IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 4, 2020, accepted January 13, 2020, date of publication January 20, 2020, date of current version February 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2967629

Formal Verification of a Hybrid Machine Learning-Based Fault Prediction Model in Internet of Things Applications

ALIREZA SOURI[1], AMIN SALIH MOHAMMED[2,3], MOAYAD YOUSIF POTRUS[3], MAZHAR HUSSAIN MALIK[4], FATEMEH SAFARA[5], AND MEHDI HOSSEINZADEH[6]

[1]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran 1477893855, Iran
[2]Department of Computer Engineering, Lebanese French University, Erbil 44001, Iraq
[3]Department of Software and Informatics Engineering, Salahaddin University—Erbil, Erbil 44001, Iraq
[4]Department of Computing, Global College of Engineering and Technology—Muscat, Muscat 112, Oman
[5]Department of Computer Engineering, Islamshahr Branch, Islamic Azad University, Islamshahr 8975143465, Iran
[6]Health Management and Economics Research Center, Iran University of Medical Sciences, Tehran 1449614535, Iran

Corresponding author: Mehdi Hosseinzadeh (hosseinzadeh.m@iums.ac.ir)

This work derives from the Research Project with code 98-2-37-15609 and Approval ID IR.IUMS.REC.1398.861.

ABSTRACT By increasing the complexity of the Internet of Things (IoT) applications, fault prediction become an important challenge in interactions between human, and smart devices. Fault prediction is one of the key factors to achieve better arranging the IoT applications. Most of the current research studies evaluated the fault prediction methods using simulation environments. However, formal verification of the correctness of a fault prediction method has not been reported yet. This paper presents a behavioral modeling and formal verification of a hybrid machine learning-based fault prediction model with Multi-Layer Perceptron (MLP) and Particle Swarm Optimization (PSO) algorithms. In particular, the PSO is used for feature selection. Then, the fault prediction is considered as a behavior to be verified formally. The fault prediction behavior is divided into two types of behaviors: dimension reduction behavior and prediction behavior. For each of the behaviors, one formal model is designed. The behavioral models designed are mapped into the Labeled Transition System (LTS). The Process Analysis Toolkit (PAT) model checker is employed to evaluate the behavioral models. The accuracy of the fault prediction method is done by some existing specifications such as deadlock-free and reachability properties in terms of linear temporal logic formulas. Also, the verification of the fault prediction behaviors is used to detect the defect metrics of information-centric IoT applications. Experimental results showed that our proposed verification method has minimum verification time and memory usage for evaluating critical specification rules than other research studies.

INDEX TERMS Internet of Things applications, fault prediction, formal verification, process analysis toolkit, multi-layer perceptron, particle swarm optimization.

I. INTRODUCTION

The quality of Internet of Things (IoT) applications [1] has grown rapidly throughout recent years, and issues related to that have gained more importance for software developers [2], [3]. A serious step for changing the testing procedure is based on the capacity of assessing the fault prediction process, evaluating the degree of a product module and then testing [4], [5]. Fault prediction before testing helps the developer to eliminate costs and fault detection after testing provides feedbacks for procedures of maintenance [6], [7]. Programming deficiencies can't be straightforwardly measured at the programming phase [8]–[10]. Nonetheless, to find relations between quantifiable software properties and faults [11], [12] can help detecting faults.

Traditional methods, testing or simulation, cannot overcome the challenges posed [13]–[15] in fault prediction. Two important points in this regard are high cost and time overhead. In addition, reenactment is not suitable for supporting transient properties since all the possible states of the framework doesn't consider. To solve this problem formal method can be employed. Formal methods are based on mathematical logic. Generally, formal methods can be divided into formal specification and formal verification [16].

The associate editor coordinating the review of this manuscript and approving it for publication was Mu-Yen Chen.

Interaction behaviors between fault proneness and formal verification are called formal specifications [17], [18]. Most of the papers on fault prediction examine their proposed method through simulation and experiments. Another way to verifying an information-centric IoT application is model checking [19]–[21].

In this paper, fault behaviors are separated into two behaviors: dimension reduction behavior and fault prediction behavior which can be showed by behavioral models [22], [23]. The PSO is employed for dimension reduction and selection of an optimal solution. Then, the MLP is used to predict faults in IoT applications. To verify the proposed fault prediction method formally, logical problems are analyzed and behavioral specifications are checked. Contributions of this paper are as follow

- To propose a machine learning-based faults prediction model using the MLP and PSO algorithms in IoT applications.
- To present a behavior model to separate dimension reduction and prediction behaviors from each other.
- To enable the procedure of mapping two behaviors by formal verification approach based on Binary Decision Diagram (BDD).
- To guarantee and evaluate the logical problems such as deadlock-free, fairness, and reachability of the proposed fault prediction model.

The reminder of this paper is organized as follows. Section 2 is dedicated to related work conducted on fault prediction method and formal verification method. Section 3 provides the verification method on behavioral modeling, the approach proposed, and software metrics. Section 4 illustrates the formal verification and model checking process for the fault prediction method behaviors. Section 5 gives the model checking method to deal with some linear temporal logic specifications. Section 6 includes conclusions and future works.

Table 1 shows a list of abbreviations in this paper.

TABLE 1. List of abbreviations.

Acronym	Full text
ABC	Ant Bee Colony
RD	Reduced Dimensionality
LTS	Labeled Transition System
IoT	Internet of Things
MLP	Multi-Layer Perceptron
PSO	Particle Swarm Optimization
SVM	Support Vector Machine
ANN	Artificial Neural Network
BP	Back Propagation
KNN	K-Nearest Neighbor
ELM	Extreme Learning Method
GA	Genetic Algorithm
NB	Naive Bayes
BDD	Binary Decision Diagram
PAT	Process Analysis Toolkit
LTL	Linear Temporal Logic
PR	Prediction

II. RELATED WORK

In IoT based applications that are growing every day, finding the faults and assigning them to the appropriate developer to repair is very crucial. Three main directions followed in the previous studies [24] on IoT based applications are: 1) defining and specifying metrics to calculate complexity of software, 2) verifying correctness and validating thoroughness, 3) identifying and investigating models which try to predict the faults with regard to the software metrics defined [25], [26].

Fault prediction in software can be defined by software metrics, which give quantitative imageries of program qualities. Various reviews give clear proof that software metrics are related to fault-proneness [27]. A few techniques have been investigated to create prescient models of fault prediction in software. Statistical strategies [28] have been suggested as well. Much work has focused on finding the best way to choose the software metrics that will probably show the fault-proneness [29].

Some metrics have been presented to describe software quality in forms of static and dynamic platforms. In the static platform, features of code structure are measured as metrics [30]. Static measurements are a number of supervisors [31], [32] and a number of bunches [33]. Dynamic platforms measure testing perfectionism. Basic element measurements depend on auxiliary and information stream scope [6]. The connection between product measurements and blame inclination, and also numerous quantifiable programming characteristics has been exactly demonstrated by many researchers [22], [34]–[37].

Assurance of faults prediction modules is a vital procedure since it distinguishes modules that need itemized testing and reproduction [38]. Early fault prediction algorithms depend on measurements [39]. A huge part of recent works has focused on machine learning techniques [40] such as Support Vector Machine (SVM) [41], Naive Bayes algorithm (NB) [22], and Artificial Neural Network (ANN) [42]. Multi-Layer Perceptron (MLP) is a widely used machine learning algorithm with supervised learning. The Particle Swarm Optimization (PSO) algorithm is an evolutionary algorithm for optimization purposes. Rui *et al.* [43] proposed a method of fault detection in a power IoT equipment. They proposed a multi-spectral method to fuse images through deep learning. A convolutional neural network with deep learning is designed to detect faults in images of power devices. Their proposed method helps in locating the points of faults accurately and quickly.

In [44] a hybrid approach of the Extreme Learning Method (ELM) and Genetic Algorithm (GA) is proposed. With extracting useful information from the fault reports, a vector space model is constructed based on the information and a minimal feature set is selected. The features are fed into

an ensemble classifier that is a GA-based ELM training algorithm. Their proposed algorithm outperformed KNN, Naïve Bayes and SVM.

