteids ranging from 1 to 4, and data from some motes may be missing or truncated. The temperature is in degrees Celsius. Humidity is
temperature-corrected relative humidity, ranging from 0% to 100%. Light is in Lux (a value of 1 Lux corresponds to moonlight, 400 Lux to a bright office, and 100,000 Lux to full sunlight). Voltage is expressed in volts, between 2 and 3; the batteries in this case were lithium ion cells, which maintain a fairly constant voltage over their lifetime; note that voltage variations are highly correlated with temperature. In our scheme, only three features, namely Humidity, Temperature, and Voltage, were the focus.

In the present research study, Fig. 7 illustrates the distributed network with hierarchical routing that consists of clusters (C), their coordinators known as Cluster Heads (CHs) as well as member sensor nodes (S). In our scheme, it is assumed that the Cluster Head is a Sink Node (SN) in every cluster. The sink node monitors the behaviour of sensor nodes by collecting data from the member sensor nodes and transmitting the critical status, or the attack information of the sensor nodes, to a Base Station (BS). The proposed D-FICC algorithm is mapped into the base station to identify the anomalous behaviour of sensors affected by attackers.

The route from a sensor node to a base station is deemed a distributed-hierarchical path that creates a hierarchical system with numerous routes, which is the main feature of cluster-based WSNs. Sensor nodes function independently to prevent the failure of all sensor nodes in a situation where one fails. The sensor node redundancy approach increases the overall reliability in distributed hierarchical systems. Fig. 7 illustrates how sensor nodes send collected data from a sink node to a base station via other adjacent sink nodes, and the base station receives data only if all sink nodes within the routing formation are actively functioning. Hence, a set of clusters on a route counts as a set of independent distributed-connected elements. An attack in this scenario can target the WSN in multiple ways. For instance, Denial-of-Service attacks can originate either from within or outside the wireless sensor network.

5.2. Data sets

Real sensor data representing different data distributions for normal and anomalous behavior are used. The data sets consist of observations from real sensor motes as well as artificially created data points introduced to act as anomalies. The first data set is based on the publicly available Intel lab data comprising real measurements collected from 54 sensors deployed at the Intel Berkeley Research Lab. The data gathered includes time stamped topology information along with temperature, humidity, light and voltage values as the measured parameters from Mica2Dot sensors with implemented weatherboards. The data format is presented in Table 2. Since the data is not annotated and does not contain any labeled information on observed anomalies, it is used for evaluation as follows. Ten different data distributions are created by considering the available node data of the first ten sensor motes (motes 01–10). The three attributes of temperature, humidity, and voltage together are used as data vectors corresponding to individual observations, with 4700 observations per single node [54].

First, the data is cleaned manually by identifying extreme values and removing them using scatter plots. Upon completing this step, the set of randomly generated data is introduced in each mote with Poisson’s statistical function. The communication routing protocol works with the sensor networks to transmit data from the cluster head to the base station. The base station is responsible for identifying anomalous behavior using the proposed D-FICC algorithm. Therefore, the artificial attack data is simulated using Matlab data generation algorithms.

5.3. Evaluation metrics

In this section, the proposed algorithm is analyzed. For all experiments, 10 motes (sensors 1–10) were used and the proposed D-FICC algorithm was adopted in mote 5, which is associated to the base station agent. To verify and validate the proposed method, a Matalab 7 platform on a PC with Intel corei5, 3.10 GHz CPU, 4 GB memory and 500 GB hard disk with Windows 7 Professional operating system, was utilized.

The proposed method parameters throughout the experiment are tuned as follows:

- The amount of preliminary countries fluctuates between 200 and 250.
- 5% of the initial countries are selected as imperialists.
- 10% of each empire population undergoes revolution.

The proposed method’s accuracy was evaluated based on two criteria. First, clustering quality was measured with the Silhouette Coefficient (SC). Furthermore, a utility function (UF) was used to measure detection accuracy. In the next sections, the experiments are discussed in detail. Based on the presented results, the superiority of our proposed method over existing methods in this domain will be illustrated.

