International Journal for Housing Science and Its Applications
Publish August 10, 2025. Volume 46, Issue 4 Pages 7101-7109

https://doi.org/10.70517/ijhsa464614

Data Structure Modeling and Software Test Case Generation Methodology for a Networked System for Housing Buildings

Hongyu Shi[1,*]
[1] Sichuan Vocational and Technical College of Communications, Chengdu, Sichuan, 611130, China
Corresponding authors: (e-mail: shihongyucd2024@126.com).

Abstract Intelligent building networking system realizes real-time monitoring and control of the building environment through sensor technology and data communication protocols, which has become an important trend in the development of modern building intelligence. However, the traditional software testing methods have the problems of low test coverage and low efficiency when facing complex IoT systems. Based on this, this paper proposes a data structure modeling and test case automatic generation method based on improved genetic algorithm. The method constructs a four-layer system architecture including input layer, coding layer, core processing layer and output layer, adopts a combination of binary coding and floating-point coding to represent test cases, designs a multi-objective fitness function that integrally considers code coverage, execution efficiency and resource utilization, and realizes the optimal generation of test cases through improved selection, crossover and mutation operations. The experimental results show that the execution time of the improved genetic algorithm in function optimization is 51.19 seconds, which is 4.65 seconds faster than that of the IAGA algorithm, and the convergence state can be reached in an average of 125.354 iterations, which is significantly better than the comparison algorithm. Validation on seven standard test programs shows that the proposed method achieves an average coverage of 82.5%, which is 8.9% and 4.7% higher than the DTG and NDTG algorithms, respectively. The method can effectively improve the quality of test cases and provides technical support for the reliability assurance of smart building networking systems.

Index Terms smart building networking system, data structure modeling, improved genetic algorithm, test case generation, code coverage, fitness function

I. Introduction

At present, China is in the stage of rapid development of urbanization and industrialization, which will definitely bring more market space to the construction industry [1]. The construction industry has become a pillar of China's economy, so only the healthy and green development of the construction industry can ensure the sound development of the national economy [2]. Based on this, in order to better keep up with the process of social modernization, relevant employees should also introduce IoT technology when planning and implementing intelligent building systems and network systems, hardware and software equipment [3]-[5]. Improvement of smart building systems and built-in functionality ensures that smart building systems can be of key use in creating comfortable living and office spaces [6], [7].

In intelligent system buildings, modern communication, information and computer network technologies are mainly utilized to monitor and control the actual operation of all phases of the construction project and facilities within the building [8], [9]. It meets the construction requirements of public control and information exchange, ensures that the construction environment introduces safety, comfort and environmental protection functions, and improves the rationality of the investment in construction projects [10], [11]. As the construction industry tends to digitize and model the model development, more advanced sensor and information network technologies are applied to IoT building systems to monitor the condition of the building during the construction phase, the condition of the equipment and the space utilization [12]-[15]. The structural modeling and health monitoring methodology of buildings is a key technology to ensure the safety of urban buildings, which is of great practical value for early warning of potential structural failures through continuous monitoring of the structure's indicators [16]-[18].

The current construction industry is experiencing an unprecedented digital transformation, and the rapid development of Internet of Things (IoT) technology has endowed traditional buildings with intelligent characteristics. Modern intelligent building systems integrate a large number of sensing devices, control units and communication modules, forming a complex networked architecture. These systems need to process real-time data from a variety of sensors such as temperature, humidity, light, air quality, etc., while coordinating the operation of multiple subsystems such as air conditioning, lighting, security, etc. However, the increase in system complexity also poses software quality assurance challenges. Traditional manual testing methods are not only time-consuming and labor-intensive, but also difficult to cover the full functional path of the system, especially when dealing with complex scenarios with multiple variables and constraints, which often have testing blind spots. The existence of software defects may lead to abnormal operation of the system, affect the normal use of the building, and even bring potential safety hazards. Therefore, how to efficiently generate high-quality test cases to ensure the reliability and stability of intelligent building networking systems has become a technical challenge to be solved.

