Detection of Anomalous Behavior in a Robot System Based on Machine Learning

Mahfuzul Nissan
Department of Computer Science
University of New Orleans
New Orleans, LA, USA
minissan@uno.edu

Sharmin Aktar
Department of Computer Science
University of New Orleans
New Orleans, LA, USA
saktar@uno.edu

***Abstract*—Ensuring the safe and reliable operation of robotic systems is paramount to prevent potential disasters and safeguard human well-being. Despite rigorous design and engineering practices, these systems can still experience malfunctions, leading to safety risks and operational inefficiencies. In this study, we present machine learning-based approach for detecting anomalies in system logs to enhance the safety and reliability of robotic systems. We collected logs from different scenarios using Coppeliasim and developed an anomaly detection system using various machine learning models, including Logistic Regression (LR), Support Vector Machine (SVM) and an Autoencoder. Our system was evaluated in two contexts, with results showing that the LR model consistently demonstrated superior performance in both contexts, while the Autoencoder model showed promise as an alternative in Context 2. This research highlights the potential of machine learning techniques, specifically autoencoders, in detecting anomalous behavior and improving the security and reliability of robotic systems.**

***Index Terms*—Robotic Systems, Anomaly Detection, Machine Learning, System Logs, Logistic Regression, Autoencoder, Contextual Evaluation, Coppeliasim, Quadcopter, Pioneer Robot**

I. Introduction

The rapid growth of robotic systems across various industries has revolutionized the way tasks are performed, offering tremendous improvements in productivity, efficiency, and automation. However, ensuring the safe and reliable operation of these systems is paramount to prevent potential disasters and safeguard human well-being. Despite rigorous design and engineering practices, robotic systems can still experience malfunctions, leading to safety risks and operational inefficiencies. Consequently, researchers and engineers are continually striving to develop innovative methods that enhance the safety and reliability of robotic systems, minimizing the occurrence of anomalies and maximizing system performance.

To address these challenges, advanced control techniques, safety sensors, and machine learning algorithms have emerged as pivotal tools in detecting and preventing potential anomalies in robotic system behavior. By leveraging these technologies, researchers are paving the way for a safer and more secure future for robotic systems. Among the promising machine learning techniques, autoencoders have garnered considerable attention due to their potential to detect anomalous behavior and improve the security and reliability of robotic systems.

In this research work, we explore the use of machine learning techniques to improve the safety and reliability of robotic systems. We collected system logs from two different scenarios involving a quadcopter and two Pioneer robots following predefined D* paths. These logs contained both normal and anomalous data. We then used supervised machine learning models, including logistic regression and support vector machines, as well as an autoencoder-based anomaly detection system to identify deviations from normal behavior. Our results demonstrate the effectiveness of these approaches in detecting anomalies and improving the security and reliability of robotic systems.

II. Related Work

In the realm of proactive anomaly detection for robot navigation, Ji et al. [1] proposed a proactive anomaly detection network (PAAD) that addresses the limitations of reactive anomaly detection methods. Traditional approaches lack the ability to alert robots before an actual failure occurs, leading to potential damage. PAAD overcomes this by predicting the probability of future failure based on planned motions and current observations using multi-sensor fusion. Their experiments demonstrated superior failure identification performance and real-time detection of anomalous behaviors while maintaining a low false detection rate.

Olivato et al. [2] conducted a comparative analysis on the use of autoencoders. As the deployment of robots in public spaces increases, so does the risk of cyber-security attacks. The authors addressed the problem of automatically detecting anomalous behaviors resulting from such attacks. Their approach involved extracting system logs, transforming the data into images, and training different autoencoder architectures to classify robot behaviors and detect anomalies. Experimental results in autonomous boats and social robot scenarios demonstrated the effectiveness and general applicability of their proposed method.

Park et al. [3] proposed an LSTM-based variational autoencoder (LSTM-VAE) that fuses multimodal sensory signals and reconstructs their expected distribution for anomaly detection in robot-assisted feeding. Their approach outperformed baseline detectors, achieving a higher AUC of 0.8710. The use of multimodal fusion techniques and advanced machine learning

algorithms holds promise in enhancing anomaly detection and promoting safer operations in complex robotic systems.