5.3.1. Silhouette Coefficient (SC)

To evaluate D-FICCA, the Silhouette Coefficient was employed. Methods already being used are RMSE, SSE in clustering result evaluation. RMSE was based on the distance of every data point in a cluster to its centroid. Despite its popular usage in the evaluation of clustering results in literature, RMSE favoured spherical clusters, as opposed to arbitrarily shaped clusters. In our experiments, we found that the Silhouette function was a more appropriate measure that did not have this limitation. Therefore, it was included in the evaluation. A Silhouette width measured the compactness of the cluster, which was more appropriate to discover non-spherical clusters [53,55].

The Silhouette Coefficient [56] is a criteria for measuring clustering quality. It actually reflects the compactness of the cluster to which the data point belongs, besides capturing the degree of separation from other clusters. In fact, for each data point the Silhouette Coefficient calculates the average distance between that point and all other objects in the cluster to which the data point belongs a(o) as well as the minimum average distance from every data point to
all clusters to which the data point does not belong $b(o)$. The Silhouette Coefficient $S(o)$ is calculated mathematically in Eq. (17):

$$S(o) = \frac{b(o) - a(o)}{\max\{a(o), b(o)\}} - 1 < S(o) < 1$$

where the value of $a(o)$ reflects the cluster’s compactness. The $b(o)$ value captures the degree to which $o$ is separated from other clusters. The smaller value of $a(o)$ indicates the compactness of the cluster. A larger $b(o)$ value indicates the separation of $o$ from other clusters. If the Silhouette Coefficient value of $o$ approaches one (1), the cluster containing $o$ is compact [$a(o) < b(o)$]. If the Silhouette Coefficient value is negative [$a(o) > b(o)$], then $o$ is closer to the objects in alternative clusters.

5.3.2. Utility function (UF)

To appraise the efficacy of the associations determined by D-FICCA and to establish the applicability of the rule at every point in time, Eq. (18) was utilized in this work, as suggested by [57], and [58]. Table 3 below describes the parameters of the utility function:

$$U = \rho \ast SP - \beta \ast FN - \theta \ast FP$$

5.3.3. Parameter setting

The parameters required for implementing the D-FICCA algorithm are $\gamma, \beta, \xi, N_{pop}, N_{imp}$ are shown in Table 4.

6. Evaluation and analysis

The proposed D-FICCA algorithm is evaluated using real data sets collected by IRL. This is performed in terms of clustering accuracy, false alarm rate, and quality of clustering for both normal and anomalous data points, in comparison with existing empirical clustering methods, e.g. K-MICA [30], K-mean [59], and DBSCAN [60].
Table 2
Intel data set format; each observation consists of 4 mote attributes and 3 measured parameters.

<table>
<thead>
<tr>
<th>Attr. 1</th>
<th>Attr. 2</th>
<th>Attr. 3</th>
<th>Attr. 4</th>
<th>Para. 1</th>
<th>Para. 2</th>
<th>Para. 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: y-m-d</td>
<td>Time: h:m:s</td>
<td>Epoch</td>
<td>Moteid</td>
<td>Temperature</td>
<td>Humidity</td>
<td>Voltage</td>
</tr>
</tbody>
</table>

Table 3
Parameters of the utility function.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>It is a utility</td>
</tr>
<tr>
<td>ρ</td>
<td>Symbols the weight of effective prediction, ρ = 0.75</td>
</tr>
<tr>
<td>SP</td>
<td>Characterizes true confident rate of attack patterns</td>
</tr>
<tr>
<td>β</td>
<td>Signifies the weight of failed estimate (attack but no defense), β = 1</td>
</tr>
<tr>
<td>FN</td>
<td>Represents false negative of attack patterns. There are attacks but no defense</td>
</tr>
<tr>
<td>θ</td>
<td>Denotes the weight of failed prediction (defense but no attack), θ = 1</td>
</tr>
<tr>
<td>FP</td>
<td>Represents false positive of attack patterns. There is defense but no attack</td>
</tr>
</tbody>
</table>

6.1. Average cost for the proposed D-FICCA

To assess the proposed method, 10 sensors that differ in terms of attack time value were collected. The selected test sensors are seen in Fig. 8, and the same test was iterated 500 times on each sensor. Moreover, in every round, the value of one anomaly was randomly altered. The average cost for a proposed D-FICCA is shown in Fig. 8.