Another issue in IoT environments is occurring faults in the result of software aging. Liu and Meng [45] proposed a method to predict software aging. The method works based on a neural network with Back Propagation (BP) error. The weights and thresholds are determining using Artificial Bee Colony (ABC) algorithm. In other words, ABC is used to optimize the BP model. They showed that in compare with the standard BP neural network, their proposed method converges faster and predicts more accurate.

A part of research in the field of fault prediction in an IoT environment is conducted on the prediction of faults in industrial processes. As an instance, in [46] a cloud control architecture is proposed. Deep learning analysis is performed to detect faults in the manufacturing process. As another example, Xenakis *et al.* [47] proposed a fault detection method for an IoT environment. The method is introduced for industrial automation.

In [48] a fault prediction and diagnosis solution are presented. IoT enabling technology offered by SAP is exploited and a method is proposed to predict and detect a fault in an Industry's process of data collection. First, the device sensor data is analyzed without having any knowledge of the physical manufacturing system, and the causal relationship of the physical devices are discovered. Therefore, faults of certain devices can be predicted through monitoring of the healthy index of these devices in real-time. In addition, possible faults of other devices could be predicted using the casual relationship discovered in the steps before.

In [49], a fault prediction is proposed which works based on key mechanical equipment groups to improve the efficiency and intelligence level of fault prediction. A four-layer functional architecture is designed for comprehensive condition monitoring, reliable transmission, and intelligent information processing to predict system faults. Moreover, three canonical difficulties of the system are discussed including difficulties of combining fault prediction and IoT, difficulties of non-linearity of fault prediction, and difficulties of processing massive data.

According to the above related works, most of the researches evaluated their proposed method only by simulation experiments and statistical analysis. The main defect of these evaluations is that there is no complete and integrated dataset for showing all of the data metrics. So, analyzing the accuracy of an evolutionary approach in a fault prediction method is considered in this paper as an essential and important factor. Behavioral modeling for fault prediction which is based on ANN and PSO approaches are proposed and explained in the next section.

III. FAULT PREDICTION METHOD

In this section, the proposed hybrid of the MLP and PSO for fault prediction presented in [8] is formally verified. The behavior of the proposed fault prediction method is divided into two separate behaviors: dimension reduction behavior and fault prediction behavior. These two behaviors are modeled first, and then the behavioral models are verified.

A. DIMENSION REDUCTION AND FAULT PREDICTION

The PSO is used as a part of fault prediction method, the hybrid approach introduced in this paper for fault prediction. In the PSO, particles, are sailed through multi-dimensional space exploration. For each particle i there is a location path X_i. Particles move in a group in a d-dimensional problem space. V_i is the speed of each particle. Equations 1 and 2 illustrate the particle speed and location respectively [50]:

$$V_i(t+1) = w \cdot V_i(t) + c_1 r_1 (P_{i,best}(t) - X_i(t)) + c_2 r_2 (P_{global}(t) - X_i(t)) \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

where t is replication value, w is appropriate weight, c_1 and c_2 are positive coefficients, r_1 and r_2 are random numbers regularly distributed in the range [0, 1]. P_{ibest} and P_{global} are the best previously visited location of the particle i and the finest cost of all particle location values, respectively. The initial speeds of particles are limited to a range of [0, 1]. Equation 3 and 4 illustrate the moving of particles:

$$X_i(t+1) = p_i(t) + \alpha . |G_{i,best} - X_i(t)| ln 1/u_i(t), \quad if\ s \geq 0.5 \quad (3)$$

$$X_i(t+1) = p_i(t) + \alpha . |G_{i,best} - X_i(t)| ln 1/u_i(t), \quad if\ s < 0.5 \quad (4)$$

where u_i and s are random numbers uniformly distributed in the range of [0, 1], and parameter α is called contraction-expansion coefficient. The mean of best positions is shown by $G_{i,best}$ [6]. In the prediction phase, all input and preferred outputs of the MLP have been normalized in the range [0, 1]. Metrics selected by the PSO are fed into the MLP for iprediction.

B. BEHAVIORAL MODELING OF THE FAULT PREDICTION APPROACH

We illustrate the fault prediction approach based on [8]. The reduced dimensionality is performed using the PSO method. Training and testing methods are applied using the MLP mechanism. Figure 1 illustrates the reduced dimensionality behavior. First, the specified metrics are inputted to the mechanism. Reduced Dimensionality (RD) behavior normalizes the metrics based on the data refinement method. Population initialized, X_i checked and V_i computed. Greatest beforehand stayed position of the particle i is considered. If $P_{i,best} > max$, the mean of the best positions of all particles calculated and then, $G_{i,best}$ is updated. If $G_{i,best} < max$, $G_{i,best}$ is selected and the population is reduced. Finally, the minimum metrics are reached.

Figure 2 describes the Prediction (PR) behavior. First, in the minimized metrics, data are divided into test data and

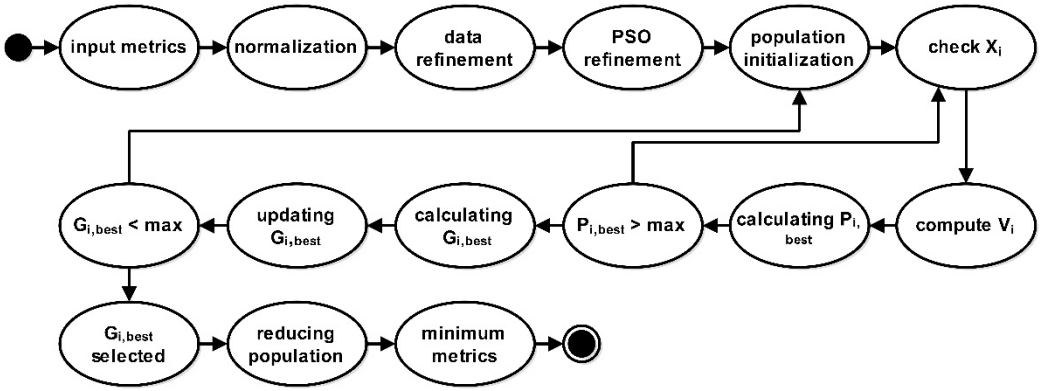

FIGURE 1. The reduced dimensionality behavior.

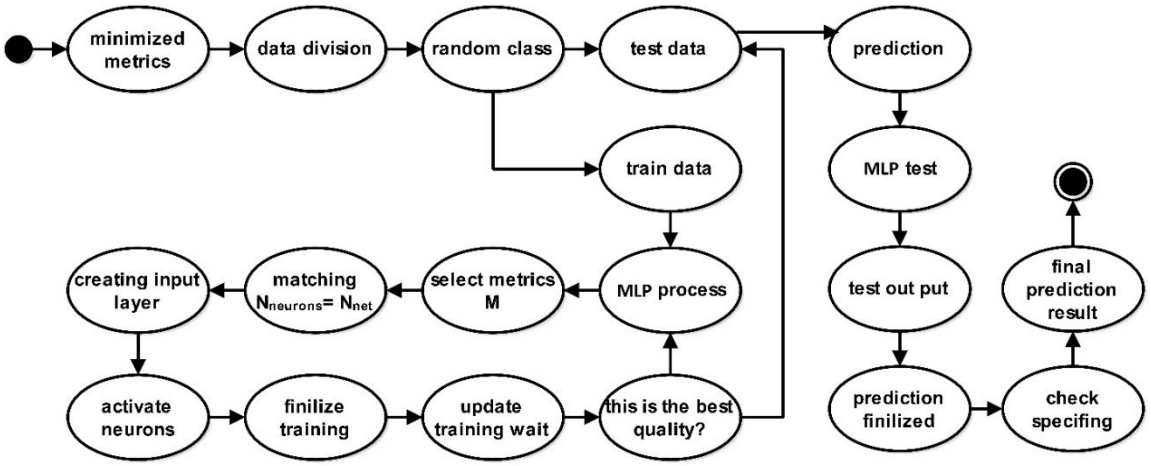

FIGURE 2. The prediction behavior.

train data. The MLP process is done in the train data, metrics M is selected and $N_{neurons} = N_{net}$ is matched. By creating the input layer, neurons become active and training data are finished. If the training procedure reaches the best quality, the test data will begin. By finishing the test process, specifications are checked and the final prediction results are reached.

