UDC 004.7 **doi:** https://doi.org/10.20998/2522-9052.2026.1.06

Volodymyr Panchenko, Heorhii Kuchuk, Valentyn Noskov, Sergey Leonov, Oksana Lipchanska

National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine

METHOD OF TEST POOL SYNTHESIS FOR AN INTELLIGENT HIGH-DENSITY IOT EDGE-LAYER GATEWAY

Abstract. Relevance. High-density IoT environments are characterized by a large concentration of sensors and devices that exchange data intensively within a limited space. Under such conditions, edge-layer intelligent gateways become particularly important. These gateways can locally process information, optimize traffic, and ensure consistent interaction among heterogeneous devices. The development of a test pool for an edge-layer intelligent gateway in high-density IoT is relevant due to the rapid growth in the number of connected devices and the increase in their spatial density. In such conditions, the gateway must maintain stable operation despite high levels of radio interference and competition for network resources. An additional challenge is the heterogeneity of the IoT environment, as devices use different protocols, have different data formats, and exhibit diverse load profiles. Without a specially constructed test pool, it is impossible to reliably evaluate the behavior of the gateway under a realistic mix of technologies and topologies. However, due to substantial heterogeneity, the space of possible test-pool configurations has very high dimensionality. Moreover, there are significant time and resource constraints associated with operating the test pool.
The subject of this study is the methods for constructing test pools. **The purpose of the article** is to develop a method for synthesizing a test pool for an edge-layer intelligent gateway in high-density IoT. **The following results** were obtained. A five-layer architecture of an edge-layer intelligent gateway for high-density IoT is proposed. The operational specifics of the gateway and the particular aspects of its testing are identified. The task of synthesizing the test pool is reduced to a combinatorial problem of selecting an optimal configuration within an extremely large state space. To solve it, the use of a classical genetic algorithm is proposed. The proposed algorithm made it possible, within an acceptable time, to obtain a test pool with nearly minimal execution time, a minimal number of tests, and maximal coverage of the gateway components. **Conclusion.** The proposed method enables the construction of a test pool for an intelligent gateway within a high-dimensional state space while meeting the specified requirements. Future research concerns the development of a method for reducing the dimensionality of the state space of individual tests for gateway components.

Keywords: Internet of Things; computer system; intelligent gateway; edge layer; IoT sensors.

Introduction

Problem Statement. The rapid development of the Internet of Things (IoT) has contributed to the formation of heterogeneous infrastructures [1]. Modern large-scale IoT deployments are characterized by significant variations in spatial device density [2]. In such high-density IoT environments, complex interactions emerge that cannot be accurately characterized by traditional communication and data processing models [3]. This issue is particularly acute during the operation of the IoT ecosystem's edge layer [4]. The edge layer determines the key characteristics of data flows, load balancing, and system energy efficiency [5]. Notwithstanding its criticality, the behavior of the high-density IoT edge layer remains largely unexplored [6]. Existing approaches do not adequately address the real-world operating conditions of heterogeneous environments [7]. Consequently, operational models of edge layer devices often lead to inaccurate estimates of latency, throughput, reliability, and other critical performance indicators [8].

High-density IoT environments require a specialized edge gateway designed to integrate diverse data flows [9]. This gateway functions as a bridge linking edge devices with higher-level computing tiers, such as fog and cloud layers. It provides adaptive data preprocessing, thereby reducing the load on the network infrastructure and improving system scalability [10]. Owing to heterogeneous device density and fluctuations in traffic load, the gateway needs to support dynamic resource balancing [11, 12]. The effective functioning of the edge layer gateway requires comprehensive testing, as it is critical for facilitating interoperability among heterogeneous devices and services [13]. Testing allows for assessing the gateway's capability to maintain stable operation under dynamic network topology shifts and load fluctuations [14]. Particular attention must be paid to verifying its performance in scenarios featuring non-uniform device density, which is typical of high-density IoT infrastructures. A significant challenge is the high level of heterogeneity, encompassing diverse communication protocols, data formats, computational capabilities, and device energy resources. This hinders the development of universal test scenarios that accurately capture the system's actual operating environment [15]. An additional layer of complexity arises from the necessity to ensure proper routing and real-time processing of incoming data from diverse device classes. The issue of interface and protocol harmonization also arises, which may lead to data incompatibility during testing [16].

The significant heterogeneity of devices and communication channels results in a large number of isolated tests aimed at checking individual gateway components [17]. Therefore, the problem arises of selecting a test pool that achieves maximal gateway coverage under time constraints.

Literature review. Paper [18] proposes a metamodeling-based approach for creating tests for IoT systems characterized by high heterogeneity of devices and standards. By formalizing system behavior, this method facilitates automated test case generation, thereby streamlining the testing process for complex IoT architectures comprising gateways and varied end nodes.

The authors of [19] propose a method for the automatic generation of test cases for IoT devices using modern Natural Language Processing (NLP) models. This represents a significant advancement, as it enables the generation of structured, programmatically described

tests, potentially enhancing the testing efficiency of IoT gateways and peripheral components.

There are various approaches to constructing an optimal test subset from a large pool of tests for IoT edge gateways.

Paper [20] suggests employing automated test selection frameworks to analyze the existing test set and determine the tests with the highest fault detection capability. In article [21], the testing of the IoT edge layer gateway focuses solely on stabilizing load control. Article [22] employs multi-objective evolutionary algorithms and linkage learning techniques for test selection.

