©SHUTTERSTOCK.COM/DENIS BELITSKY

# Intelligent Generation of Test Cases for a Parallel Testing System: A Case Study on Railway Systems


**Yuantao Jiao, Jian Wang, and Runmei Li***
*School of Automation and Intelligence, Beijing Jiaotong University, Beijing 100044, China. E-mail: 22120232@bjtu.edu.cn, wangj@bjtu.edu.cn, and rmli@bjtu.edu.cn.*

**Fei-Yue Wang**
*National Key Laboratory for Multi-Modal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. E-mail: feiyue.wang@ia.ac.cn.*

**Hongtao Zhao**
*Signal and Communication Research Institute, China Academy of Railway Sciences, Beijing 100081, China. E-mail: zhaoht@rails.cn.*

**Gang Xiong**
*National Key Laboratory for Multi-Modal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. E-mail: gang.xiong@ia.ac.cn.*



*Digital Object Identifier 10.1109/MITS.2024.3435828*
*Date of publication 14 August 2024; date of current version 8 May 2025.*



*Corresponding author







1939-1390/24©2024IEEE

***Abstract*—Currently, the testing approach for complex systems relies on manual test case generation, resulting in problems, such as low efficiency and incomplete test case development. To address these challenges and improve the quality of test case generation for complex systems, this article proposes an intelligent method for generating test cases in parallel testing systems. Within the domain of railway systems, characterized by inherent intricacies, the centralized traffic control system serves as a pertinent example. This typical large and intricate railway transport system presents significant challenges, particularly in ensuring safety and conducting comprehensive functional testing. The design of the parallel testing system is detailed using the artificial systems, computational experiments, and parallel execution methodology, where an artificial system is built in a data-driven way that realistically replicates the test environment of a real system. Computational experiments were conducted on a test case library using the bidirectional encoder representations from transformers (BERT) model of natural language processing. BERT's next sentence prediction task was used for the associative learning of test case pairs. Finally, the test case intelligent generation software completes the parallel execution of the final testing task, which can intelligently generate the best relevant test cases based on the results of the computational experiments of the BERT model. This approach serves as a valuable tool for test engineers, enabling them to streamline test case formulations and enhance the efficiency of testing in complex systems.**

## Introduction

A parallel system consists of a real system and one or more software-defined artificial systems (virtual or ideal) that correspond to it [1]. The three components of a parallel system are artificial systems, computational experiments, and parallel execution (ACP) [2]. In recent years, the ACP approach-based parallel system theory has been widely applied in research on modeling, computations, and analyses of large and complex systems, such as intelligent transportation [3], [4], health care [5], and image generation [6].

A centralized traffic control (CTC) system is responsible for ensuring safe and efficient train operation in high-speed railway transport organizations. This is achieved by dynamically adjusting the stage plan and providing train scheduling services for the railway transportation system [7]. With the rapid expansion of high-speed rail operations and the iterative upgrade of existing equipment, CTC system structures have become more complex, and CTC system products continue to be optimized, leading to increasing testing requirements for CTC systems [8]. Efficient and continuous reliability verification and functional testing of CTC systems have become core issues in the development process [9].

In 2010, researchers applied parallel systems to test large-scale distributed systems and created a parallel testing methodology [10]. This method has been successfully used to test automatic driving systems [4], [12] and bus driving systems [13]. These studies generated test scenarios using actual system data. The powerful computational experiments and execution capabilities of parallel systems rely upon verifying the reliability of the system [14]. Dong et al. [15] initially achieved credibility assessment of the parallel test method for railways by modeling and generating scenarios for railway train operation control systems with artificial systems. Xu et al. [16] conducted a preliminary design of computational experiments on railway CTC systems using parallel testing methods. This approach effectively reduced the testing time and improved the efficiency.

At present, CTC system tests rely mainly on manual effort [17]. Most test cases lean toward routine testing and are limited by manual testing experience, resulting in uncertainty in the development of test cases. Therefore, extreme working condition test cases have not been well explored [18].