Although the existing automated testing methods have improved the testing efficiency to a certain extent, they are still insufficient when facing the special characteristics of building networking systems. Such systems are characterized by high real-time requirements, strong concurrent processing capability, strict fault tolerance requirements, etc., which put forward higher requirements on the quality of test cases. The use cases generated by traditional random testing methods often lack targeting and cannot effectively cover the critical execution path. Although symbolic execution-based methods can systematically explore program paths, they face the path explosion problem when dealing with complex constraints. Genetic algorithm, as a heuristic search method, is able to find an approximate optimal solution in a large-scale search space, which provides a new idea for the automatic generation of test cases.

Based on the above analysis, this study proposes an automatic test case generation method for intelligent building networking system based on improved genetic algorithm. First, a data structure model adapted to the characteristics of the building networking system is established, and a multi-level system architecture is designed to support the test case generation process. Secondly, to address the shortcomings of the traditional genetic algorithm in terms of convergence speed and local search ability, the crossover and mutation operations are improved to enhance the search efficiency of the algorithm. Finally, a fitness function that integrates multiple quality metrics is designed to ensure that the generated test cases can take into account multiple objectives such as code coverage, execution efficiency and resource utilization. The effectiveness and superiority of the proposed method is demonstrated through experimental validation on standard test programs.

II. Fundamentals and technical architecture of networking systems for intelligent buildings

II. A. Fundamentals of IoT technology in buildings

Internet of Things (IoT) technology is the basis for realizing building intelligence, the core of which lies in the use of sensors and data communication protocols to achieve real-time monitoring and control of the building environment.

Sensor technology plays a crucial role in intelligent building networking systems. Sensors in buildings are mainly used to collect various data about the building environment, such as temperature, humidity, light intensity, air quality index and so on. The above sensors are highly sensitive and reliable, and can capture environmental changes in real time and transmit the data to the central control system. For example, temperature sensors can monitor the temperature of a room, and when the temperature is out of the set range, the air conditioning system can be automatically adjusted to keep the indoor temperature comfortable and stable.

Data communication protocols are another key component of IoT technology, which are responsible for defining how devices communicate and transmit collected data to processing centers or other devices. Common data communication protocols used in smart building applications include Zigbee, WiFi, Bluetooth, and more specialized building automation protocols such as BACnet and Modbus. Each of these protocols has its own characteristics, e.g. Zigbee is suitable for data transmission from low-power devices, while BACnet is designed for building automation and is able to support complex interactions and data sharing between a large number of devices.

II. B. Composition of intelligent building systems

The core components of an intelligent building system include an automation control system and an energy management system, and the above systems are the key to realizing building intelligence and improving energy efficiency.

The automation control system is mainly responsible for the automated management of building equipment and environment, such as air-conditioning system, lighting system, security system and so on. Building automation control system is a typical automation control system which controls and monitors various facilities and environmental conditions in a building through a central computer, distributed microprocessors and a variety of sensor devices. Building automation control systems are usually designed in a hierarchical structure, including multiple levels of control units and various input/output devices. The said control units may be decentralized installed microprocessors responsible for collecting data from sensors (e.g., temperature and humidity sensors) and controlling various devices (e.g., fans, pumps, etc.) according to preset programs. The decentralized control method helps to reduce the burden on the central processor and improve the response speed and reliability of the system.

III. Data structure modeling and software test case generation

III. A. Overall system architecture

The overall architecture of the system is divided into 4 parts. (1) The input layer receives the input requirements of the software. (2) The coding layer transforms the test cases into individual forms that can be processed by the genetic algorithm. (3) The core processing layer treats each test case as an individual for selection, crossover and mutation in subsequent genetic operations. (4) The output layer outputs an optimized set of high-quality test cases with corresponding performance analysis results.

III. B. Test Case Coding Representation

During the encoding process, test cases need to be represented as individuals that can be manipulated by the genetic algorithm [19], using different ways of representation depending on the type and range of parameters of the test case. For discrete test parameters, binary encoding is used. Assuming that the test case contains n discrete input parameter, if the parameter value range is [0,1], the binary encoding is of the form test_case = [0,1,0,1,1]. For continuous input parameters such as floating point numbers, the test case can be represented using floating point encoding, e.g., test_case = [0.75,1.2,0.33,0.99], Floating point encoding allows the test case to represent more complex and fine-grained input parameters.