With respect to the presentation of the RD and PR mechanisms, these behavioral models are mapped into the Labeled Transition Systeme (LTS).

Definition 1: An LTS of $\mathbb{FP}$ is a multi-tuple $\mathbb{LT} = (\mathbb{S}, s, \mathbb{A}, \mathbb{R})$ where [51]:

- $\mathbb{S}$ shows a set of existing states on the RD and PR mechanisms.
- s depicts the the initial state of each mechanism, $s \in \mathbb{S}$.
- $\mathbb{A}$ presents a set of actions on the RD and PR mechanisms.
- $\mathbb{R}$ illustrates a set of transition relations on the RD and PR mechanisms where $\mathbb{R} \subseteq \mathbb{S} \times \mathbb{A} \times \mathbb{S}$ that shows the relation $s_1 \xrightarrow{a} s_2 (s_1, s_2 \in \mathbb{S}$ and $a \in \mathbb{A})$ is applied for transition relaion $(s_1, a, s_2) \in \mathbb{T}$.

According to fault prediction behavioral models, a set of states and events are as follow:

RD_States=(Begin, Input_metrics, Normalization, Data_refi, PSO_refi, Pop_init, Initializing, Check_x, Comput_v, Calc_Pbest, Pbest_B_max, Calc_Mbest, Update_M, Mbest_L_max, Mbest_selec, Redu_Pop, Min_metrics);

PR_States = (start, Mini_metrics, Data_divi, Rand_class, Train, Test, MLP_proc, Select_metr_M, Matching_N, Creat_layer, Activ_neur, Finaliz_train, Update_train, Best_Q, MLP_test, Output, Pred_finalized, Check_spec, Final_result);

A path on the $\mathbb{RD}$ and $\mathbb{PR}$ behavioral models of the fault prediction mechanism is defined as follows:

Definition 2: A directed path ***DP*** is a set of the restricted states and actions informs of transition relations with initial state s_i and final state s_j (s_i and $s_j \in \mathbb{S}$) that is shown an example as follows:

$$\textbf{\textit{DP (1)}} = Start \xrightarrow{send}, Mini\_metrics \xrightarrow{check} Data\_divi \xrightarrow{seperation} Rand\_class$$

is a directed path in the PR model.

IV. MODEL CHECKING APPROACH FOR THE FAULT PREDICTION BEHAVIORS

This section presents a model checking approach to the proposed fault prediction behaviors. First, the Linear Temporal Logic (LTL) properties are defined as a temporal logic

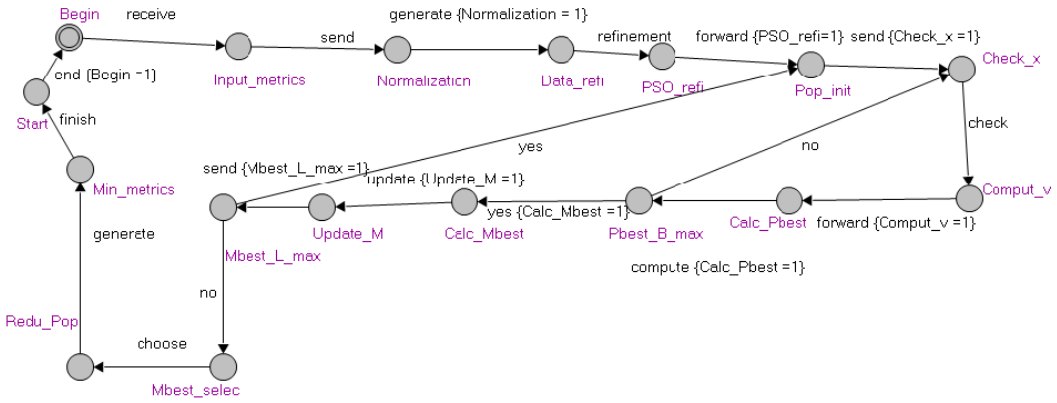

FIGURE 3. The RD model in the PAT environment.

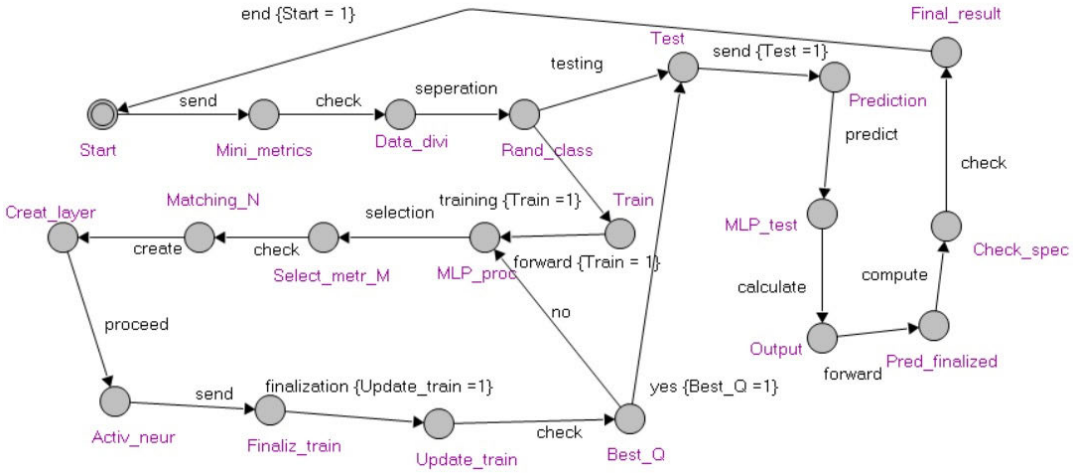

FIGURE 4. The PR model in the PAT environment.

language for the system behaviors in this section. Then, the proposed fault prediction behaviors are modeled using the LTS in the PAT[1] model checker that includes reduced dimensionality and prediction behaviors.

For showing the correctness of the proposed fault prediction behaviors, some critical properties are defined as informs of the LTL rules. We define LTL formulas to verify in the model checker as follow:

$$\psi ::= \text{True}|P|\neg\psi|\psi \vee \psi'|\psi \wedge \psi'|G\psi|X\psi|F\psi|\psi \text{U} \psi'$$

Let $\beta = s_0, s_1, s_2 \ldots$ as a set of states and ψ as an LTL property. Existing notions and operands of LTL are presented as follows [36]–[38]:

- $\beta \models \top$and $\beta \not\models \bot$;
- $\beta \models \psi_1 \vee \ldots \vee \psi_n$ IFF for some j = 1 …n we have $\beta \models \psi$j;
- $\beta \models \neg\psi$ IFF $\beta \not\models \psi$;
- $\beta \models X\psi$ IFF β 1 $\models \psi$;
- $\beta \models \psi$ U ψ' IFF for some $k = 0, 1 \ldots$ we have $p_k \models \psi'$ and $p_0 \models \psi \ldots p_{k-1} \models \psi$;

[1]http://patroot.com/

The timed operators in the LTL are shown as follows:

- X notation depicts "Next" timed operand: If $\beta \in$ LTL (property), then X (β) $\in$ LTL (property).
- G notation shows "Globally" timed operator: If $\beta \in$ LTL (property), then G (β) $\in$ LTL (property).
- F notation shows "Future" timed operator: If $\alpha \in$ LTL (property), then F (β) $\in$ LTL (property).
- U notation illustrates "Until" timed operator: If $\alpha, \beta \in$ LTL (property), then α U $\beta \in$ LTL (property).

In the PAT environment, all of the states are displayed as nodes. Each action navigates a transition relation between two states. If an action has a non-deterministic condition, then a guard will perform for this action.

Figure 3 illustrates the RD behavioral model of the LTS in the PAT environment. There are 17 states in this diagram, which are connected to each other by edges. This shows the modeling of reducing dimensionality. For example, in order to transfer producing input metrics state to refinement state, the normalization action must be acted.