The results prove that our proposed method has high sensitivity with respect to minor changes in the sensor nodes due to attacks. The main reason for such performance in achieving superior UF and SC values is that the current method is based on D-FICCA. Performance analysis on data distributions based on Intel Research Lab data was done first. Data classification accuracy was investigated with respect to identifying anomalous and normal data points by calculating the values for sensitivity and specificity. Considering the assumed distributed hierarchical topology, results are given for each of the five local clustering phases at nodes S1–S10 and the final analysis at gateway node S5. Fig. 8 signifies the selected test sensors, and the same test was iterated 500 times on each sensor. Moreover, the value of one anomaly in every round was randomly altered.

6.2. Detection accuracy (D-FICCA)

The proposed Density-based Fuzzy Imperialist Competitive Clustering Algorithm (D-FICCA) with the utility function $UF = \rho \times SP - \beta \times FN - \theta \times FP$ was compared with existing soft computing-based clustering methods (K-MICA, K-means, and DBSCAN) regarding the attack detection precision of modeled Denial-of-Service attacks. According to a comparison between the average utility function and D-FICCA with cost maximization, the latter yielded an improvement as opposed to the K-MICA algorithm in sensor 5 (base station) (Table 5).

It is evident that the D-FICCA mechanism attained the utmost detection accuracy. It can also be inferred from Fig. 9 that detection accuracy per percentage of attacks is higher with the D-FICCA algorithm than with other methods.

In Fig. 9, the X-axis represents the percentage of malicious nodes in an attack, and the Y-axis indicates the accuracy rate. At higher attack frequencies, the proposed method (D-FICCA) displays greater accuracy scores.

6.3. Clustering quality

The quality of clustering results was compared to existing methods using the Silhouette Coefficient, which calculates the compactness of a cluster and the degree of separation from other clusters. The Silhouette Coefficient of D-FICCA in contrast to K-means, DBSCAN, and K-MICA is shown in Table 6. Very good clustering results were noted for D-FICCA on the 10 sensors of a real data set.

7. Conclusion and discussion

In this paper, the interaction between sensor nodes, sink node, base station and attackers in WSNs was studied, after which a hybrid, intrusion detection, density-based fuzzy imperialist competitive clustering algorithm (D-FICCA) theoretic detection and clustering mechanism was proposed. This system combines DBSCAN-based density clustering with fuzzy sets elements. As such, the proposed D-FICCA adapts to the base station agent to reinforce detection function against incoming DDoS attacks that may cause congestion and downtime in WSN communication as a result of flooding packets. This strategy-based modified ICA develops as a result of the continuous self-learning from prior attacks and the behavior in the fuzzy

Table 4
Values of the parameters for each of the five algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameter</th>
<th>Value</th>
<th>Algorithm</th>
<th>Parameter</th>
<th>Value</th>
<th>Algorithm</th>
<th>Parameter</th>
<th>Value</th>
<th>Algorithm</th>
<th>Parameter</th>
<th>Value</th>
<th>Algorithm</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FICCA</td>
<td>$N_{imp}$</td>
<td>100</td>
<td>K-MICA</td>
<td>$N_{imp}$</td>
<td>100</td>
<td>MICA</td>
<td>$N_{imp}$</td>
<td>100</td>
<td>ICA</td>
<td>$N_{imp}$</td>
<td>100</td>
<td>DBSCAN</td>
<td>MinPts</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>$N_{imp}$</td>
<td>5</td>
<td></td>
<td>$N_{imp}$</td>
<td>5</td>
<td></td>
<td>$N_{imp}$</td>
<td>5</td>
<td></td>
<td>$\epsilon$</td>
<td>10</td>
<td>K-means</td>
<td>$k$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\gamma$</td>
<td>0.4</td>
<td></td>
<td>$\gamma$</td>
<td>0.06</td>
<td></td>
<td>$\gamma$</td>
<td>0.06</td>
<td></td>
<td>$\gamma$</td>
<td>0.06</td>
<td></td>
<td>$\epsilon$</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>$\epsilon$</td>
<td>0.6</td>
<td></td>
<td>$\epsilon$</td>
<td>0.7</td>
<td></td>
<td>$\epsilon$</td>
<td>0.7</td>
<td></td>
<td>$\epsilon$</td>
<td>0.7</td>
<td></td>
<td>$\epsilon$</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td>iterations</td>
<td>500</td>
<td></td>
<td>iterations</td>
<td>500</td>
<td></td>
<td>iterations</td>
<td>500</td>
<td></td>
<td>iterations</td>
<td>500</td>
<td></td>
<td>iterations</td>
<td>500</td>
</tr>
</tbody>
</table>
learning decision making process to overcome the attacker. By defining fuzzy incentives for cooperation between imperialists and disincentives for fraudulent behavior, it has been determined that repeated interaction sustains cooperation, builds confidence and enhances reputation, something additionally offered by D-FICCA.
The density-based fuzzy imperialist competitive clustering algorithm (D-FICCA) theoretic, as a mechanism in IDS, is an invaluable tool for increasingly securing next-generation complex heterogeneous computing and networking environments against sophisticated attacks and attackers, beyond what is encountered today. A future initiative is to extend the proposed D-FICCA mechanism by incorporating data from various attack types and sources to further enhance its decision making capabilities in order to thwart existing or new attacks. Also as part of future research work on complementing D-FICCA, studying a game-based evolutionary algorithm is considered extremely significant.