III. C. Adaptation function

The purpose of designing the fitness function is to measure the quality of test cases, which is often evaluated by combining several objectives. Commonly evaluated objectives are code coverage execution efficiency and resource utilization $R(x)$, etc., which are calculated as shown in Equation (1) to Equation (3), respectively:

$$C(x)=\frac{L_{cov}(x)}{L_{total}} \tag{1}$$

$$T(x)=\frac{1}{t(x)} \tag{2}$$

$$R(x)=\frac{1}{r(x)} \tag{3}$$

where $L_{cov}(x)$ is the number of lines of code covered by test case x; L_{total} is the total number of lines of code, and the ratio of the two is the code coverage $C(x)$; $t(x)$ is the execution time of test case x, and the inverse of the time is used to measure the execution efficiency $T(x)$, and the shorter the execution time of the test case, the better the fitness; $R(x)$ is the resource consumption of test case x, and the inverse of resource consumption $r(x)$ is also usually used to express the optimization goal. Based on this, the fitness function is shown in Equation (4):

$$f(x)=w_1\cdot C(x)+w_2\cdot T(x)+w_3\cdot R(x) \tag{4}$$

where w_1, w_2, and w_3 are weighting coefficients to adjust the influence of each objective in the fitness function. The weights are adjusted according to different testing needs.

III. D. Genetic manipulation realization

Genetic operations optimize and improve the quality of test cases. The implementation of genetic operations directly affects the diversity and adaptability of test cases, ensuring that the generated test cases can effectively cover more code paths. The specific genetic operation covers 3 steps: selection, crossover and mutation.

The selection operation is shown in equation (5):

$$P(x_i)=\frac{f(x_i)}{\sum_{j=1}^{N}f(x_j)} \tag{5}$$

where N is the number of individuals in the population; $f(x_j)$ is the fitness of the j th individual. The selection operation ensures that high-quality individuals have a higher probability of being passed on to the next generation.

The crossover operation is used to simulate genetic recombination in the biological inheritance process by generating new offspring individuals from the crossover of two parent individuals. Assuming that the two parent individuals are x_1 and x_2 and the cut point is located at position k, the crossover offspring individuals $x_{1'}$ and $x_{2'}$ are shown in Equation (6):

$$\begin{cases} x_{1'} = (x_1[1:k], x_2[k+1:N]) \\ x_{2'} = (x_2[1:k], x_1[k+1:N]) \end{cases} \tag{6}$$

Through crossover operation, new individuals can inherit the advantageous features of their parents and improve the overall fitness.

Mutation operation is used to randomly change the gene value of an individual, simulating the mutation process in biological evolution, increasing the diversity of the population, and avoiding the algorithm from falling into local optimality. Assuming that individual x is mutated at m, the mutated individual is shown in equation (7):

$$x' = \left(x[1:m-1], 1-x[m], x[m+1:N]\right) \tag{7}$$

The mutation rate is generally set to a small probability to ensure that variation occurs only in a small number of individuals, maintaining a balance between overall stability and diversity.

III. E. Evolutionary cycles

Assuming that the size of the population is N during the evolutionary process, the fitness of individuals increases generation by generation after the population has evolved for G generations. The average fitness of the population in the g th generation is shown in Equation (8):

$$\bar{f}_g = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{8}$$

In each generation, through crossover and mutation operations, individuals with higher fitness in the population are passed on to the next generation, while individuals with lower fitness are gradually eliminated.

IV. Automatic generation of test cases based on improved genetic algorithm

IV. A. Application of Improved Genetic Algorithm to Functions Based on

In order to verify the improvement effect of crossover probability and mutation probability proposed in this paper, the improved genetic algorithm proposed in this paper is compared with AGA algorithm [20] and IAGA algorithm. The function $z = \frac{1}{2} - \frac{\sin^2\sqrt{x^2+y^2} - \frac{1}{2}}{\left(1+0.001\times\left(x^2+y^2\right)\right)^2}$ is solved by applying the above three algorithms.

All three algorithms use binary coding, the selection method uses roulette, the crossover operation uses single-point crossover, and the mutation operation uses a simple mutation operation.In order to more intuitively show the settings of the parameters of the three different algorithms, the parameters are listed in this paper as shown in Table 1.