However, this approach fails to consider the specifics of intelligent IoT edge layer gateways. The studies presented in [23, 24] are more focused on energy efficiency and anomaly detection issues.

The approaches proposed in [25] are primarily designed for resource-constrained environments.

To achieve this, various metrics are used during test case analysis.

This reduces the time required by ensuring key tests are executed first.

Consequently, all the considered works [18–25] overlook the particularities of the edge layer gateway.

The purpose of the research is to develop a method of test pool synthesis for an intelligent high-density IoT edge-layer gateway. To achieve the purpose, the following tasks are solved:

1) analyze the operational features of an intelligent high-density IoT edge layer gateway;

2) identify the specific characteristics of testing an intelligent high-density IoT edge layer gateway;

3) develop a genetic algorithm for synthesizing a test pool for the Intelligent Edge Layer Gateway (IELG) in High-Density IoT (HDIoT).

1. Intelligent Edge-Layer Gateway Architecture for High-Density IoT

The IELG HDIoT represents a complex, intelligent computing unit. It is capable of simultaneously processing data from a large number of IoT devices, reaching up to tens of thousands [26]. The specific features of the IELG are as follows [27]:

– heterogeneity of communication channels and protocols used by HDIoT devices;

– protocol conversion capability;

– support for the dynamic mobility of various IoT sensors;

– local analytics capability, specifically the integration of Artificial Intelligence for preliminary filtering, classification, and short-term forecasting;

– local data processing followed by aggregation and transmission to the fog layer;

– Over-The-Air (OTA) update capability, defined as the remote updating of software, firmware, or device configurations via wireless or network channels without physical access to the device;

– assurance of real-time operation in heterogeneous and unstable networks.

The IELG typically consists of five main layers (Fig. 1).

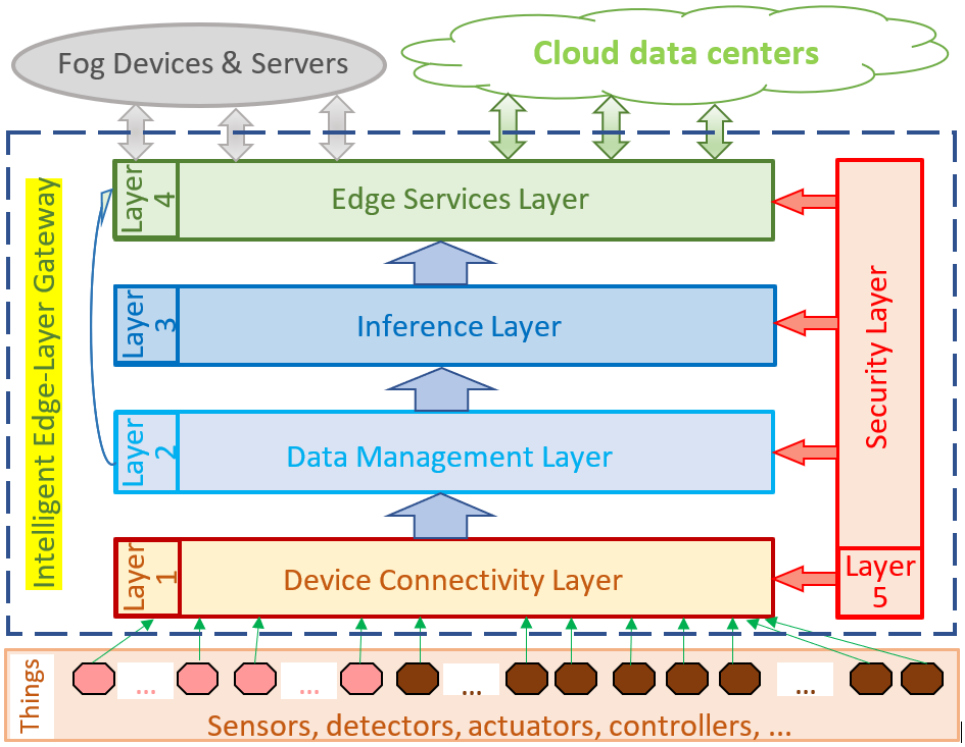

Fig. 1. IELG Architecture for HDIoT

1.1. **Device Connectivity Layer**. This layer provides connectivity between the gateway and IoT end devices. It supports both wired interfaces (RS485/Modbus, CAN, Ethernet, BACnet, OPC-UA), and wireless technologies (Wi-Fi, BLE, Zigbee, Z-Wave, Thread, LoRaWAN, NB-IoT, 4G/5G). The key functions of this layer include:

– collecting data from sensors;

– performing initial data validation and normalization;

– managing communication protocols such as MQTT, CoAP, Modbus, OPC-UA.

1.2. **Data Management Layer**. This level performs data preprocessing for subsequent AI analysis. This involves the following operations:

– buffering data in local queues and caching;

– filtering and deduplicating data, handling noise;

– aggregating data within time windows;

– converting formats;

– managing routing;

– transmitting data to the fog or cloud IoT layers.

The following technologies are employed: Lightweight databases (SQLite, InfluxDB, TimescaleDB) and message brokers (MQTT Broker, Kafka-lite, EMQX).

1.3. Edge AI Layer (**Inference Layer**). This layer serves as the 'brain' of the intelligent gateway. It is capable of autonomously executing the following tasks when required:

– running ML/DL models locally;

– detecting anomalies;

– short-term prediction (predictive maintenance);

– computer vision tasks;

– making decisions in real-time.