Utkin et al. [4] proposed a preprocessing procedure that utilizes autoencoder, to address two key challenges. Firstly, the autoencoder is used to reduce the dimensionality of training data and calculate the Mahalanobis distance, a reliable metric for detecting anomalies in robot systems or sensors. Secondly, the autoencoder is employed for transfer learning, where it is trained using target data representing extreme operational conditions. By reconstructing the source data, which consists of normal and anomalous observations from normal operation conditions, the authors establish an optimal threshold for decision-making based on the Mahalanobis distance.

III. Background and Motivation

In recent years, the widespread adoption of robotic systems across various industries has revolutionized productivity and efficiency. These systems play a vital role in automating tasks, enhancing precision, and improving overall operational capabilities. However, alongside these advancements, ensuring the safe and reliable operation of robotic systems has become a paramount concern. Despite the best design and engineering practices, these systems are not immune to malfunctions and anomalous behaviors that can lead to potential safety risks and operational failures.

The detection and prevention of anomalous behavior in robotic systems have emerged as critical research areas. Promptly identifying and mitigating abnormal system behavior is crucial to prevent disasters, minimize damage, and safeguard human well-being. Traditional reactive approaches to anomaly detection solely rely on monitoring the current state of the robot and identifying deviations from predefined norms. However, these methods lack the ability to anticipate failures and alert the system before an actual failure occurs. Such delays can be detrimental, resulting in substantial damage to the robot itself, the surrounding environment, or even human operators.

To overcome these limitations, there is a growing interest in leveraging machine learning techniques for proactive anomaly detection in robotic systems. Machine learning algorithms, particularly those based on deep learning, have shown remarkable capabilities in capturing complex patterns and detecting anomalies in various domains. Among these techniques, autoencoders have emerged as a promising approach for anomaly detection in robotic systems. Autoencoders are neural networks designed to learn and reconstruct the input data while compressing it into a lower-dimensional representation. By training on normal behavior patterns, autoencoders can identify deviations from the learned representations, enabling the detection of anomalous behavior.

The motivation behind this research lies in the need to enhance the security and reliability of robotic systems. Anomalous behavior in these systems can lead to severe consequences, ranging from operational disruptions to physical harm. Reactive approaches that detect anomalies based solely on the current robot state are insufficient in mitigating risks associated with unpredictable failures. By adopting machine learning approach, specifically autoencoders, we aim to develop a robust anomaly detection system capable of identifying abnormal behavior.

IV. Methodology

In this section, we describe the methodology used in our work on two scenarios simulated in the CoppeliaSim environment. The first scenario, referred to as Context 1, involves the navigation of a quadcopter. The second scenario, referred to as Context 2, involves the navigation of a Pioneer robot. Our approach involved following the D* path planning algorithm to create normal and anomalous data and detecting anomalies using machine learning models. The methodology section is divided into three subsections. The first subsection provides general information about the D* algorithm, which was used for path planning. The second and third subsections provide general information about the use of Support Vector Machine (SVM) and Logistic Regression (LR) machine learning models, respectively, for anomaly detection.

A. D* Algorithm

The D* algorithm also known as D-Star, is a popular path planning algorithm used in robotics and autonomous systems. It is designed to handle changing environments by enabling efficient and dynamic re-planning. The algorithm operates on a grid-based map representation of the environment, where each cell stores information about the cost of reaching that cell and an estimate of the remaining cost to the goal.

During exploration, the algorithm selects the cell with the minimum cost and updates the costs of its neighboring cells based on factors such as distance and the presence of obstacles. It dynamically adjusts the costs as it encounters changes in the environment, such as the addition or removal of obstacles.

The D* algorithm continues to explore and update the costs until it reaches the goal or determines that no feasible path exists. It provides a dynamic path from the start location to the goal, adapting to changes in the environment as they occur. Algorithm 1 shows the D* algorithm.

B. Logistic Regression

Logistic regression is a widely used statistical model for binary classification tasks, where the goal is to predict the probability of an event occurring or not occurring based on a set of input features. It is a linear classifier that maps the input features to the probability of the binary outcome using a logistic function.