Table 1: Real number parameter setting

Parameter name	AGA	This article optimizes	IAGA
Chromosome length	18	18	18
Population size	90	90	90
Iteration number	300	300	300
K1	0.6	0.6	1
K2	0.9	1	0.6
K3	0.05	-	-
K4	0.110	-	
(x_min,x_max)	(-5,5)	(-5,5)	(-5,5)
(y_min,y_max)	(-5,5)	(-5,5)	(-5,5)

In the experiment, the above function was iterated 300 times in three different algorithms, and the difference between the maximum fitness function value ($f_{\max}$) and the average fitness function value ($f_{\min}$) was recorded during the iterations, as well as the time, number of iterations, and number of times of convergence of the above function after it had run fifty times in the three different algorithms.

Figure 1 shows the experimental results of the three algorithms for 300 iterations in the same operating environment. In the figure, it can be found that among the three algorithms, the AGA algorithm is the worst, in the case of 300 iterations, the value of the maximum fitness value minus the average fitness value has been fluctuating, and there is no tendency to converge, so it can be seen that the AGA algorithm is quite unstable in solving the complex quadratic function.The IAGA algorithm and the optimization algorithm of this paper have the same tendency of the final results, but the IAGA algorithm has a tendency to fall vertically between about 50 and 65 iterations. IAGA algorithm has been in a fluctuating state before about 40 iterations, and the convergence tends to fall vertically between about 50 and 65 iterations, but the maximum fitness value minus the average fitness value has converged to 0 at about 170 iterations, so it can be seen that the IAGA algorithm has a very good test result. In this paper, the optimization algorithm in the iteration of about 20 times to 55 times between the existence of violent fluctuations, but in the iteration of 100 times after the gentle tendency to decline towards convergence, can be found in the iteration of roughly 120 times the maximum fitness value minus the average fitness value of the results have been converged to zero.

From the figure, it can be intuitively seen in about 15 iterations of this paper optimization algorithm and IAGA algorithm has appeared different situations, this paper optimization algorithm's maximum fitness value minus the average fitness value has appeared a downward trend, the IAGA algorithm only appeared in about 55 iterations of the maximum fitness value minus the average fitness value decline in the case of the IAGA algorithm. Although the contrast occurs at about 115 iterations, the optimization algorithm of this paper still outperforms the IAGA algorithm in the subsequent iterations.

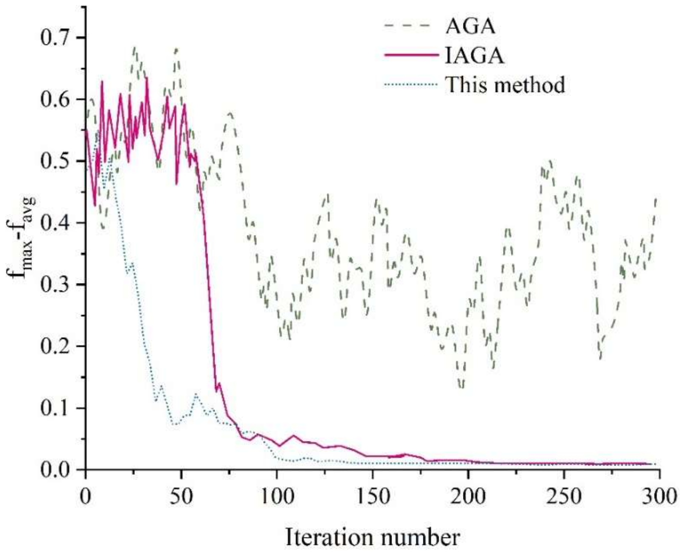


Figure 1: The f_{max}-f_{avg} values for the three algorithms after 300 iterations

To prove the general applicability of the algorithms, all three algorithms were made to execute 100 times and the experimental results of the 100 average experiments were recorded and found out as shown in Table 2, the significant values of the three algorithms in the case of 100 executions.

In the table, due to the limitation of the number of iterations, the AGA algorithm does not have a tendency to converge, so in the table the data of the AGA algorithm is recorded only the average time of execution. In the reality of the data in the table, it can be intuitively found that the IAGA algorithm is the slowest execution in the test, which has an execution time of about 55.84s, and the fastest is the optimization algorithm proposed in this paper, which has an execution time of 51.19s, which is 4.65s faster than the IAGA algorithm.In a three-dimensional graph, only the number of iterations is only in the case of 300 times, which is a very considerable value. In the case of the same number of iterations, it can be seen that the convergence speed of the optimization algorithm proposed in this paper is significantly better than that of the IAGA algorithm, and the optimization algorithm has already reached the convergence state after an average of 125.354 iterations, while the IAGA has to be iterated about 146 times.