Figure 4 shows the PR behavioral model of the LTS in the PAT environment. There are 20 states in this diagram. In this model train and test are demodulated. Figure 5 shows the state

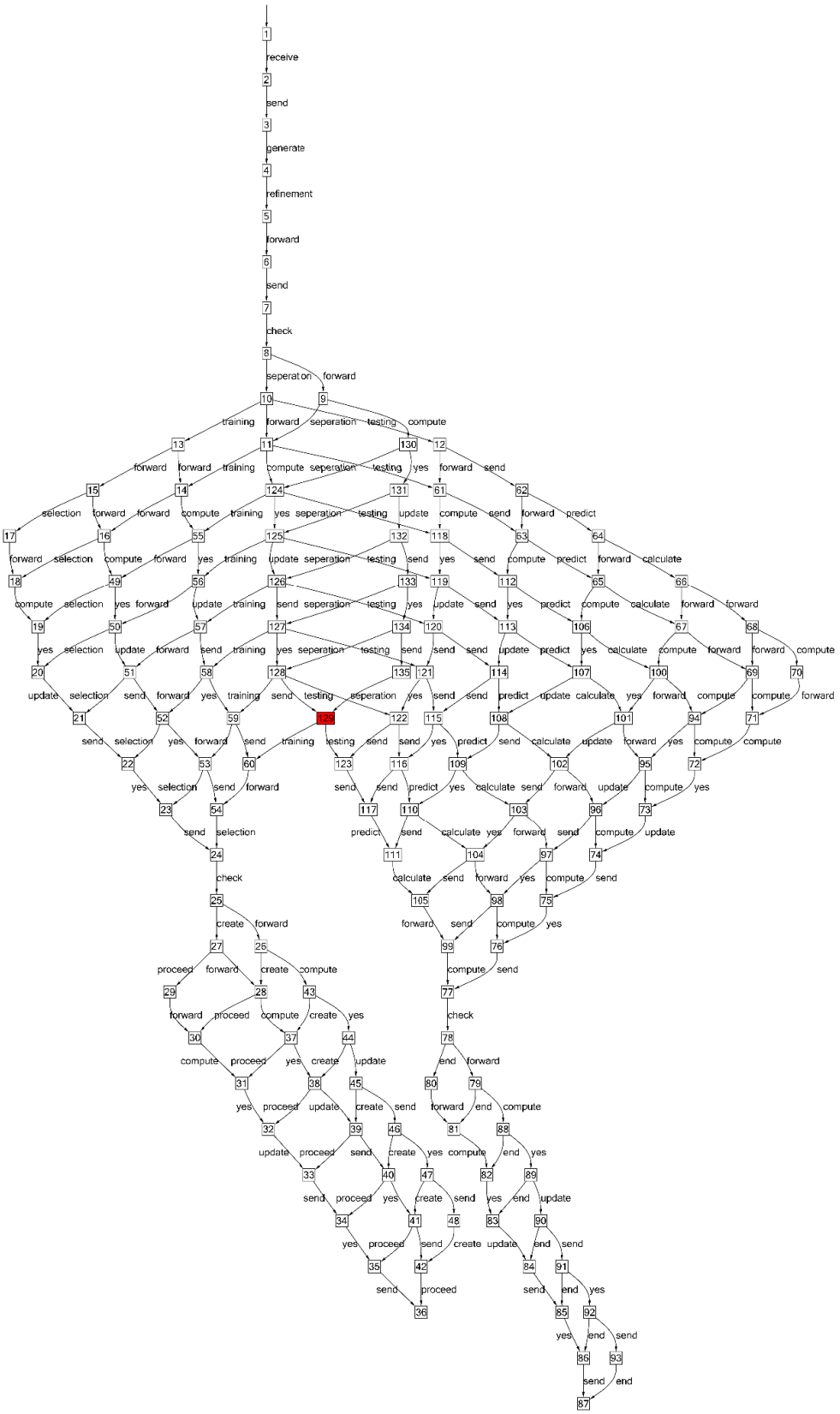

FIGURE 5. The state space of the LTS model in the simulation phase.

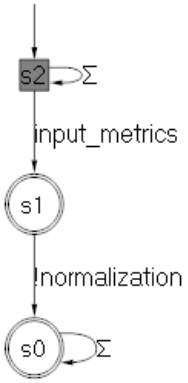

FIGURE 6. Fullfilment of L1 property in the state machine.

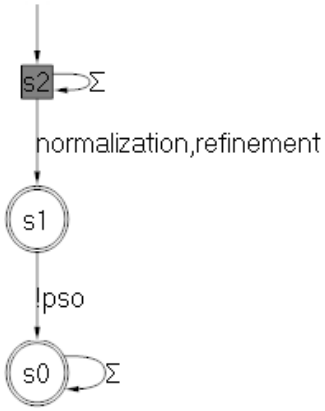

FIGURE 7. The satisfaction of L2 property in the state machine.

space of the RD and PR behaviors in the PAT environment. After generating this state space, the expected specification rules will examine to rule's satisfaction.

After describing the LTL structure, some critical properties are defined as informs of the LTL rules for the fault prediction model. Let -> as the consistency association between states in each specification:

- L1 **G**(`input_metrics` -> **X** `normalization`);

✓ Globally, after input data metrics, the normalization phase is coming.

Figure 6 shows the L1 property with state satisfaction conditions.

- L2 **G**(`normalization` **&&** `refinement` -> **X** pso);

FIGURE 8. The satisfaction of L3 property in the state machine.

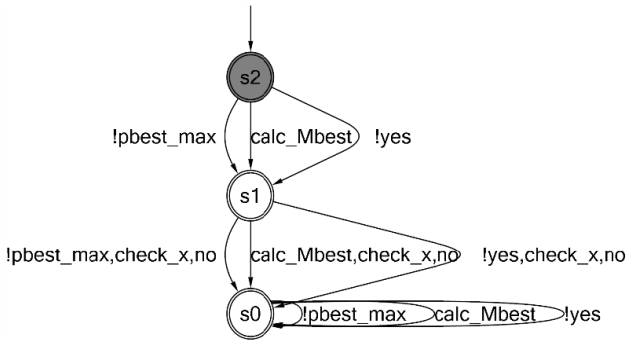

FIGURE 9. The satisfaction of L4 property in the state machine.

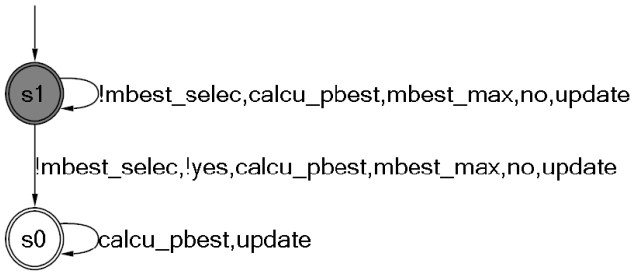

FIGURE 10. The satisfaction of L5 property in state machine.

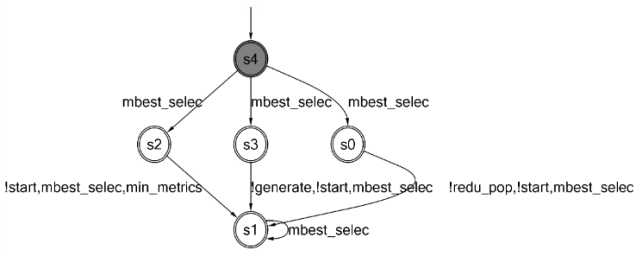

FIGURE 11. The satisfaction of L6 property.

✓ Globally, when the normalization phase and data refinement are checked, the next stage is the PSO phase. Figure 7 displays the L2 property with state satisfaction conditions.