The logistic regression algorithm calculates a weighted sum of the input features, applying these weights to determine their contribution to the final prediction. The logistic function, also known as the sigmoid function, is then applied to this sum to transform it into a value between 0 and 1. This value represents the predicted probability of the event occurring.

During the training phase, the model adjusts the weights of the input features to maximize the likelihood of the observed data. This is typically done using optimization algorithms such as gradient descent. Once trained, the logistic regression model

Input: Start cell, Goal cell
Output: Optimal path from start to goal
Initialize costs and estimates of all cells except goal;
Set cost of goal cell to 0;
Initialize an empty Open List;
while *Open List is not empty* **do**
  Select cell with minimum cost from Open List;
  if *Goal cell reached* **then**
    return optimal path;
  end
  else
    Update costs and estimates of neighboring cells;
    if *Changes detected in the environment* **then**
      Update costs and estimates accordingly;
    end
  end
end
return dynamic path from start to goal;

Algorithm 1: D* Algorithm

can make predictions by applying the learned weights to new instances and transforming the weighted sum using the logistic function. Equation for logistic regression:

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n)}}$$

Here, $P(y = 1|x)$ represents the probability of the event (e.g., class label y being 1) given the input features x. The coefficients β_0, β_1, β_2, ..., β_n correspond to the weights assigned to each feature x_1, x_2, ..., x_n. The logistic function $\frac{1}{1+e^{-z}}$ is used to transform the weighted sum z into a probability value between 0 and 1.

C. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm used for classification and regression tasks [6]. It works by finding an optimal hyperplane that separates the data points into different classes, maximizing the margin between the classes.

In SVM, the objective is to find the best decision boundary that maximizes the margin, which is the distance between the decision boundary and the closest data points from each class, known as support vectors. These support vectors play a crucial role in defining the decision boundary and determining the overall performance of the SVM.

Mathematically, SVM aims to solve the following optimization problem:

$$\min_{\mathbf{w},b} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i \right)$$

constrained by:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i = 1, 2, \ldots, N$$

where $\mathbf{x}_i$ is the feature vector of the i-th data point, y_i is its corresponding class label (-1 or 1), $\mathbf{w}$ is the weight vector, b is the bias term, $\boldsymbol{\xi}_i$ is the slack variable that allows for misclassifications, and C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification errors.

By solving this optimization problem, SVM finds the optimal values of $\mathbf{w}$ and b that define the decision boundary. The decision function can be expressed as:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

where $\mathbf{x}$ is the input feature vector, and the sign function determines the predicted class label based on the position of the input vector with respect to the decision boundary.

In our project, we employed the linear kernel, a fundamental component of the SVM algorithm. The linear kernel is represented mathematically as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ denote the input feature vectors. The linear kernel computes the dot product between these feature vectors, resulting in a linear function that captures the relationships between the data points.

V. Experiments

In this section, we present the experimental setup and implementation of the D* algorithm in CoppeliaSim to simulate the navigation of both a quadcopter and a Pioneer robot. The objective of these experiments was to implement and demonstrate the functionality of the D* algorithm in the CoppeliaSim environment for controlling the quadcopter and Pioneer robot. The focus was on successfully integrating and configuring the D* algorithm with both robot platforms, allowing them to navigate from their respective start locations to their goal locations within the defined workspace. While the D* algorithm itself was not modified, necessary adjustments were made to scale it to the CoppeliaSim workspace and ensure compatibility with both the quadcopter and Pioneer robot platforms.

A. CoppeliaSim

CoppeliaSim [5], also known as V-REP (Virtual Robot Experimentation Platform), is a versatile and widely-used robotics simulation software. It provides a comprehensive platform for simulating, programming, and testing various robotic systems and algorithms. CoppeliaSim allows users to create virtual environments and models, interact with them in real-time, and simulate the behavior and performance of robotic systems.

At its core, CoppeliaSim operates on a physics-based simulation engine, which accurately models the dynamics and interactions of objects within the simulated environment. This enables realistic simulations of robot movements, sensor readings, and interactions with the virtual world. CoppeliaSim supports a wide range of robot models, sensors, and actuators, allowing users to simulate diverse robotic applications.