In this context, the following accelerator processors are used: TPU: Google Coral; GPU: NVIDIA Jetson; NPU: ARM Ethos-U, Huawei Ascend; VPU: Intel Movidius.

Such processors represent the optimal solution for the IoT ecosystem. The most effective frameworks and inference engines are: TensorFlow Lite; ONNX Runtime; OpenVINO; NVIDIA TensorRT; PyTorch Mobile.

1.4. Edge Containerization and Services Layer (**Edge Services Layer**). This layer handles the deployment of components and microservices, execution of updates, AI model deployment, and application lifecycle management.

1.5. **Security Layer**. This layer spans all architectural levels and provides:

– on-device data encryption;

– TLS/DTLS for communication protocols;

– certificate management;

– access control (RBAC).

2. Key Testing Aspects of an IELG for HDIoT

Testing the intelligent high-density IoT gateway involves specific unique features, most of which arise from the presence of a vast number of heterogeneous devices operating over diverse protocols, as well as from highly non-uniform distributed loads [28]. Furthermore, testing must encompass network, hardware, behavioral, security, and intelligent aspects [29, 30].

It should also be noted that testing such a device is inherently time-consuming and resource-intensive, because:

– some tests require real Radio Frequency (RF) traffic;

– some tests take minutes to complete (e.g., network recovery);

– there are dependencies on hardware setups, timing constraints, sensor lifespans, etc.

Thus, the issue of synthesizing a test pool for the intelligent high-density IoT gateway becomes highly relevant. Considering the large number of existing tests for verifying individual gateway components, the problem of synthesis essentially reduces to identifying an optimal test set.

Therefore, it is necessary to solve a combinatorial problem of selecting an optimal configuration within a very large state space, where traditional methods become inefficient [31].

A near-optimal solution can be found more rapidly using evolutionary algorithms [32]. In particular, genetic algorithms can be considered. For this purpose, an individual test is treated as a gene, and a separate scenario is treated as a chromosome. Consequently, the current population consists of a set of scenarios, and the fitness function evaluates the quality of the current population.

The fitness function must evaluate *the effectiveness of the selected population* for fault detection, gateway loading, and verifying its reliability. Considering the features of this intelligent gateway, compactness (minimized test set) and testing speed (reduced execution time) serve as crucial factors in designing the fitness function. At the same time, component coverage should remain as high as possible. Therefore, it is necessary to first select a fitness function that accounts for the overall quality of the test suite.

Existing constraints and the convergence rate of the genetic algorithm can be tuned via crossover parameters.

3. Genetic algorithm for test pool synthesis for IELG

Let there be a requirement to test an IELG consisting of N components, composed of N_F functional, N_P protocol, N_C code, and N_S security components, where

$$N_F + N_P + N_C + N_S = N. \quad (1)$$

The tests are numbered from 1 to N, where the first N_F numbers are allocated to functional components.

Testing can be performed using a set of K tests. Each test can verify several gateway components. The 'test-component' relation is defined by the Boolean matrix

$$W = (w_{kn}),\ k = 1..K,\ n = 1..N, \quad (2)$$

where $w_{kn} = 1$, if test with index k verifies component n; otherwise $w_{kn} = 0$.

The execution time of each test is defined by the vector

$$V = (v_k),\ k = 1..K. \quad (3)$$

Some tests can be executed in parallel. However, this leads to an increase in execution time.

The grouping of tests into such subsets is defined by the matrix

$$S = (s_{k1,k2}),\ k1,\ k2 = 1..K,\ k1 \geq k2, \quad (4)$$

where $s_{k1,k2} = 0$, if tests indexed by $k1$ and $k2$ cannot be executed in parallel. In cases where parallel execution is possible, $s_{k1,k2}$ represents the percentage by which the execution time of these tests increases.

In addition, the overall execution time for IELG testing is bounded by T_{max}.

A classical genetic algorithm is proposed to find a near-optimal solution.

Each single test represents a gene. A chromosome consists of a collection of genes that represent a potential test set configuration. Formally, each chromosome is described by a Boolean vector

$$C = (c_k),\ k = 1..K, \tag{5}$$

where $c_k = 1$, if the gene with index k is included in the given chromosome; otherwise $c_k = 0$.

The number of chromosomes in the m-th generation constitutes $I(m)$. To avoid excessively small test set sizes, a constraint is imposed on the minimum number of tests in a chromosome:

$$\sum_{k=1}^{K} c_k \geq K_{min}\ , \tag{6}$$

where K_{min} – the minimum permissible number of tests allowed in the solution set.

The initial population $P(1)$ of size $I(1)$ is generated randomly, subject to the inequality (6):

$$P(1) = \{C_{11}, C_{12},\ldots,C_{1i},\ldots, C_{1I(1)}\}, \tag{7}$$

where $C_{1i} = (c_{1ik}),\ i = 1..\ I(1),\ k = 1..K.$

The sequence of populations is defined by the tuple

$$\Re = (P(1), P(2), \ldots, P(\theta), \ldots), \tag{8}$$

where θ – the sequential number of the current population, and card $P(\theta) = I(\theta)$.

Sequence (8) terminates when the current population satisfies one of the genetic algorithm's termination conditions.

To simplify the notation, the generation index is omitted when considering the current population.

3.1. Test pool execution time. The following algorithm is proposed to calculate the average duration of the test set defined by $C_i = (c_{ik})$.