- L3 **G**(`pop` **&&** `send` -> `check`) -> **F**(`comput_v` **&&** `forward` -> `compute` **&&** `calcu_pbest`);

✓ Globally, when initial population is done and the data is checked, the V value is checked and computed for

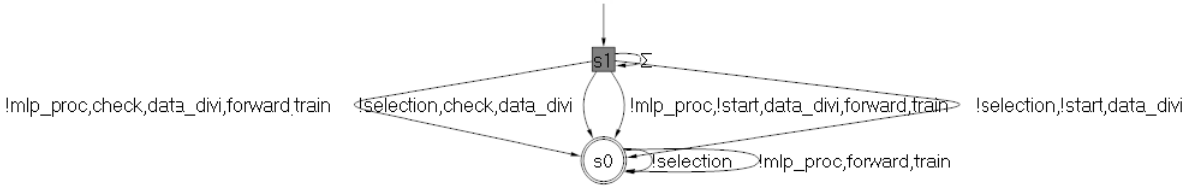

FIGURE 12. The satisfaction of L7 property in the state machine.

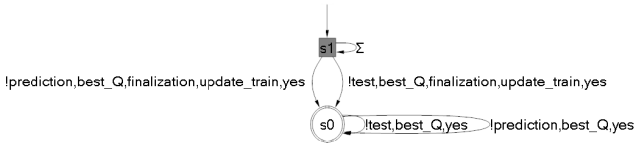

FIGURE 13. The satisfaction of L8 property in the state machine.

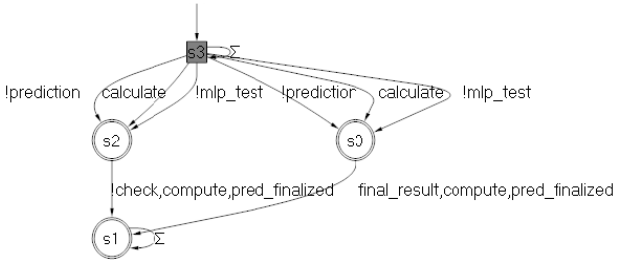

FIGURE 14. The satisfaction of L9 property in the state machine.

P_{best}. Figure 8 displays the L3 property with state satisfaction condition.

- L4 **G**(`pbest_max` **&&** `yes` **->** `calc_Gbest`) **->** **X** **!**(`check_x` **&&** `no`);

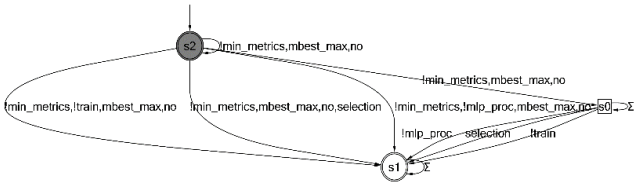

FIGURE 15. The satisfaction of L10 property in the state machine.

✓ Globally, when P_{best} is higher than max condition and G_{best} is calculated, in the next state checking X value should not occur concurrently. Figure 9 displays the L4 property with state satisfaction condition.

- L5 **G**(`calcu_pbest` **&&** `update`)**->**(`gbest_max` **&&** `yes`) **U**(`gbest_max` **&&** `no` **->** `gbest_selec`));

✓ Globally, when the P_{best} value is updated and the G_{best} is higher than max condition, finally G_{best} is selected. Figure 10 displays the L5 property with state satisfaction condition.

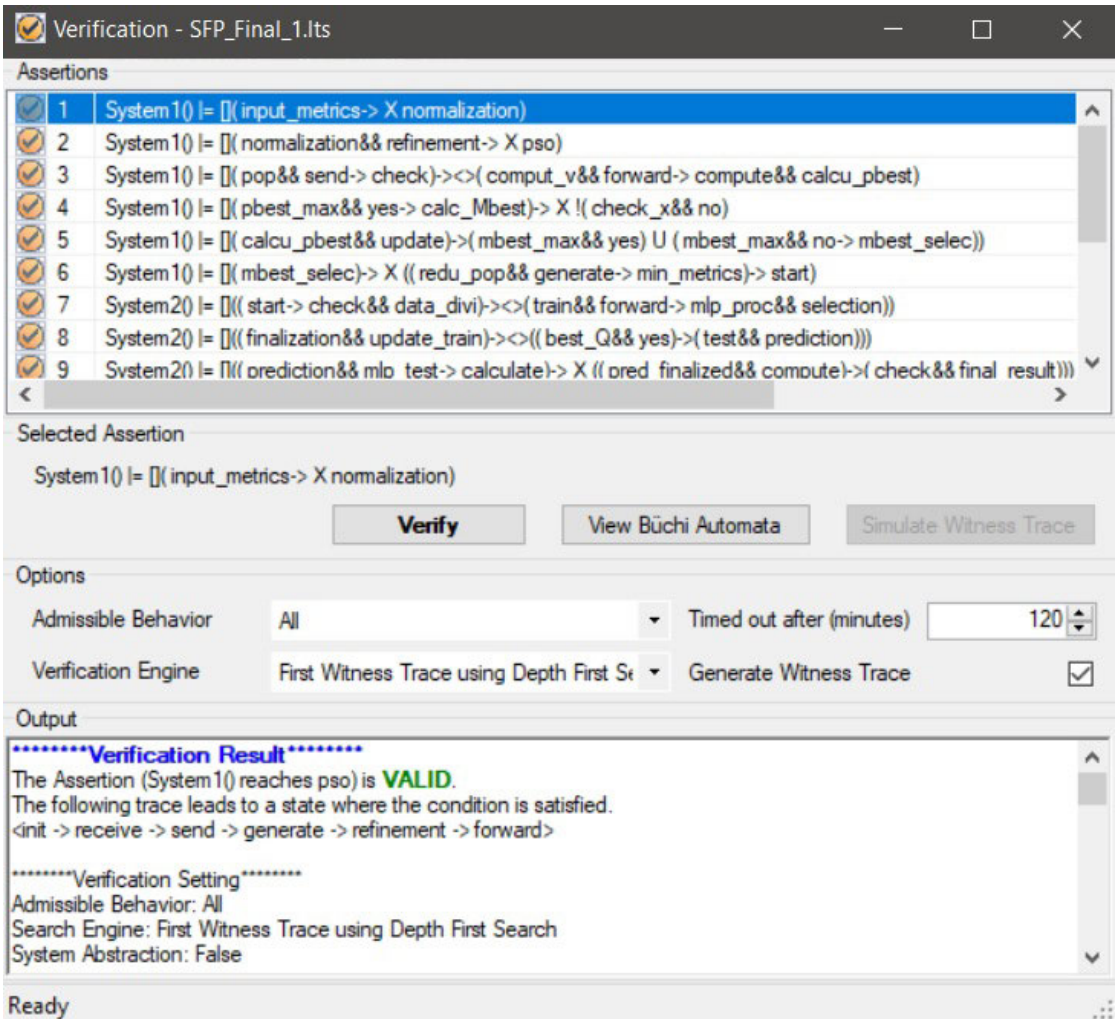

FIGURE 16. The automated verification analysis of the fault prediction approach.