B. Implementation of Quadcopter in CoppeliaSim

First, we created a workspace in CoppeliaSim with the specifications W = [xmin = -2.5, xmax = 2.5, ymin = -2.5, ymax = 2.5], representing the default floor dimensions. This workspace served as the environment in which the quadcopter would navigate. Next, we transformed the coordinates of the obstacles to fit within the CoppeliaSim workspace. This step ensured that the obstacles were correctly positioned and aligned with the virtual environment.

To define the start and goal locations, we created two cuboid objects in CoppeliaSim. The start location represented the initial position of the quadcopter, while the goal location indicated the desired destination.Using the D* algorithm, we computed a path connecting the start and goal locations within the workspace. The algorithm considered the presence of obstacles and dynamically updated the path based on any changes in the environment.

Additionally, we introduced a safety parameter to ensure that the quadcopter maintained a safe distance from the workspace boundaries. This parameter prevented the quadcopter from approaching the edges of the workspace, reducing the risk of collisions and ensuring safe navigation.

Finally, we implemented the quadcopter model in CoppeliaSim and programmed it to follow the computed path from the start to the goal location. The quadcopter's movement was synchronized with the D* algorithm, allowing it to traverse the workspace while avoiding obstacles and maintaining a safe distance from the boundaries.

C. Implementation of Pioneer Robot in CoppeliaSim

We implemented two Pioneer P3DX robots in the CoppeliaSim environment as part of our experimental setup. These robots were initialized at random states, with one robot positioned at the starting location and the other robot placed at the goal location. The goal was to exchange their initial positions while primarily following the D* path.

To achieve this, we generated collision-free path that enabled the robots to navigate through the workspace and exchange their positions while adhering to the computed D* path. These paths were carefully designed to ensure the robots could move safely without colliding with obstacles or each other.

Similar to the quadcopter implementation, we utilized the D* algorithm to compute the paths connecting the start and goal locations for the Pioneer P3DX robots. The algorithm accounted for obstacles and dynamically adjusted the paths as the robots traversed the environment.

During the experiment, the robots followed their respective paths, executing the position exchange while adhering to the computed paths. The movement of the robots was synchronized with the D* algorithm, allowing them to successfully navigate through the workspace and exchange their initial positions.

D. Data Collection

To gather comprehensive data on the behavior and dynamics of both the quadcopter and Pioneer robot, we conducted meticulous data collection during simulation runs. Two scenarios, namely the normal and abnormal scenarios, were employed to simulate different operational conditions.

In the normal scenario, both the quadcopter and Pioneer robot exhibited smooth and gradual movements within the simulated environment. They followed predefined paths and maintained stable position, orientation, velocity, acceleration, and angular velocity throughout the simulation. This scenario aimed to capture the baseline behavior of both robots.

In contrast, the anomalous scenario introduced sudden and unpredictable movements to mimic unexpected events or disturbances. Random position offsets and joint velocity fluctuations were applied to both the quadcopter and Pioneer robot, causing them to deviate from their intended paths. This scenario aimed to simulate challenging situations and assess the robustness of their control systems.

During each simulation run, we logged various data points for both the quadcopter and Pioneer robot, including the timestamp, position, orientation, velocity, acceleration, and angular velocity. This information was recorded at regular intervals throughout the simulation to capture the dynamic behavior of both robots. All the logged data for the quadcopter and Pioneer robot was saved to separate text files for further analysis and processing.

The acceleration profiles of the quadcopter and Pioneer robot were analyzed to better understand their behavior and dynamics in different scenarios. Figures 1a and 1b illustrate the relationship between acceleration and time for both normal and anomalous logs in Context 1 and Context 2, respectively.