Step 0. Preliminarily, two auxiliary vectors of length K are initialized:

$$R = (r_k) = (c_k),\ k = 1..K; \tag{9}$$

$$T = (t_k),\ k = 1..K,$$

$$t_k = 0\ \forall\ k \in 1..K. \tag{10}$$

Additionally, at this step, a variable j is introduced as the index of the current test under analysis, where $j = 1..K;\ j = 0$.

Step 1. At this step, the process proceeds to the next test; $j = j + 1$. If $j > K$, the process proceeds to step 3.

Step 2. If $r_j = 0$, the current test is not included in the given chromosome; therefore, the process proceeds to Step 1.

Otherwise, the test execution time is established and recorded in the corresponding element of vector T:

$$t_j = v_j. \tag{11}$$

Subsequently, a check is performed for tests that can be executed in parallel with test j. This is accomplished via a sequential analysis of the rows of matrix S, starting from the current test:

```
for η in range (j, K + 1):
    if (r_η = 1) and (s_iη > 0):
        r_η = 0
        if v_η · (1 + s_iη) > 0:
            t_j = v_η · (1 + s_iη)
```

Consequently, the possibility of parallel execution of certain tests is taken into account. The process then proceeds to Step 1.

Step 3. Upon completion, the test execution duration vector T is obtained. Thus, the mean duration of the test set specified by chromosome C_i, is calculated as

$$\tau_i = \tau(C_i) = \sum_{k=1}^{K} t_k\ . \tag{12}$$

3.2. Fitness function formulation. The test count for the n-th component ($n = 1..N$) using chromosome C_i is computed as

$$\xi_n^{'} = \sum_{k=1}^{K} c_{ik} \cdot w_{kn}\ . \tag{13}$$

When formulating the fitness function, priority should be given to components with lower test count in the synthesized set. Furthermore, functional components take precedence over other component groups. Therefore, during test coverage normalization, the following normalized values are derived for the components:

$$\xi_n = \begin{cases} 0, & \xi_n^{'} = 0; \\ 1/\xi_n^{'}, & \xi_n^{'} \neq 0,\ n > N_F; \\ 2/\xi_n^{'}, & \xi_n^{'} \neq 0,\ n \leq N_F. \end{cases} \tag{14}$$

Thus, the fitness function for chromosome C_i takes the following form:

$$\Phi(C_i) = \omega_1 \cdot \frac{\sum_{n=1}^{N} \xi_n}{N + N_F} + \omega_2 \cdot \frac{\tau_{max} - \tau_i}{\tau_{max} - \tau_{min}}, \tag{15}$$

where ω_1, ω_2 are weight coefficients, $\omega_1 + \omega_2 = 1$; τ_{max}, τ_{min} – are the maximum and minimum possible durations for gateway testing using subsets chosen from the K test pool.

The values τ_{max} and τ_{min} are calculated using the presented algorithm for determining the average test set duration:

$$\tau_{max} = \tau(C_{max}),\quad C_{max} = (1,\ldots,1,\ldots 1); \tag{16}$$

$$\tau_{min} = \tau(C_{min}),\quad \sum_{k=1}^{K} c_{min,k} = K_{min}, \tag{17}$$

where chromosome C_{max} contains all possible tests, and chromosome C_{min} represents the minimum set size, containing the fastest available tests.

3.3. Crossover operator specifics. The crossover operator operates on the current population. Therefore, it is necessary to determine the essential general characteristics of the population.

The following parameter characterizes the coverage of gateway components by the current population's chromosomes:

$$\Delta_+ = \frac{\sum_{k=1}^{K}\left(\delta\left(\sum_{i=1}^{I} c_{ik}\right)\right)}{K}, \quad (18)$$

where $$\delta(x) = \begin{cases} 0, & x = 0; \\ 1 & x \neq 0. \end{cases}$$

This parameter takes values in the interval from $1/K$ to 1, with the maximum value $\Delta_+ = 1$ being achieved in the case of complete test coverage of the gateway components.

One of the algorithm termination conditions is exceeding the minimum predefined fraction of tested gateway components Δ_{min}, i.e.,

$$\Delta_+ > \Delta_{\min}. \quad (19)$$

Let M_{N_-} denote the set of gateway components that remain untested within the current population.

Accordingly, the fraction of untested gateway components is

$$\Delta_- = 1 - \Delta_+, \quad (20)$$

while the number of unserved components is:

$$N_- = \sum_{k=1}^{K}\left(1 - \delta\left(\sum_{i=1}^{I} c_{ik}\right)\right). \quad (21)$$

Therefore, card $M_{N_-} = N_-$.

Based on the objective of enhancing component test coverage, the following conditions are added to the crossover mechanism:

1) if condition (19) is not satisfied for the current population, the size of the next population is increased, i.e.,

$$I(\theta+1) > I(\theta), \quad (22)$$

where θ – is the index of the current population;

2) genes from the set of untested gateway components M_{N_-}, are subject to mutation; moreover, as N_- increases, the percentage of mutated genes increases too.

Furthermore, the testing time constraint is significant for the intelligent gateway. Consequently, chromosomes where $T_i > T_{max}$, are excluded from reproduction.