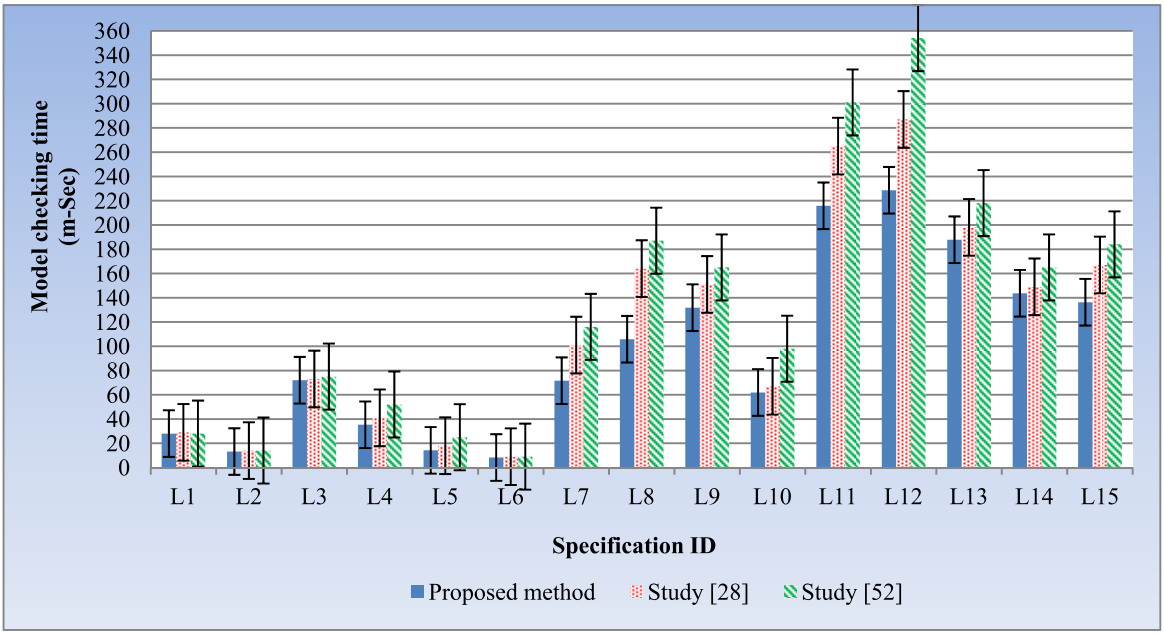

FIGURE 17. The model checking time for existing critical specifications in the PAT.

- L6 **G**(gbest_selec) -> **X** ((redu_pop **&&** generate -> min_metrics) -> start);

✓ Globally, when the G_{best} is selected, the next state is reducing population and generating the minimum metrics. Figure 11 displays the L6 property with state satisfaction condition.

- L7 **G**((start -> check **&&** data_divi)-> **F**(train **&&** forward->mlp_proc **&&** selection));

Globally, when the prediction phase is started, the data division is specified for training and testing operations. Figure 12 displays the L7 property with state satisfaction condition

- L8 **G**((finalization **&&** update_train) -> **F**((best_Q **&&** yes) -> (test **&&** prediction)));

✓ Globally, the finalization state and update training are done, eventually if best quality is reachable then the test operation is started. Figure 13 displays the L8 property with state satisfaction condition.

- L9 **G**((prediction **&&** ann_test -> calculate) -> **X** ((pred_finalized **&&** compute) -> (check **&&** final_result)));

✓ Globally, the prediction using the MLP is calculated, the next state is finalized using train phase; after that, the final result is sent to checking data metrics. Figure 14 displays the L9 property with state satisfaction condition.

- L10 **G** **!**(train **&&** mlp_proc -> selection) **U** (gbest_max **&&** no -> min_metrics);

✓ Globally, the train operation using the MLP selection has not performed until the G_{best} generate minimum metrics. Figure 15 displays the L10 property with state satisfaction condition.

- L11 RD deadlock-free;

✓ The RD system is deadlock- free in everywhereto recognize reducing dimensionality method based on the PSO.

- L12 PR deadlock-free;

✓ The PR system is deadlock -free in everywhere to predict existing faults in the system.

- L13 reaches the train;

✓ The train state is potentially reachable always in the generated state space.

- L14 reaches test;

✓ The test state is potentially reachable always in the generated state space.

- L15 reaches PSO;

✓ The PSO state is potentially reachable always in the generated state space.

V. EXPERIMENTAL RESULTS

This section shows the experimental results on the formal verification of the fault prediction behaviors that are performed by an Intel Core i5, 2.6 GHz, 8GB RAM, Windows 10 system and by PAT model checker 3.4.1. There are some advantages to compare with the proposed approach and other studies. Initially, the presented approach uses the MLP and PSO methods for decreasing the dimensionality phase and prediction approach. The Second advantage is using the PAT model checker for proving the accuracy of system behavior. The PAT model checker has some advantages such as graphical environment, graphical state space generation and simple development language [13].

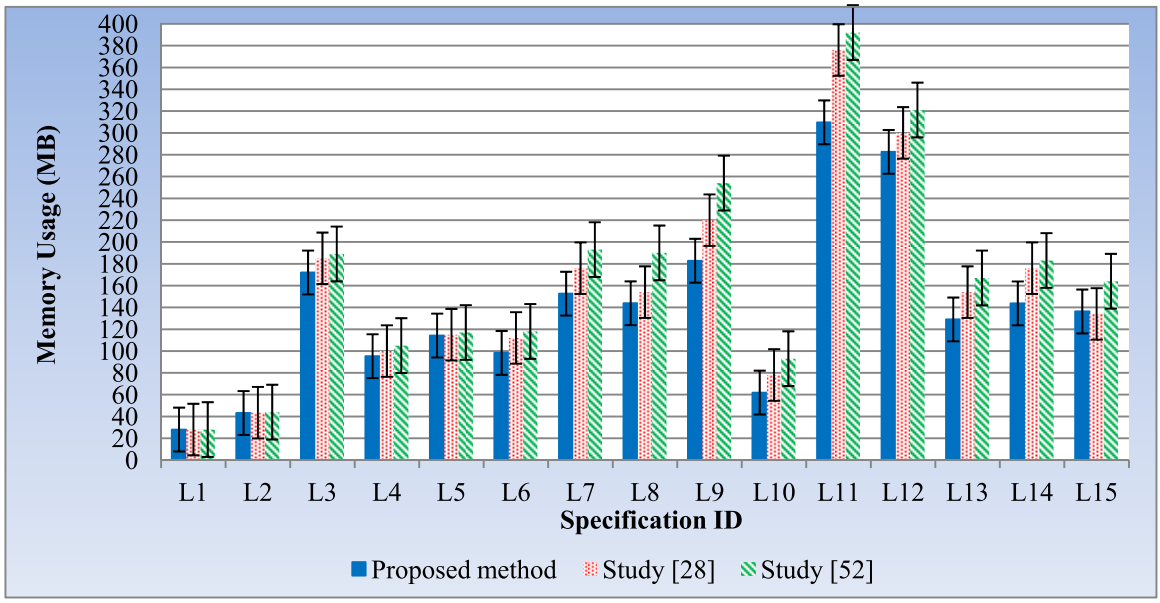

FIGURE 18. The memory usage for existing specifications in the PAT.

To evaluate existing specification rules for the proposed fault prediction approach, Figure 16 shows the automated verification analysis in the PAT tool. Based on verification results, all properties have been satisfied according to state space exploration. Also, the proposed method is reachable and deadlock free on all aspects of the proposed approach.

To evaluate the proposed verification method on the fault prediction approach with other prediction approaches, we analyze our method with [52] and [28] research studies. These studies have modeled using the LTS method that is compared with our proposed verification method. In [28], a hybrid approach using ANN and SVM algorithm was applied to detect fault diagnosis in smart devices. Also, in [52], the authors have applied an evolutionary algorithm to evaluate failure prediction in smart evolving systems.

Figure 17 illustrates the model checking time for each LTL specification using the PAT tool. The maximum time belongs to the L12 specification ID by 230 millisecond that checks the deadlock-free condition for the proposed PR model. The proposed verification method has a minimum model checking time for evaluating each LTL specification than other prediction approaches.

Also, Figure 18 demonstrates the memory usage to verify the specifications of the LTS model. The maximum memory consumption belongs to the L11 specification ID by 310 MB that checks the deadlock-free condition for the proposed RD model. According to experimental results, the proposed verification method for fault prediction approach has minimum memory usage for verifying each LTL specification than other research studies. Based on verification results for evaluating the correctness of the proposed fault prediction approach, our verification method supports minimum model checking time and memory usage for critical LTL specification rules. To evaluate the existing critical specification rules, the correctness of the proposed fault prediction approach is proven based on the LTS method. Also, all functional properties as the LTL formulas on the proposed fault prediction approach have been guaranteed.