Facing the urgent demand for the development of test intelligence for CTCs, this article introduces the parallel test method for test case generation for CTC systems. The following innovative work is proposed, as shown in Figure 1:

1) An artificial CTC system is structured by modeling a virtual platform using computers and local area network (LAN)-based communication technologies. This artificial CTC system and the real CTC system can realize direct parallel interaction, which together constitute a parallel CTC system, solving the problem of test modeling for large and complex systems.
2) A framework for parallel test case generation using the bidirectional encoder representations from transformers (BERT) model is established. This approach addresses the computational experiment challenges of parallel CTC test cases. A test case library is formed by integrating historical sets and formalizing test personnel experience. Intelligent test case generation is achieved by utilizing natural language processing (NLP) and calculating the optimal probability through text similarity. This enhances the test case design efficiency and supports the optimization of the computational experiment of parallel test systems. The BERT model of NLP

Railway CTC systems have undergone several developmental stages, both domestically and internationally, due to the continuous modernization of railway transportation and advancements in technology.

is capable of autonomously learning the strategy design of test cases, thereby replacing the traditional test case design method based on expert experience with a test case library that can continuously learn and adapt, thus achieving the goal of intelligence.

3) Based on the real–artificial parallel system, test case intelligent generation software is developed, and the parallel execution module of the CTC system test system is constructed. This module can replace manual testing work, effectively avoiding repetitiveness and redundancy problems.

The article is structured as follows. The "CTC System Development and Parallel Testing System Construction" section discusses CTC system development and parallel testing, followed by the "Intelligent Generation of Test Cases for Parallel CTC Systems" section. The "Parallel Execution of CTC System Test Cases" section presents the parallel execution of CTC system test cases, and the final section concludes the article.

## CTC System Development and Parallel Testing System Construction

### *The Development of the Railway CTC System*

Railway CTC systems have undergone several developmental stages, both domestically and internationally, due to the continuous modernization of railway transportation and advancements in technology. For example, the Train à Grand Vitesse (TGV) Railway in France focuses on train dispatching and schedule management [19]. Germany has integrated dispatching into existing regional dispatching systems using a three-level management approach [20]. The COSMOS system was implemented on Japan's Shinkansen network.