4. Discussion of results

To verify the obtained results, a generalized model of an AI-Enabled Secure Multi-Protocol IoT Edge Gateway was considered. This gateway operates at the network edge, handling sensor data processing, security, and autonomous analytics. The testing targeted 18 components responsible for network functions, protocol integration, machine learning, and hardware management:

- n1, token validation & revocation list;
- n2, MQTT connector (QoS handling);
- n3, MQTT session & state management;
- n4, CoAP large-payload handling;
- n5, REST API gateway;
- n6, packet parser;
- n7, load balancer & dispatcher;
- n8, upstream health check;
- n9, stream pre-processor;
- n10, anomaly detector & ML pipeline;
- n11, OTA & update manager;
- n12, sensor ingestion pipeline;
- n13, buffering subsystem;
- n14, power management & battery monitor;
- n15, hardware control & reset manager;
- n16, cache subsystem;
- n17, metrics exporter & telemetry;
- n18, notification pipeline.

A set of 30 tests were proposed for the testing process. Each test was capable of testing between 1 and 6 different gateway components.

Figure 2 illustrates the simulation results regarding the relationship between total testing time, gateway component coverage requirements, and the genetic algorithm's initial population size.

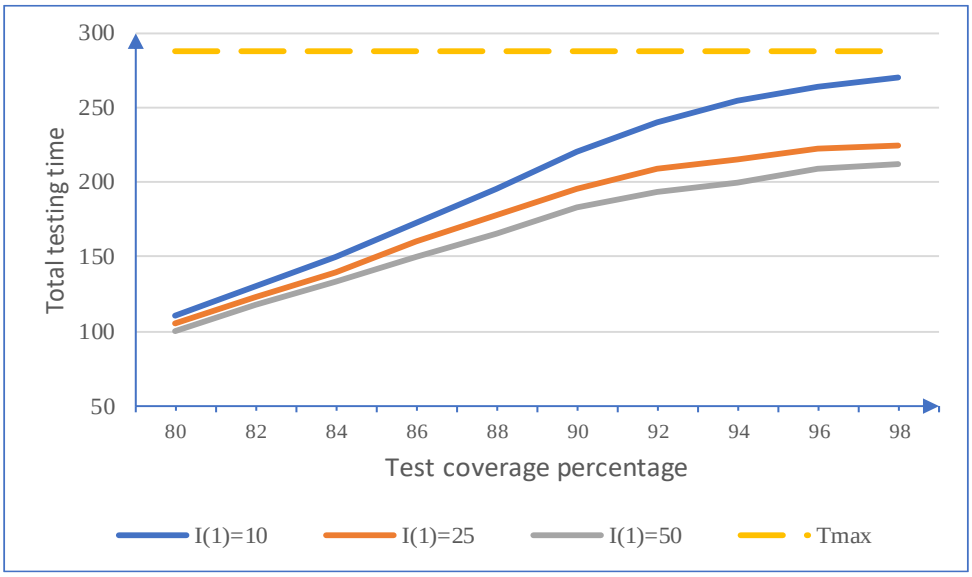

Fig. 2. Results of the gateway testing time calculation

It is evident that the testing time rises with stricter coverage requirements. For small initial populations, the testing time grows quickly, converging to the maximum. However, a significant increasing the initial population size (when the population size is greater than the chromosome size) provides negligible time benefits.

Similar conclusions regarding the initial population size can also be extended to the analysis of the tests used (Fig. 3).

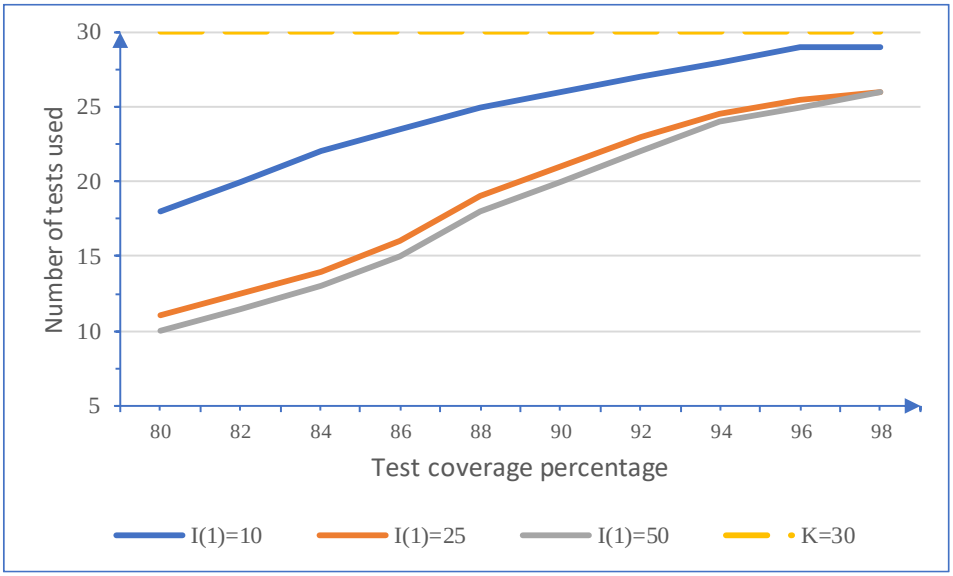

Fig. 3. Test count analysis

Conclusions

The article proposes a method for synthesizing a test pool for an edge-layer intelligent gateway in high-density IoT. The following tasks were considered when developing the method:

1. A five-layer architecture of an edge-layer intelligent gateway for high-density IoT is proposed. The operational specifics of the gateway and the particular aspects of its testing are identified.

2. The operational specifics of the gateway and the particular aspects of its testing are identified.

3. The task of synthesizing the test pool is reduced to a combinatorial problem of selecting an optimal configuration within an extremely large state space. To solve it, the use of a classical genetic algorithm is proposed.