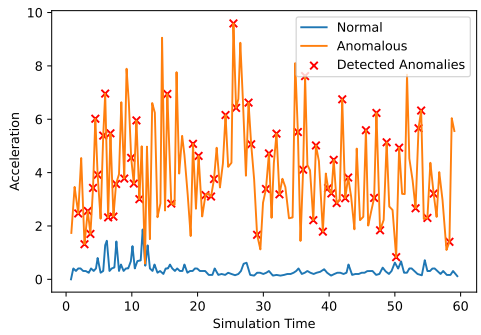
In Context 1, the normal scenario (Figure 1a) shows smooth and consistent acceleration for both robots throughout the simulation, reflecting stable and predictable movements within the simulated environment. In contrast, the anomalous scenario (Figure 1a) introduces sudden changes in acceleration for both robots, indicating the presence of unexpected events or disturbances. Notably, our anomaly detection system was able to correctly identify these points as anomalous.

Similarly, in Context 2, the acceleration profiles for normal and anomalous logs are shown in Figure 1b. The normal scenario (Figure 1b) displays consistent acceleration patterns, while the anomalous scenario (Figure 1b) exhibits abrupt changes in acceleration, indicating challenging situations that affect the robots' movements.

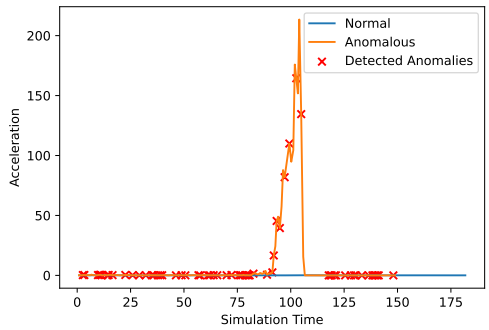
These acceleration profiles offer valuable insights into the dynamic behavior of the quadcopter and Pioneer robot under various operational conditions. They highlight the differences between normal and anomalous scenarios and allow for a thorough analysis of the control systems' robustness.

E. Results

In this section, we present the results of our classification experiments for Context 1 and Context 2. The performance of two different models, Logistic Regression (LR) and Support



(a) Acceleration vs Simulation Time for Context 1



(b) Acceleration vs Simulation Time for Context 2

Fig. 1: Comparison of Acceleration vs Simulation Time

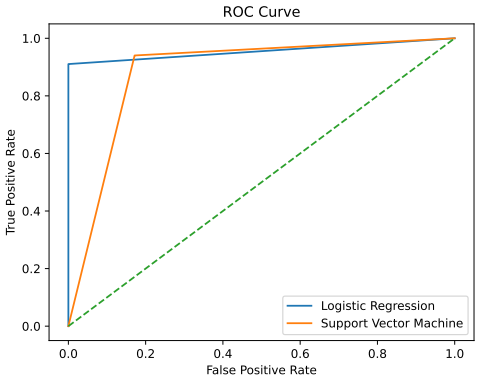
Vector Machine (SVM), was evaluated using various metrics, including ROC score, precision, recall, accuracy, and F1-score. Each reported result represents the average of 10 iterations to ensure robustness and reliability of the findings.

For Context 1, the LR model demonstrated exceptional performance across all evaluation metrics. It achieved an impressive ROC score of 0.9552 and high values for precision (0.9597), recall (0.9562), accuracy (0.9562), and F1-score (0.9561) (see Table I). These results suggest that LR effectively classified the data instances in Context 1. In contrast, the SVM model exhibited slightly lower performance in Context 1. It achieved an ROC score of 0.8844 and lower values for precision (0.8888), recall (0.8832), accuracy (0.8832), and F1-score (0.8830) compared to LR.