Table 2: Test results of three algorithms

Genetic algorithm	Average time of execution /s	Average iteration times	Average convergence	Coverage ratio(%)
AGA	55.6822	-	-	-
This paper optimizes the algorithm	51.1944	125.354	50	100
IAGA	55.8367	146.574	50	100

From the above results, it can be shown that the automated test case generation method based on genetic algorithm is superior to the IAGA algorithm and also supports the feasibility and effectiveness of the algorithm.

IV. B. Experimental preparation

IV. B. 1) Experimental setup

The performance of the algorithms in this paper was tested above, and the following experiments will be used to validate the method proposed in the paper by using the commonly used experimental programs in the generation of seven test cases as the subjects. The constituent units of the experimental system include fitness function calculation module, program analysis module, similarity module, SAGA running module, etc. Comparative analysis shows that one of the most important of these modules, i.e., SAGA.

(1) Program Analysis Module

Static analysis of the program is the first step in generating test cases, mainly on the program's entire executable path through the program analysis module to complete the relevant analysis, and the use of control flow diagrams can be more intuitive to show the path. To determine the execution path of the program, the first step is to determine the control flow diagram, so that in the process of processing can be based on the program execution path analysis, so as to search for the target path.

(2) Realization module based on path similarity algorithm

The control flow diagram can be determined according to Soot, and then analyze the target path and execution path according to the result, calculate the path similarity according to the algorithm in the previous section, and then use the path representation method to represent the path processing, so as to satisfy the requirements of the subsequent analysis of the fitness function.

(3) Based on the calculation module under the fitness function

In the design of the entire tool in the fitness function calculation module is also very important, the fitness function of the construction, find the target path are through this module to complete, and the construction of each branch of the program, the branch function at the same time in the corresponding branch of the insertion of stakes, and then combined with the path similarity to calculate the fitness value of the individuals in the population.

(4) SAGA module

In this paper, the automated test case generation method based on genetic algorithm is operated in this module, first of all, the population is initialized, and the amount of data is obtained based on the analysis of the corresponding number of populations. Then, the program insertion module is used to analyze and obtain the relevant individual fitness.

IV. B. 2) Experimental objectives

The experiment takes seven programs under test as subjects to validate the method proposed in the paper. These programs under test are commonly used in test case generation and are representative. The main purpose of this part is to apply the algorithm proposed in this paper to the generation of test cases.

IV. B. 3) Experimental design

In the research process in order to verify the effectiveness and application value of the method of this paper, the comparative experimental method will be used. The control algorithm chosen for the study is the DTG, which uses the principle data flow model in generating test cases, and determines the fitness by analyzing it from the fitness function level, while introducing a genetic algorithm. A comparison was also made with the NDTG algorithm, which calculates the fitness by branching distance and also uses a genetic algorithm.

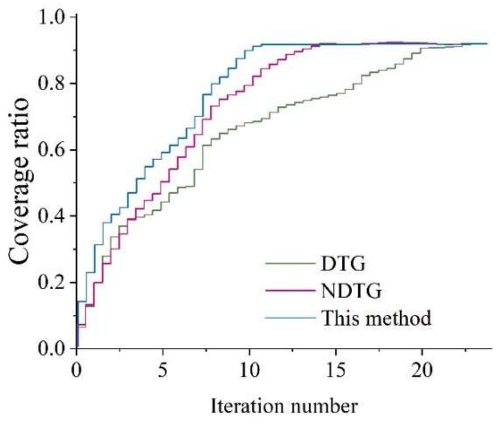
IV. C. Experimental results and analysis