The proposed algorithm made it possible, within an acceptable time, to obtain a test pool with nearly minimal execution time, a minimal number of tests, and maximal coverage of the gateway components.

Future research concerns the development of a method for reducing the dimensionality of the state space of individual tests for gateway components.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

Acknowledgements

The study was funded by the Ministry of Education and Science of Ukraine in the framework of the research project 0125U001544 on the topic “Methodology for ensuring the processes of monitoring and controlling the implementation of project and program portfolios for project offices in the context of Ukraine's reconstruction”.

REFERENCES

1. Lee, B.M. (2025), “Efficient Resource Management for Massive MIMO in High-Density Massive IoT Networks”, *IEEE Transactions on Mobile Computing*, vol. 24(3), pp. 1963–1980, doi: https://doi.org/10.1109/TMC.2024.3486712
2. Kuchuk, H., Mozhaiev, O., Tiulieniev, S., Mozhaiev, M., Kuchuk, N., Tymoshchyk, L., Lubentsov, A., Onishchenko, Y., Gnusov, Y. and Tsuranov, M. (2025), “Devising a method for increasing data transmission speed in monitoring systems based on the mobile high-density Internet of Things”, *Eastern-European Journal of Enterprise Technologies*, *3*(4 (135)), pp. 52–61, doi: https://doi.org/10.15587/1729-4061.2025.330644
3. Kuchuk, H., Mozhaiev, O., Tiulieniev, S., Mozhaiev, M., Kuchuk, N., Lubentsov, A., Onishchenko, Yu., Gnusov, Yu., Brendel, O. and Roh, V. (2025), “Devising a method for energy-efficient control over a data transmission process across the mobile high-density Internet of Things”, *Eastern European Journal of Enterprise Technologies*, vol. 4(4(136)), pp. 46–57, doi: https://doi.org/10.15587/1729-4061.2025.336111
4. Moreno-Motta, J., Moreno-Vera, F. and Moreno, F.A. (2022), “MorArch: A Software Architecture for Interoperability to Improve the Communication in the Edge Layer of a Smart IoT Ecosystem”, *Lecture Notes in Networks and Systems*, vol 286, Springer, Singapore, pp. 185–195, doi: https://doi.org/10.1007/978-981-16-4016-2_18