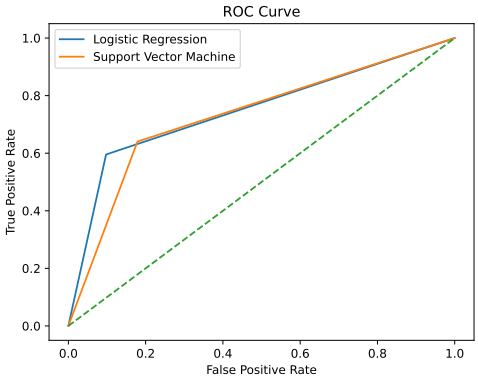
Moving to Context 2, we introduced an additional model, the Autoencoder. The results showed that the LR model achieved a ROC score of 0.7675 and demonstrated good performance across all evaluation metrics. It achieved a precision of 0.8488, recall of 0.8038, accuracy of 0.8038, and an F1-score of 0.7888. Similarly, the SVM model achieved a ROC score of 0.7308, indicating a relatively lower discriminative ability compared to LR. It exhibited precision, recall, accuracy, and F1-score values of 0.7435, 0.7453, 0.7453, and 0.7427, respectively, which were lower than those of LR. Interestingly, the Autoencoder model achieved the best results among the three models in Context 2. It achieved a ROC score of 0.7490, which is comparable to the LR model, and demonstrated good performance with precision, recall, accuracy, and F1-score values of 0.7804, 0.7736, 0.7736, and 0.7660, respectively (see Table II). These findings suggest that the Autoencoder model shows promise and can be considered as a favorable alternative to both LR and SVM, exhibiting the best overall performance in terms of classification metrics in Context 2.

To provide a visual representation of the model performance, Figure 1 presents the ROC curve for Context 1, showcasing the performance of the LR, SVM, and Autoencoder models. The plot demonstrates the true positive rate (sensitivity) against the false positive rate (1 - specificity) at various classification thresholds. The ROC curve allows us to compare and assess the discriminative ability of the different models.

Similarly, Figure 2 presents the ROC curve for Context 2, again featuring the LR, SVM, and Autoencoder models. This curve illustrates the performance of the models in terms of their true positive rate and false positive rate at different classification thresholds.



(a) ROC Curve for Context 1



(b) ROC Curve for Context 2

Fig. 2: Comparison of ROC Curves

VI. Conclusion Future Work

In conclusion, our study involved collecting system logs from different scenarios using Coppeliasim and developing a

TABLE I: Comparison of Classification Results for Context 1 (Average of 10 iterations)

Model	ROC Score	Precision	Recall	Accuracy	F1-score
LR	0.9552	0.9597	0.9562	0.9562	0.9561
SVM	0.8844	0.8888	0.8832	0.8832	0.8830

TABLE II: Comparison of Classification Results for Context 2 (Average of 10 iterations)

Model	ROC Score	Precision	Recall	Accuracy	F1-score
Autoencoder	0.7675	0.8488	0.8038	0.8038	0.7888
SVM	0.7308	0.7435	0.7453	0.7453	0.7427
LR	0.7490	0.7804	0.7736	0.7736	0.7660

machine learning-based anomaly detection system. We evaluated the performance of this system in two contexts using different models, including Logistic Regression (LR) and an Autoencoder. Our results showed that the LR model consistently demonstrated superior performance in both contexts, while the Autoencoder model showed promise as an alternative in Context 2.

In terms of future work, we plan to focus on handling more complex logs and exploring different models to enhance the anomaly detection capabilities of our system. Additionally, we aim to evaluate the performance of our system under diverse scenarios to further improve its effectiveness.

REFERENCES

[1] Ji, Tianchen, et al. "Proactive anomaly detection for robot navigation with multi-sensor fusion." IEEE Robotics and Automation Letters 7.2 (2022): 4975-4982.

[2] Olivato, Matteo, et al. "A comparative analysis on the use of autoencoders for robot security anomaly detection." 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019.

[3] Park, Daehyung, Yuuna Hoshi, and Charles C. Kemp. "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder." IEEE Robotics and Automation Letters 3.3 (2018): 1544-1551.

[4] Utkin, Lev V., V. S. Zaborovskii, and Sergey G. Popov. "Detection of anomalous behavior in a robot system based on deep learning elements." Automatic Control and Computer Sciences 50 (2016): 726-733.

[5] E. Rohmer and S. P. N. Singh and M. Freese, CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework, Proc. of The International Conference on Intelligent Robots and Systems (IROS), 2013, www.coppeliarobotics.com

[6] S. Huang, N. Cai, P. P. Pacheco, S. Narrandes, Y. Wang, W. Xu, Applications of support vector machine (svm) learning in cancer genomics, Cancer genomics & proteomics 15 (1)(2018) 41–51.