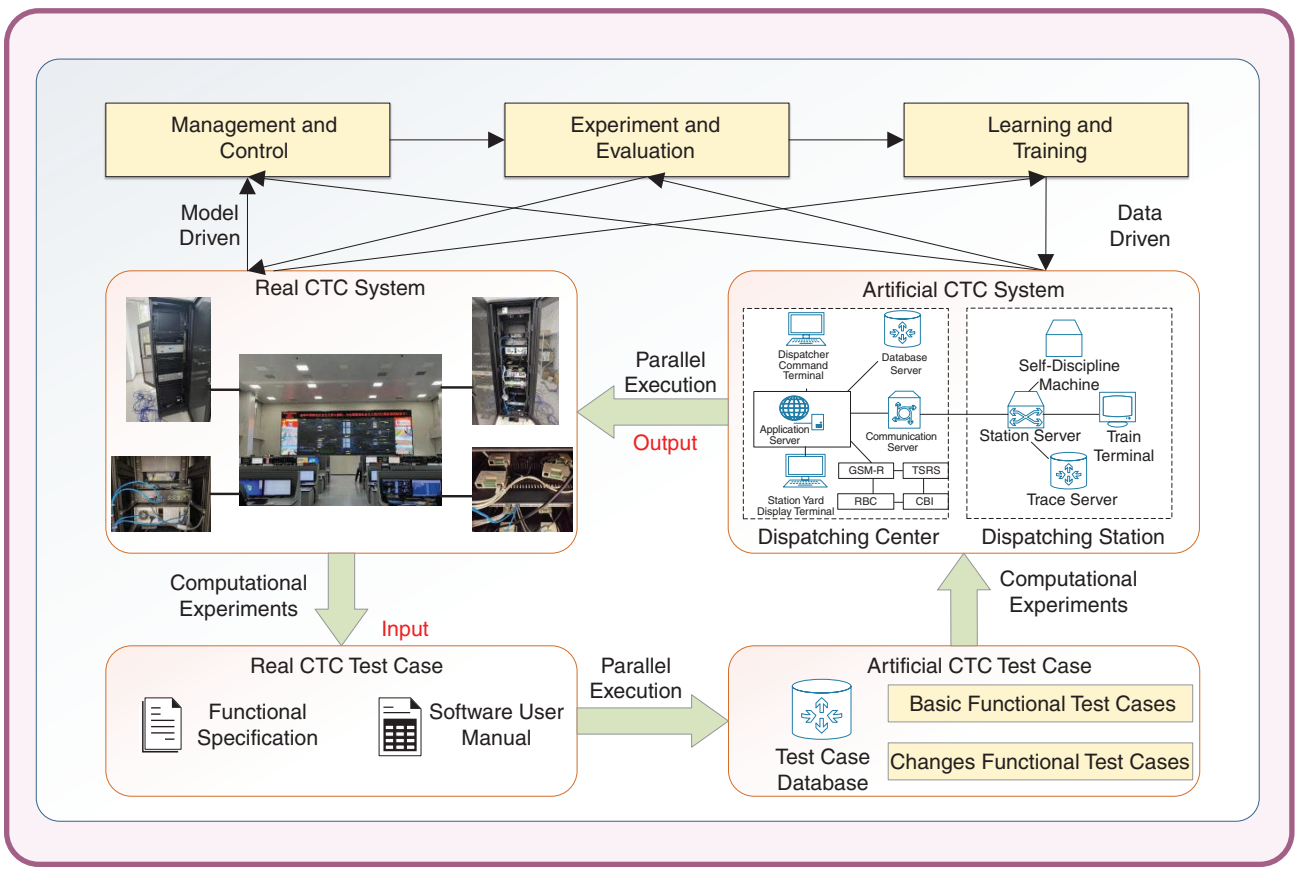



**FIG 1** Parallel CTC system test case generation framework.

A decentralized autonomous paradigm was established that considers the unique characteristics of the country's terrain and railway networks. The system ensures normal railway operations in the event of system failures through a wide-area decentralized autonomous system [21].

The development of the centralized dispatching system in China initially started in 1994, when the former Electricity Bureau of the Ministry of Railways proposed the construction of the Train Dispatch Command System. Subsequently, as the operating mileage of China's high-speed railway continued to expand, the Ministry of Railways formulated the Technical Specification for Decentralized Self-discipline CTC System, which comprehensively adopts the CTC system to implement centralized control of scheduling, to solve the contradiction between train adjustments and scheduling.

The CTC system in China was first introduced with the Japanese concept of a decentralized autonomous system. It was later optimized to create the CTC 1.0 system [22]. As railway construction in China progressed, the functions of the CTC system expanded, leading to the development of the CTC 2.0 system. In 2010, the Chinese Train Control System matured, leading to the expansion of interfaces with other signaling systems, such as the temporary speed restriction server (TSRS) and radio block center (RBC). Furthermore, integration with the Transportation Dispatch Management System and Dynamic Monitoring System was improved [23]. In 2015, the China Railway Corporation established new generation CTC technical conditions (Q/CR 518-2016), which defined the characteristics of the CTC system and marked the advent of the CTC 3.0 system [24].

System complexity increases the number of testing tasks. The testing of the CTC system is divided into several stages:

- *CTC unit testing* [25]: This stage relies on initial design documents to verify the correctness of basic CTC modules.
- *CTC integration testing* [26]: This verifies model reliability by connecting basic modules from unit testing and validating the data transfer between modules.
- *CTC functional testing* [27]: This verifies the functionality, performance, and security of the hardware and software environments on a complete CTC system.
- *CTC interface test* [28]: This test focuses on testing data exchange, transmission, and control management with other railway signaling systems and databases to verify reliability.
- *Regression testing* [29]: The final step reevaluates the modified version.

All testing activities rely on subsystems categorized into central and stationary parts, including human–machine interaction terminal software, interface software, and communication server software [30]. This article focuses on functional testing, specifically testing human–machine interaction terminals. The testing objects crucial to this study are the dispatch command terminal, which is used for drawing operation charts and issuing dispatch commands and train plans, and the train operation terminal, which is used by train service personnel to control managed stations through the station operation interface, as shown in Figure 2.

Testing the railway CTC system involves several steps, including determining the test scenarios, generating test cases, evaluating test case sets, simplifying the number of test cases, reproducing and verifying test tasks, and performing secondary verification of regression testing. The key aspect of testing is generating test cases based on determined testing scenarios [31].

### *Test Cases for the CTC System*

A test case is a combination of inputs, execution conditions, and expected results used to verify specific functionalities in a software system [32]. Test cases represent the beginning of system functionality testing and serve as the basis for performing computational experiments in parallel testing. Parallel testing involves steps designed to verify functional completeness between two or more systems. These test cases ensure consistency in