VI. CONCLUSION AND FUTURE WORKS

This paper presented behavioral modeling on fault prediction approach for IoT applications using MLP and PSO algorithms. Fault detection behaviors are separated into two types: reducing dimensionality behavior and prediction behavior. The interactions between the two behaviors are shaped using the LTS method. Analyzing logical problems and checking behavioral specifications are the procedures to verify the proposed fault prediction approach. The PAT model checker is used to perform the behavioral models. The accuracy of the fault prediction approach was proved according to the deadlock-free and reachability properties in terms of linear temporal logic formulas. The maximum time belongs to the L12 specification ID by 229 milliseconds that checks the deadlock-free condition for the proposed PR model. Also, the maximum memory consumption belongs to the L11 specification ID by 310 MB that checks the deadlock-free condition for the proposed RD model. Finally, the model checking time and memory usage for formal verification of the proposed fault prediction approach is lower than existing research studies. In future work, a hybrid analysis of fault prediction using meta-heuristic algorithms will be researched using verification approaches. Also, other functional properties of fault prediction approach such as completeness, soundness, and fairness are analyzed in future research.

ACKNOWLEDGMENT

This article derives from the Research Project with code 98-2-37-15609 and Approval ID IR.IUMS.REC.1398.861.

REFERENCES

[1] A. Souri, A. Hussien, M. Hoseyninezhad, and M. Norouzi, "A systematic review of IoT communication strategies for an efficient smart environment," *Trans. Emerg. Telecommun. Technol.*, Aug. 2019, Art. no. e3736.

[2] R. Mahajan, S. K. Gupta, and R. K. Bedi, "Design of software fault prediction model using BR Technique," *Procedia Comput. Sci.*, vol. 46, pp. 849–858, Jan. 2015.

[3] I. Umesh and G. N. Srinivasan, "Dynamic software aging detection-based fault tolerant software rejuvenation model for virtualized environment," in *Proc. Int. Conf. Data Eng. Commun. Technol.*, 2017, pp. 779–787.

[4] Y. Abdi, S. Parsa, and Y. Seyfari, "A hybrid one-class rule learning approach based on swarm intelligence for software fault prediction," *Innov. Syst. Softw. Eng*, vol. 11, no. 4, pp. 289–301, Dec. 2015.

[5] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, "Reliability and high availability in cloud computing environments: A reference roadmap," *Hum.-Centric Comput. Inf. Sci.*, vol. 8, no. 1, p. 20, Dec. 2018.

[6] I. Alsmadi and H. Najadat, "Evaluating the change of software fault behavior with dataset attributes based on categorical correlation," *Adv. Eng. Softw.*, vol. 42, no. 8, pp. 535–546, Aug. 2011.

[7] S. Chatterjee and A. Roy, "Web software fault prediction under fuzzy environment using MODULO-M multivariate overlapping fuzzy clustering algorithm and newly proposed revised prediction algorithm," *Appl. Soft Comput.*, vol. 22, pp. 372–396, Sep. 2014.

[8] C. Jin and S.-W. Jin, "Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization," *Appl. Soft Comput.*, vol. 35, pp. 717–725, Oct. 2015.

[9] P. García Nieto, E. García-Gonzalo, F. S. Lasheras, and F. De Cos Juez, "Hybrid PSO–SVM-based method for forecasting of the remaining useful life for aircraft engines and evaluation of its reliability," *Rel. Eng. Syst. Saf.*, vol. 138, pp. 219–231, Jun. 2015.

[10] V. Balasubramanian, F. Zaman, M. Aloqaily, I. A. Ridhawi, Y. Jararweh, and H. B. Salameh, "A mobility management architecture for seamless delivery of 5G-IoT services," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[11] Ö. Arar and K. Ayan, "Deriving thresholds of software metrics to predict faults on open source software: Replicated case studies," *Expert Syst. Appl.*, vol. 61, pp. 106–121, Nov. 2016.

[12] J. Hryszko and L. Madeyski, "Assessment of the software defect prediction cost effectiveness in an industrial project," in *Software Engineering: Challenges and Solutions*. Cham, Switzerland: Springer, 2017, pp. 77–90.

[13] I. Kaur, G. S. Narula, and V. Jain, "Differential analysis of token metric and object oriented metrics for fault prediction," *Int. J. Inf. Technol.*, vol. 9, no. 1, pp. 93–100, Mar. 2017.

[14] S. Zafar, S. Jangsher, O. Bouachir, M. Aloqaily, and J. Ben Othman, "QoS enhancement with deep learning-based interference prediction in mobile IoT," *Comput. Commun.*, vol. 148, pp. 86–97, Dec. 2019.

[15] Y. Kotb, I. Al Ridhawi, M. Aloqaily, T. Baker, Y. Jararweh, and H. Tawfik, "Cloud-based multi-agent cooperation for IoT devices using workflow-nets," *J. Grid Comput.*, vol. 17, no. 4, pp. 625–650, Dec. 2019.

[16] X. Wu and H. Zhu, "Formalization and analysis of the REST architecture from the process algebra perspective," *Future Gener. Comput. Syst.*, vol. 56, pp. 153–168, Mar. 2016.

[17] G. Denaro, S. Morasca, and M. Pezzé, "Deriving models of software fault-proneness," presented at the 14th Int. Conf. Softw. Eng. Knowl. Eng., Ischia, Italy, 2002.

[18] A. Souri, "Formal specification and verification of a data replication approach in distributed systems," *Int. J. Next-Gener. Comput.*, vol. 7, no. 1, pp. 1–21, 2016.

[19] C. Baier and J.-P. Katoen, *Principles of Model Checking* (Representation and Mind Series). Cambridge, MA, USA: MIT Press, 2008.

[20] S. Razzaghzadeh, A. H. Navin, A. M. Rahmani, and M. Hosseinzadeh, "Probabilistic modeling to achieve load balancing in expert clouds," *Ad Hoc Netw.*, vol. 59, pp. 12–23, May 2017.

[21] I. Al Ridhawi, M. Aloqaily, Y. Kotb, Y. Jararweh, and T. Baker, "A profitable and energy-efficient cooperative fog solution for IoT services," *IEEE Trans. Ind. Inf.*, to be published.

[22] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7346–7354, May 2009.

[23] Z.-W. Zhang, X.-Y. Jing, and T.-J. Wang, "Label propagation based semi-supervised learning for software defect prediction," *Autom. Softw. Eng.*, vol. 24, no. 1, pp. 47–69, Mar. 2017.

[24] P. Singh, N. R. Pal, S. Verma, and O. P. Vyas, "Fuzzy rule-based approach for software fault prediction," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 5, pp. 826–837, May 2017.

[25] E. Erturk and E. A. Sezer, "A comparison of some soft computing methods for software fault prediction," *Expert Syst. Appl.*, vol. 42, no. 4, pp. 1872–1879, Mar. 2015.

[26] L. Kumar, S. Misra, and S. K. Rath, "An empirical analysis of the effectiveness of software metrics and fault prediction model for identifying faulty classes," *Comput. Standards Interface*, vol. 53, pp. 1–32, Aug. 2017.

[27] H. R. Sabohi and H. Iranmanesh, "Optimal nonlinear observer with PSO approach in chaotic systems based on synchronization," presented at the 11th Int. Conf. Appl. Elect. Comput. Eng., Athens, Greece, 2012.

[28] A. Azadeh, M. Saberi, A. Kazem, V. Ebrahimipour, A. Nourmohammadzadeh, and Z. Saberi, "A flexible algorithm for fault diagnosis in a centrifugal pump with corrupted data and noise based on ANN and support vector machine with hyper-parameters optimization," *Appl. Soft Comput.*, vol. 13, no. 3, pp. 1478–1485, Mar. 2013.

[29] S. Otoum, B. Kantarci, and H. T. Mouftah, "Mitigating false negative intruder decisions in WSN-based smart grid monitoring," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2017, pp. 153–158.

[30] B. Liu, S. Nejati, and L. C. Briand, "Improving fault localization for Simulink models using search-based testing and prediction models," in *Proc. IEEE 24th Int. Conf. Softw. Anal., Evol. Reeng. (SANER)*, Feb. 2017, pp. 359–370.

[31] P. Deng, G. Ren, W. Yuan, F. Chen, and Q. Hua, "An integrated framework of formal methods for interaction behaviors among industrial equipments," *Microprocessors Microsyst.*, vol. 39, no. 8, pp. 1296–1304, Nov. 2015.