(1) Comparison of DUP coverage

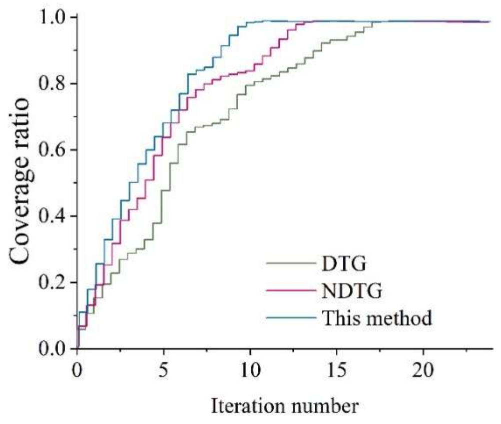
The results of the experiments are shown in Table 3, which specifically shows the variation of program coverage of each algorithm in generating test cases. The last group shows the average coverage of test cases obtained through the three algorithms, which are 73.6%, 77.8% and 82.5% respectively. Comparative analysis shows that in the actual testing process, the coverage rate of relatively simple programs under test is very high, and in some cases it can reach 97.6%, such as Monitoring, Energy and other programs.

Table 3: Coverage based on different methods

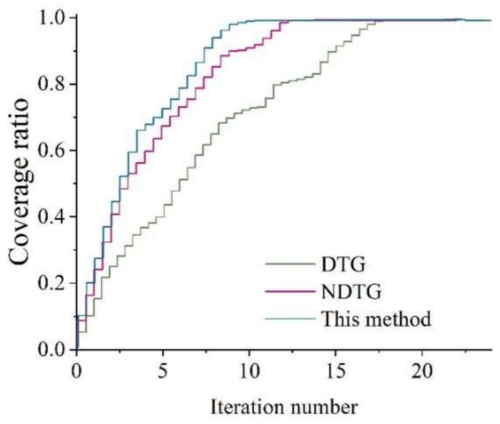
Programmed program	DTG	NDTG	SAGS
Test 1	97.6%	97.6%	97.6%
Test 2	97.6%	97.6%	97.6%
Test 3	97.6%	97.6%	97.6%
Test 4	71.2%	73.5%	75.3%
Test 5	81.4%	81.4%	84.8%
Test 6	50.2%	55.6%	60.5%
Test 7	47.4%	50.7%	54.2%
Test 8	73.6%	77.8%	82.5%