5. Vakaliuk, T.A., Andreiev, O.V., Dubyna, O.F., Korenivska, O.L. and Andreieva, Y.O. (2024), "Use of wireless technologies in IoT projects", *Journal of Edge Computing*, vol. 3, no. 2, pp.147–167, doi: https://doi.org/10.55056/jec.750
6. Kuchuk, N., Kashkevich, S., Radchenko, V., Andrusenko, Y. and Kuchuk, H. (2024), "Applying edge computing in the execution IoT operative transactions", *Advanced Information Systems*, vol. 8, no. 4, pp. 49–59, doi: https://doi.org/10.20998/2522-9052.2024.4.07
7. Abdelwahed, S.H., Hefny, I.M., Hegazy, M., Said, L.A., and Soltan, A. (2025), "Survey of IoT multi-protocol gateways: Architectures, protocols and cybersecurity", Internet of Things (The Netherlands), vol. 33, article number 101703, doi: https://doi.org/10.1016/j.iot.2025.101703
8. Kuchuk, H., Husieva, Y., Novoselov, S., Lysytsia, D., Krykhovetskyi, H. (2025), "Load Balancing of the layers Iot Fog-Cloud support network", *Advanced Information Systems*, vol. 9, no. 1, pp. 91–98, doi: https://doi.org/10.20998/2522-9052.2025.1.11
9. Castellanos, W., Macias, J., Pinilla, H., and Alvarado, J. D. (2020), "Internet of things: a multiprotocol gateway as solution of the interoperability problem", *Mechatronics, Electronics and Telecommunications Advances Towar Industry 4.0*, pp. 85–105, doi: https://doi.org/10.48550/arXiv.2108.00098
10. Pagliari, E., Davoli, L. and Ferrari, G. (2024), "Harnessing Communication Heterogeneity: Architectural Design, Analytical Modeling, and Performance Evaluation of an IoT Multi-Interface Gateway", *IEEE Internet of Things Journal*, vol. 11(5), pp. 8030–8051, doi: https://doi.org/10.1109/JIOT.2023.3317672
11. Rezanov, B. and Kuchuk, H. (2023), "Model of elemental data flow distribution in the Internet of Things supporting Fog platform", *Innovative Technologies and Scientific Solutions for* gateway *Industries*, vol. 2023(3), pp. 88–97, doi: https://doi.org/10.30837/ITSSI.2023.25.088
12. Kuchuk, H., Mozhaiev, O., Kuchuk, N., Tiulieniev, S., Mozhaiev, M., Gnusov, Y., Tsuranov, M., Bykova, T., Klivets, S., and Kuleshov, A. (2024), "Devising a method for the virtual clustering of the Internet of Things edge environment", *Eastern-European Journal of Enterprise Technologies*, vol. 1, no. 9 (127), pp. 60–71, doi: https://doi.org/10.15587/1729-4061.2024.298431
13. Cao, W., Kosenko, V. and Semenov, S. (2022), "Study of the efficiency of the software security improving method and substantiation of practical recommendations for its use", *Innovative Technologies and Scientific Solutions for Industries*, vol. 1(19), pp. 55–64, doi: https://doi.org/10.30837/ITSSI.2022.19.055
14. Shah, Q. A., Shafi, I., Ahmad, J., Alfarhood, S., Safran, M., and Ashraf, I. (2023), "A Meta Modeling-Based Interoperability and Integration Testing Platform for IoT Systems", *Sensors*, vol. *23*(21), 8730, doi: https://doi.org/10.3390/s23218730
15. Oliveira, F., Costa, D.G., Assis, F., and Silva, I. (2024), "Internet of Intelligent Things: A convergence of embedded systems, edge computing and machine learning", Internet of Things (The Netherlands), vol. 26, 101153, doi: https://doi.org/10.1016/j.iot.2024.101153
16. Zhurylo, O., Liashenko, O. & Avetisova, K. (2023), "Hardware Security Overview of Fog Computing End Devices in the Internet of Things", *Innovative Technologies and Scientific Solutions for Industries*, vol, 23, pp. 57–71, doi: https://doi.org/10.30837/ITSSI.2023.23.057
17. Urblik, L., Kajati, E., Papcun, P. and Zolotová, I. (2024), "Containerization in Edge Intelligence: A Review", *Electronics*, vol. 13, no. 7, 1335, doi: https://doi.org/10.3390/electronics13071335
18. Shah, Q. A., Shafi, I., Ahmad, J., Alfarhood, S., Safran, M., and Ashraf, I. (2023), "A Meta Modeling-Based Interoperability and Integration Testing Platform for IoT Systems", *Sensors*, vol. *23*, no. 21, 8730, doi: https://doi.org/10.3390/s23218730
19. Kumar, S., Napte, K., Rani, R. and Pippal, S.K. (2025), "A method for IoT devices test case generation using language models", *Methodsx*, vol. 14, 103340, doi: https://doi.org/10.1016/j.mex.2025.103340
20. Marques, F., Morgado, A., Fragoso Santos, J. and Janota, M. (2022), "TestSelector: Automatic Test Suite Selection for Student Projects", Lecture Notes in Computer Science, vol 13498. Springer, Cham. https://doi.org/10.1007/978-3-031-17196-3_17
21. Kuchuk, H., Mozhaiev, O., Tiulieniev, S., Mozhaiev, M., Kuchuk, N., Tymoshchyk, L., Lubentsov, A., Gnusov, Y., Klivets, S. and Kuleshov, A. (2025), "Devising a method for stabilizing control over a load on a cluster gateway in the internet of things edge layer", *Eastern-European Journal of Enterprise Technologies*, vol. *2*(9(134)), pp. 24–32, doi: https://doi.org/10.15587/1729-4061.2025.326040
22. Olsthoorn, M. and Panichella, A. (2021), "Multi-objective Test Case Selection Through Linkage Learning-Based Crossover", Lecture Notes in Computer Science(), vol 12914, Springer, Cham. doi: https://doi.org/10.1007/978-3-030-88106-1_7
23. Yareshchenko, V. and Kosenko, V. (2024), "low-power coding method in data transmission systems", *Innovative Technologies and Scientific Solutions for Industries*, vol. 3(29), pp. 121–129, doi: https://doi.org/10.30837/2522-9818.2024.3.121
24. Semenov, S., Mozhaiev, O., Kuchuk, N., Mozhaiev, M., Tiulieniev, S., Gnusov, Yu., Yevstrat, D.,Chyrva, Y., Kuchuk, H. (2022), "Devising a procedure for defining the general criteria of abnormal behavior of a computer system based on the improved criterion of uniformity of input data samples", *Eastern-European Journal of Enterprise Technologies*, vol. 6(4-120), pp. 40–49, doi: https://doi.org/10.15587/1729-4061.2022.269128
25. Garg, K. and Shekhar, S. (2024), "Test case prioritization based on fault sensitivity analysis using ranked NSGA-2", *Int. j. inf. tecnol.*, vol.16, pp. 2875–2881, doi: https://doi.org/10.1007/s41870-024-01868-0
26. Dogea, R., Yan, X. T., and Millar, R. (2023), "Implementation of an edge-fog-cloud computing IoT architecture in aircraft components*", MRS Communications*, vol. 13(3), pp. 416–424, doi: https://doi.org/10.1557/s43579-023-00364-z
27. Zhang, Y., Yu, H., Zhou, W., and Man, M. (2023), "Application and research of IoT architecture for end-net-cloud edge computing*", Electronics*, vol. *12*(1), 1, doi: https://doi.org/10.3390/electronics12010001
28. Kuchuk, H., Kalinin, Y., Dotsenko, N., Chumachenko, I. and Pakhomov, Y. (2024), "Decomposition of integrated high-density IoT data flow", *Advanced Information Systems*, vol. 8, no. 3, pp. 77–84, doi: https://doi.org/10.20998/2522-9052.2024.3.09
29. Kyrychok, R., Laptiev, O., Lisnevsky, R., Kozlovsky, V. and Klobukov V. (2022), "Development of a method for checking vulnerabilities of a corporate network using Bernstein transformations", *Eastern-European Journal of Enterprise Technologies*, vol. 1, no. 9 (115), pp. 93–101, doi: https://doi.org/10.15587/1729-4061.2022.253530
30. Mani Kiran, C.V.N.S., Jagadeesh Babu, B. and Singh, M.K. (2023), "Study of Different Types of Smart Sensors for IoT Application Sensors", S*mart Innovation, Systems and Technologies*, vol. 290, pp. 101–107, doi: https://doi.org/10.1007/978-981-19-0108-9_11