Table 5
Detection Accuracy of D-FICCA for DDoS attacks.

<table>
<thead>
<tr>
<th>Experiments</th>
<th>K-MICA</th>
<th>D-FICCA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SP</td>
<td>FP</td>
</tr>
<tr>
<td>Case 1</td>
<td>96.3</td>
<td>1.2</td>
</tr>
<tr>
<td>Case 2</td>
<td>96.4</td>
<td>1.3</td>
</tr>
<tr>
<td>Case 3</td>
<td>97.1</td>
<td>1.5</td>
</tr>
<tr>
<td>Case 4</td>
<td>97.5</td>
<td>1.7</td>
</tr>
<tr>
<td>Case 5</td>
<td>97.7</td>
<td>1.9</td>
</tr>
<tr>
<td>Case 6</td>
<td>98.5</td>
<td>2.1</td>
</tr>
<tr>
<td>Case 7</td>
<td>98.7</td>
<td>2.4</td>
</tr>
<tr>
<td>Case 8</td>
<td>98.9</td>
<td>2.6</td>
</tr>
<tr>
<td>Case 9</td>
<td>99</td>
<td>3.1</td>
</tr>
<tr>
<td>Case 10</td>
<td>99.1</td>
<td>3.2</td>
</tr>
<tr>
<td>Case 11</td>
<td>99.4</td>
<td>3.3</td>
</tr>
<tr>
<td>Case 12</td>
<td>99.5</td>
<td>3.5</td>
</tr>
<tr>
<td>Case 13</td>
<td>99.6</td>
<td>3.7</td>
</tr>
<tr>
<td>Case 14</td>
<td>99.7</td>
<td>3.8</td>
</tr>
<tr>
<td>Case 15</td>
<td>98.39</td>
<td>2.52</td>
</tr>
</tbody>
</table>

Table 6
Clustering quality comparison based on the Silhouette Coefficient.

<table>
<thead>
<tr>
<th>Sensor 1</th>
<th>Sensor 2</th>
<th>Sensor 3</th>
<th>Sensor 4</th>
<th>Sensor 6</th>
<th>Sensor 7</th>
<th>Sensor 8</th>
<th>Sensor 9</th>
<th>Sensor 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-means</td>
<td>0.92</td>
<td>0.93</td>
<td>0.94</td>
<td>0.95</td>
<td>0.82</td>
<td>0.93</td>
<td>0.92</td>
<td>0.93</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>0.94</td>
<td>0.94</td>
<td>0.96</td>
<td>0.97</td>
<td>0.97</td>
<td>0.96</td>
<td>0.94</td>
<td>0.95</td>
</tr>
<tr>
<td>K-MICA</td>
<td>0.96</td>
<td>0.97</td>
<td>0.97</td>
<td>0.96</td>
<td>0.93</td>
<td>0.97</td>
<td>0.96</td>
<td>0.96</td>
</tr>
<tr>
<td>D-FICCA</td>
<td>0.97</td>
<td>0.98</td>
<td>0.99</td>
<td>0.97</td>
<td>0.94</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
</tr>
</tbody>
</table>

Fig. 9. Comparison of detection accuracy values.
Acknowledgments

This work is supported by the Ministry of Science, Technology and Innovation, Malaysia under Grant eScienceFund 01-01-03-SF0914.

Reference


S. Shamshirband et al. / Measurement 55 (2014) 212–226