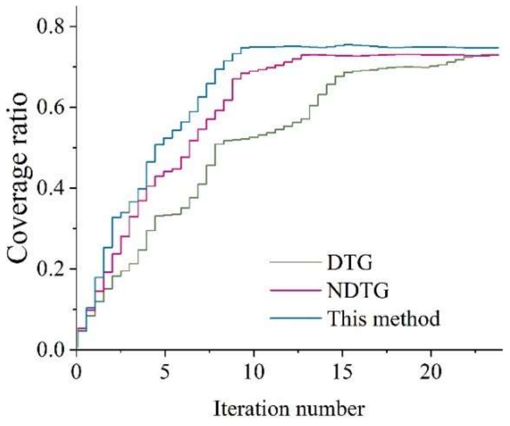
(a) Energy dissipation



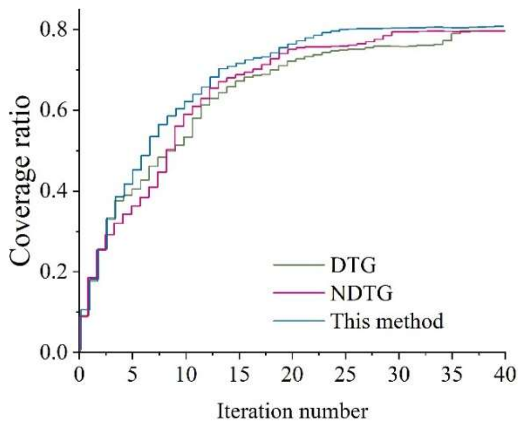
(b) Monitoring



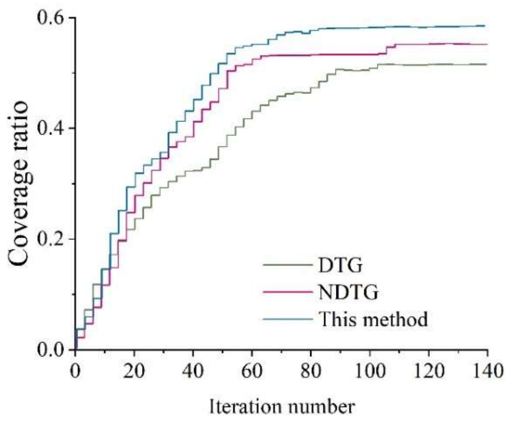
(c) Energy



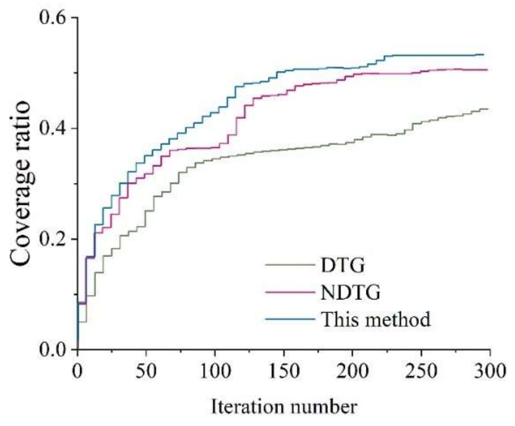
(d) Environment



(e) Product Details



(f) Max Min



(g) Triangle Classifier

Figure 2: Comparison of coverage

(2) Comparison of evolutionary generations

The corresponding number of iterations of the seven tested programs in the process of generating use cases is shown in Fig. 2, where the horizontal and vertical coordinates correspond to the number of iterations, coverage, respectively, where (a), (b), (c), (d), (e), (f), and (g) represent the tested programs Energy dissipation, Monitoring, Energy, Environment, respectively, Product Details, Max Min, and Triangle Classifier, respectively.

Comparative analysis of the results of this figure shows that under the condition of the same number of iterations, the coverage of SAGA is higher and the difference of results is more obvious. After a certain number of iterations, the coverage rate gradually becomes smooth, so it can be known that the reason for this situation is that the algorithm basically reaches a converged state, so that the coverage rate will basically remain unchanged after reaching a certain number of iterations at runtime. From the results obtained in the figure, it can also be found that the coverage of simple programs such as Monitoring and Energy is fast, and can be fully covered within 25 generations in general. Compared with other algorithms, the optimization algorithm in this paper can better meet the requirements of the application effect, and also play a role in promoting and supporting the convergence of the algorithm.

V. Conclusion

In this paper, through the in-depth analysis of the smart building networking system, we constructed a test case automatic generation framework based on the improved genetic algorithm, which effectively solves the problems of low coverage and low efficiency of the traditional testing methods. Experimental validation shows that the improved genetic algorithm can converge after an average of 125.354 iterations, which improves the convergence speed significantly compared with the 146.574 iterations of the IAGA algorithm. In the coverage test, the validation results for seven standard test programs show that the proposed method achieves an average coverage of 82.5%, which realizes a performance improvement of 8.9% and 4.7% compared to 73.6% for the DTG algorithm and 77.8% for the NDTG algorithm, respectively. In particular, the method shows stronger adaptability when dealing with complex programs, e.g., the coverage rates of the Test 6 and Test 7 programs reach 60.5% and 54.2%, respectively, which are significantly better than the comparison algorithms.

The improved design of the fitness function achieves a comprehensive assessment of test case quality by integrating the three dimensions of code coverage, execution efficiency and resource utilization. The optimized design of genetic operation enhances the global search capability and local optimization performance of the algorithm, and effectively avoids the premature convergence problem. The construction of multi-level system architecture provides flexible encoding for different types of test parameters, supporting both binary encoding for discrete parameters and floating-point encoding for continuous parameters. The method provides a strong technical support for the software quality assurance of intelligent building networking system, which has an important practical value for improving the reliability and stability of the system, and at the same time provides new ideas and references for the automated testing research in related fields.

References

[1] Arabasy, M., Hussein, M. F., Abu Osba, R., & Al Dweik, S. (2025). Smart housing: integrating machine learning in sustainable urban planning, interior design, and development. Asian Journal of Civil Engineering, 26(1), 59-71.

[2] AlFaris, F., Juaidi, A., & Manzano-Agugliaro, F. (2017). Intelligent homes' technologies to optimize the energy performance for the net zero energy home. Energy and Buildings, 153, 262-274.