[32] Y. Chen, Z. Zhen, H. Yu, and J. Xu, "Application of fault tree analysis and fuzzy neural networks to fault diagnosis in the Internet of Things (IoT) for Aquaculture," *Sensors*, vol. 17, no. 12, p. 153, Jan. 2017.

[33] I. Arora, V. Tetarwal, and A. Saha, "Open issues in software defect prediction," *Procedia Comput. Sci.*, vol. 46, pp. 906–912, Mar. 2015.

[34] F. Liu and Z. Zhou, "An improved QPSO algorithm and its application in the high-dimensional complex problems," *Chemometrics Intell. Lab. Syst.*, vol. 132, pp. 82–90, Mar. 2014.

[35] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Inf. Sci.*, vol. 264, pp. 260–278, Apr. 2014.

[36] Y. Wu and R. Yang, "Software reliability modeling based on SVM and virtual sample," in *Proc. Annu. Rel. Maintainability Symp. (RAMS)*, Jan. 2013, pp. 1–6.

[37] G. Mauša and T. G. Grbac, "Co-evolutionary multi-population genetic programming for classification in software defect prediction: An empirical case study," *Appl. Soft Comput.*, vol. 55, pp. 331–351, Jun. 2017.

[38] D. Bowes, T. Hall, and J. Petrić, "Software defect prediction: Do different classifiers find the same defects?" *Softw. Qual. J.*, vol. 26, no. 2, pp. 525–552, Jun. 2018.

[39] A. Monden, J. Keung, S. Morisaki, Y. Kamei, and K.-I. Matsumoto, "A heuristic rule reduction approach to software fault-proneness prediction," in *Proc. 19th Asia–Pacific Softw. Eng. Conf.*, Dec. 2012, pp. 838–847.

[40] M. Norouzi, A. Souri, and M. Samad Zamini, "A data mining classification approach for behavioral malware detection," *J. Comput. Netw. Commun.*, vol. 2016, pp. 1–9, Mar. 2016.

[41] C. Catal, U. Sevim, and B. Diri, "Practical development of an eclipse-based software fault prediction tool using naive Bayes algorithm," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2347–2353, Mar. 2011.

[42] R. Malhotra, A. Kaur, and Y. Singh, "Empirical validation of object-oriented metrics for predicting fault proneness at different severity levels using support vector machines," *Int. J. Syst. Assur. Eng. Manag.*, vol. 1, no. 3, pp. 269–281, Sep. 2010.

[43] H. Rui, Z. Yunhao, T. Shiming, Y. Yang, and Y. Wenhai, "Fault point detection of IOT using multi-spectral image fusion based on deep learning," *J. Vis. Commun. Image Represent.*, vol. 64, Oct. 2019, Art. no. 102600.

[44] Y. Yin, X. Dong, and T. Xu, "Rapid and efficient bug assignment using ELM for IOT software," *IEEE Access*, vol. 6, pp. 52713–52724, 2018.

[45] J. Liu and L. Meng, "Integrating artificial bee colony algorithm and BP neural network for software aging prediction in IoT environment," *IEEE Access*, vol. 7, pp. 32941–32948, 2019.

[46] H. Lee, "Framework and development of fault detection classification using IoT device and cloud environment," *J. Manuf. Syst.*, vol. 43, pp. 257–270, Apr. 2017.

[47] A. Xenakis, A. Karageorgos, E. Lallas, A. E. Chis, and H. González-Vélez, "Towards distributed IoT/cloud based fault detection and maintenance in industrial automation," *Procedia Comput. Sci.*, vol. 151, pp. 683–690, Jan. 2019.

[48] C. Wang, H. T. Vo, and P. Ni, "An IoT application for fault diagnosis and prediction," in *Proc. IEEE Int. Conf. Data Sci. Data Intensive Syst.*, Dec. 2015, pp. 726–731.
[49] X. Xu, T. Chen, and M. Minami, "Intelligent fault prediction system based on Internet of Things," *Comput. Math. Appl.*, vol. 64, no. 5, pp. 833–839, Sep. 2012.
[50] B. Luitel and G. K. Venayagamoorthy, "Particle swarm optimization with quantum infusion for the design of digital filters," in *Proc. IEEE Swarm Intell. Symp.*, Sep. 2008, pp. 1–8.
[51] A. Souri, A. M. Rahmani, N. J. Navimipour, and R. Rezaei, "A symbolic model checking approach in formal verification of distributed systems," *Hum.-Centric Comput. Inf. Sci.*, vol. 9, p. 4, Dec. 2019.
[52] F. De Angelis, M. R. Di Berardini, H. Muccini, and A. Polini, "CASSANDRA: An online failure prediction strategy for dynamically evolving systems," in *Formal Methods and Software Engineering*. Cham, Switzerland: Springer, 2014, pp. 107–122.

ALIREZA SOURI received the B.S. degree in software engineering from the University College of Nabi Akram, Iran, and the M.Sc. and Ph.D. degrees in computer engineering from the Science and Research Branch, Islamic Azad University, Iran. Up to now, he has authored/coauthored more than 50 academic articles. His research interests include formal specification and verification, model checking, fog and cloud computing, and the IoT, data mining, and social networks. He is currently an Associate Editor of *Human-Centric Computing and Information Sciences* (Springer), *Cluster Computing* (Springer), and *IET Communications* (IEEE) journals.

AMIN SALIH MOHAMMED received the bachelor's and master's degrees from the Kharkiv National University of Radio Electronics, and the Ph.D. degree from the Kharkiv National University of Radio Electronics, in 2012. He has nearly 15 years of experience in teaching both UG and PG program. He is currently an Assistant Professor with the Department of Computer Networking, Lebanese French University, and the University of Salahaddin—Erbil. He has published over 26 articles in international and national journals and conferences. His fields of interests include wireless networks, ad-hoc networks, and information security.

MOAYAD YOUSIF POTRUS received the B.Sc. degree in electrical engineering and the M.Sc. degree in computer engineering from the University of Baghdad, Iraq, in 1997 and 2000, respectively, and the Ph.D. degree in computer engineering from University Sains Malaysia, in 2012. He is currently a full time Associate Professor with the Department of Software and Informatics Engineering, Salahaddin University–Erbil, Iraq. His research interests are in the field of machine learning, pattern recognition, global optimization, and software testing.

MAZHAR HUSSAIN MALIK received the Ph.D. degree in computer science with specialization in computer networks. He worked in academic and different industrial positions for various universities and companies, including institute of Southern Punjab, Multan Pakistan, SBE electronics, U.K., NCR Corporation, USA, The University of Lahore, Islamabad. Since September 2017, he has been a Senior Lecturer with the Global College of Engineering and Technology—Muscat, Muscat, Oman. He is an Editorial Board Member of peer-reviewed journals and Reviewer of IEEE Access, the *International Journal of Advanced Computer Science and Applications* (IJACSA), and the *International Journal of Computer Science and Information Security* (IJCSIS). His research interests include wireless communications, cloud computing, sensors networks, network security and forensics, and artificial intelligence.

FATEMEH SAFARA received the B.Sc. degree (Hons.) in applied mathematics from Islamic Azad University, the M.Sc. degree in data warehousing from the Tarbait Modaress University of Tehran, and the Ph.D. degree in biological signal processing from University Putra Malaysia, in 2014. She was a Faculty Member with the Information Technology Department, Iran Telecommunication Research Center, from 2000 to 2010, doing research on image mining and biometric signals. She joined Islamic Azad University Islamshahr Branch, in 2010, where she is currently an Assistant Professor with the Computer Engineering Faculty. Her current research interests include signal processing and in particular biological signal processing, data mining, the IoT applications, and cloud computing.

MEHDI HOSSEINZADEH received the B.S. degree in computer hardware engineering from Islamic Azad University Dezfol Branch, Iran, in 2003, and the M.Sc. and Ph.D. degrees in computer system architecture from the Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2005 and 2008, respectively. He is currently an Associate professor with the Iran University of Medical Sciences (IUMS), Tehran. His research interests include SDN, information technology, data mining, big data analytics, e-commerce, e-marketing, and social networks.