31. Kuchuk, H., Mozhaiev, O., Tiulieniev, S., Mozhaiev, M., Kuchuk, N., Tymoshchyk, L., Onishchenko, Yu., Tulupov, V., Bykova, T. and Roh, V. (2025), "Devising a method for forming a stable mobile cluster of the internet of things fog layer", *Eastern-European Journal of Enterprise Technologies*, 2025, vol. 1, no. 4(133), pp. 6–14, doi: https://doi.org/10.15587/1729-4061.2025.322263
32. Taha, Z.Y., Abdullah, A.A. and Rashid, T.A. (2025), "Optimizing feature selection with genetic algorithms: a review of methods and applications", *Knowl Inf Syst*, vol. 67, pp. 9739–9778, doi: https://doi.org/10.1007/s10115-025-02515-1

Received (Надійшла) 12.10.2025
Accepted for publication (Прийнята до друку) 21.01.2026

ВІДОМОСТІ ПРО АВТОРІВ/ ABOUT THE AUTHORS

Панченко Володимир Іванович – старший викладач кафедри комп'ютерної інженерії та програмування, Національний технічний університет «Харківський політехнічний інститут», Харків, Україна;
Volodymyr Panchenko – Senior Lecturer of the Computer Engineering and Programming Department, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine;
e-mail: Volodymyr.Panchenko@khpi.edu.ua; ORCID Author ID: https://orcid.org/0000-0003-3364-3398;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=58759071400.

Кучук Георгій Анатолійович – доктор технічних наук, професор, професор кафедри комп'ютерної інженерії та програмування, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Heorhii Kuchuk – Doctor of Technical Sciences, Professor, Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: kuchuk56@ukr.net; ORCID Author ID: http://orcid.org/0000-0002-2862-438X;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57057781300.

Носков Валентин Іванович – доктор технічних наук, доцент, професор кафедри комп'ютерної інженерії та програмування, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна;
Valentyn Noskov – Doctor of Technical Sciences, Associate Professor, Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: valentyn.noskov@khpi.edu.ua; ORCID Author ID: http://orcid.org/0000-0002-7879-0706;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57331254200.

Леонов Сергій Юрійович – доктор технічних наук, професор, професор кафедри комп'ютерної інженерії та програмування, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна
Sergey Leonov – Doctor of Technical Sciences, Professor, Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: serleomail@gmail.com; ORCID Author ID: http://orcid.org/0000-0001-8139-0458;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=7005357987.

Ліпчанська Оксана Валентинівна – кандидат технічних наук, доцент, доцент кафедри комп'ютерної інженерії та програмування, Національний технічний університет "Харківський політехнічний інститут", Харків, Україна
Oksana Lipchanska – Candidate of Technical Sciences, Associate Professor, Associate Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: Oksana.Lipchanska@khpi.edu.ua; ORCID Author ID: http://orcid.org/0000-0003-4173-699X;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57218516877.

Метод синтезу тестового пулу для інтелектуального шлюзу граничного шару високощільного IoT

В. І. Панченко, Г. А. Кучук, В. І. Носков, С. Ю. Леонов, О. В. Ліпчанська

Анотація. Актуальність. Високощільні IoT-середовища характеризуються великим скупченням сенсорів і пристроїв, які інтенсивно обмінюються даними в обмеженому просторі. За таких умов особливої ваги набувають інтелектуальні шлюзи граничного шару. Такі шлюзи здатні локально обробляти інформацію, оптимізувати трафік і забезпечувати узгоджену взаємодію різнорідних пристроїв. Розробка тестового пулу для інтелектуального шлюзу граничного шару високощільного IoT є актуальною через швидке зростання кількості підключених пристроїв та збільшення їх просторової щільності. У таких умовах шлюз має забезпечувати стабільну роботу попри високий рівень радіоперешкод і конкуренцію за мережеві ресурси. Додатковою проблемою є гетерогенність IoT-середовища, тому що пристрої використовують різні протоколи, мають різні формати даних і різні профілі навантаження. Без спеціально сформованого тестового пулу неможливо надійно оцінити поведінку шлюзу за умов реалістичного міксу технологій та топологій. Але внаслідок суттєвої гетерогенності простір можливих варіантів формування пулу має велику розмірність. Крім того, існують суттєві часові та ресурсні обмеження при експлуатації тестового пулу. **Предметом вивчення** в статті є методи формування тестових пулів. **Метою статті** є розробка методу синтезу тестового пулу для інтелектуального шлюзу граничного шару високощільного IoT. Отримано **такі результати**. Запропонована п'ятирівнева архітектура інтелектуального шлюзу граничного шару високощільного IoT. Визначені особливості функціонування шлюзу та специфічні особливості при проведенні його тестування. Задача синтезу тестового пулу зведена до комбінаторної задачі вибору оптимальної конфігурації у дуже великому просторі станів. Для її розв'язання запропоновано використовувати класичний генетичний алгоритм. Запропонований алгоритм дозволив за прийнятний час отримати тестовий пул з практично мінімальним часом виконання, мінімальною кількістю тестів та максимальним покриттям компонент шлюзу. **Висновок.** Запропонований метод дозволяє у просторі станів великої розмірності сформувати тестовий пул для інтелектуального шлюзу, який відповідає заданим вимогам. Напрям подальших досліджень стосується розробки методу зменшення розмірності простору станів окремих тестів компонент шлюзу.

Ключові слова: Інтернет речей; комп'ютерна система; інтелектуальний шлюз; граничний шар; сенсори IoT.