[3] Zhong, R. Y., Peng, Y., Xue, F., Fang, J., Zou, W., Luo, H., ... & Huang, G. Q. (2017). Prefabricated construction enabled by the Internet-of-Things. Automation in Construction, 76, 59-70.

[4] Hildayanti, A., & Machrizzandi, M. S. R. (2020). The application of iot (internet of things) for smart housing environments and integrated ecosystems. Nature: National Academic Journal of Architecture, 7(1), 80-88.

[5] Ghosh, A., Edwards, D. J., & Hosseini, M. R. (2021). Patterns and trends in Internet of Things (IoT) research: future applications in the construction industry. Engineering, construction and architectural management, 28(2), 457-481.

[6] Winarno, A., & Affandi, M. (2022). Design and Construction of Smart House Prototype Based Internet of Things (Iot) Using Esp8266. BEST: Journal of Applied Electrical, Science, & Technology, 4(1), 11-14.

[7] Jia, M., Komeily, A., Wang, Y., & Srinivasan, R. S. (2019). Adopting Internet of Things for the development of smart buildings: A review of enabling technologies and applications. Automation in construction, 101, 111-126.

[8] Zhang, Y. Y., Kang, K., Lin, J. R., Zhang, J. P., & Zhang, Y. (2020). Building information modeling–based cyber-physical platform for building performance monitoring. International Journal of Distributed Sensor Networks, 16(2), 1550147720908170.

[9] Valinejadshoubi, M., Bagchi, A., & Moselhi, O. (2019). Development of a BIM-based data management system for structural health monitoring with application to modular buildings: Case study. Journal of Computing in Civil Engineering, 33(3), 05019003.

[10] Li, C. Z., Xue, F., Li, X., Hong, J., & Shen, G. Q. (2018). An Internet of Things-enabled BIM platform for on-site assembly services in prefabricated construction. Automation in construction, 89, 146-161.

[11] Chammas, M., Makhoul, A., & Demerjian, J. (2019). An efficient data model for energy prediction using wireless sensors. Computers & electrical engineering, 76, 249-257.

[12] Martín-Garín, A., Millán-García, J. A., Baïri, A., Millán-Medel, J., & Sala-Lizarraga, J. M. (2018). Environmental monitoring system based on an Open Source Platform and the Internet of Things for a building energy retrofit. Automation in Construction, 87, 201-214.

[13] Xie, D., & Xie, Q. (2024). Internet of things-based study on online monitoring system of building equipment energy saving optimization control using building information modeling. Science Progress, 107(2), 00368504241228130.

[14] Wang, J., Fu, Y., & Yang, X. (2017). An integrated system for building structural health monitoring and early warning based on an Internet of things approach. International Journal of Distributed Sensor Networks, 13(1), 1550147716689101.

[15] Cao, Y., Miraba, S., Rafiei, S., Ghabussi, A., Bokaei, F., Baharom, S., ... & Assilzadeh, H. (2020). Economic application of structural health monitoring and internet of things in efficiency of building information modeling. Smart Struct Syst, 26(5), 559-573.

[16] Dilakshan, S., Rathnasinghe, A. P., & Seneviratne, L. D. I. P. (2021, July). Potential of internet of things (IOT) in the construction industry. In Proceedings The 9th World Construction Symposium| July (p. 446).

[17] Havard, N., McGrath, S., Flanagan, C., & MacNamee, C. (2018, December). Smart building based on internet of things technology. In 2018 12th International conference on sensing technology (ICST) (pp. 278-281). IEEE.

[18] Shvets, Y., & Hanák, T. (2023). Use of the internet of things in the construction industry and facility management: usage examples overview. Procedia Computer Science, 219, 1670-1677.

[19] Zhengshu Chen,Yanqiu Cui,Hongbin Cai,Haichao Zheng,Qiao Ning & Xin Ding. (2025). Multi-objective optimization of photovoltaic facades in prefabricated academic buildings using transfer learning and genetic algorithms. Energy,328,136470-136470.

[20] Kaipu Wang,Xiaoyi Ma,Yibing Li,Yabo Luo,Yingli Li & Liang Gao. (2025). An adaptive genetic algorithm based on Q-learning for energy-efficient e-waste disassembly line balancing and rebalancing considering task failures. Journal of Manufacturing Systems,80,1-19.