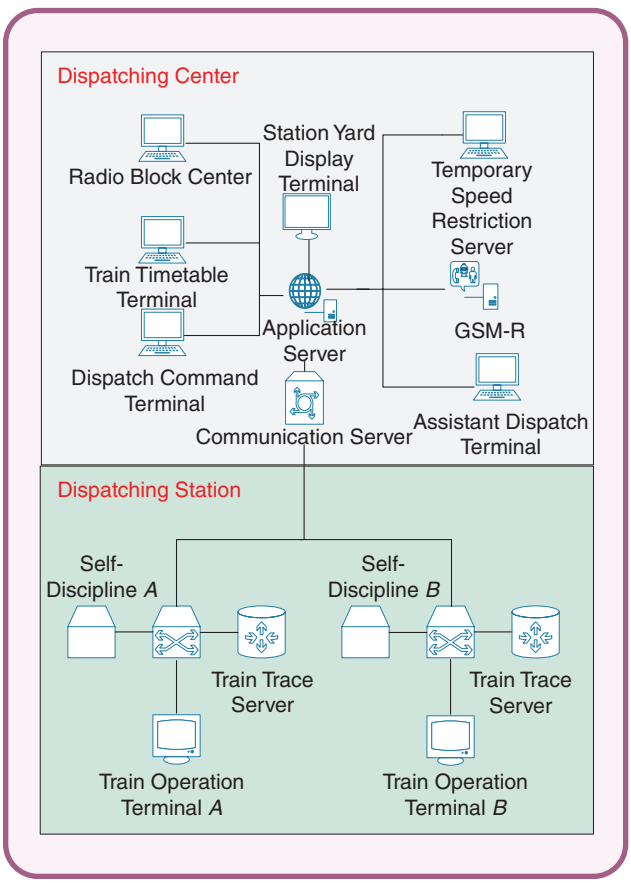



FIG 2 CTC software structure connection diagram.

This allows for testing the function of the CTC system in combination with the subsoftware introduced by the artificial system, which saves testing time and cost.

functional performance across different systems or versions and validate that the system remains stable after upgrades or changes [33].

Functional test cases for CTCs can be divided into two categories: basic functional test cases and change functional test cases. The basic functional test cases cover all of the functionalities that the CTC system should possess. Change functional test cases focus on optimizing and iterating the system based on the habits of the target user group and meeting user needs while addressing software bugs and defects identified during actual usage [34].

After setting up the testing environment in the railway dispatch center, test engineers verify both the basic and modified functions. The testing scenarios are defined by specifying the type and number of test cases for each scenario.

The written test cases are described in scenarios using natural language in Chinese. These include the functional scenario, test case number, test case input (operational steps), test case output (expected results), and test case result determination [11].

Table 1 describes the test cases for the dispatch command terminal of the CTC system, which mainly contains 11 types of modules, each of which contains several test cases, for a total of 1,593 entries. The source of these test cases is taken from the block "Real CTC Test Case" and the "Artificial CTC Test Case" in Figure 1. These test cases are fed as input data to the "Artificial CTC System" for the computational experiments block, as shown in Figure 3.

Table 1. Test case for the CTC dispatch command terminal.

| Module Name | No. of TCs |
|---|---|
| 1. System login | 23 |
| 2. Train timetable map drawing | 127 |
| 3. Adjustment chart function | 424 |
| 4. Day shift chart function | 42 |
| 5. Basic chart function | 32 |
| 6. Query desk | 80 |
| 7. Plan checking | 339 |
| 8. Comprehensive data query | 62 |
| 9. System performance | 20 |
| 10. System configuration | 215 |
| 11. Logging | 229 |
| Total | 1,593 |

### *Artificial CTC System*

The artificial CTC system introduces simulation terminal subsoftware to simulate actual railway train operation and dispatching. Figure 3 shows that the artificial CTC software comprises the dispatch command terminal, application server, train operation terminal, and subsoftware of the simulation terminal. This artificial CTC system provides a research platform for the test cases in the "Test Cases for the CTC System" section, performs computational experiments on the test cases by means of personnel agency, and builds a foundation for the parallel execution part of the subsequent sections. The software functions are as follows:

- The dispatch command terminal is responsible for creating and adjusting the train operation plan and dispatching commands. The system sends the information to the station dispatching office.
- The application server establishes the dual network structure consisting of the communication equipment and the transmission channel, and connects other subsoftware to complete the environment within the LAN of the laboratory to ensure the forwarding and transmission of the train dispatching information.
- The train operation terminal manages dispatching in the station area. It receives dispatching information from the dispatching center, analyzes the commands and sends them to the interlocking equipment at the station.
- Simulation terminal subsoftware is used to simulate the status of the actual railway operation, including information on the train stopping lanes, actual pick-up and departure times, and occupancy in the zones and stations.

The interface protocols of the artificial CTC system have also been expanded to include other systems in the CTCS, including simulation interfaces for the TSRS, GSM-R, and RBC. These expanded interface protocols are utilized to realize a comprehensive artificial system, providing an operational environment for subsequent testing tasks.

## Intelligent Generation of Test Cases for Parallel CTC Systems

### *A Framework for Generating Parallel Test Cases for CTC Systems*

Testing scenarios cannot always be executed in a real CTC system due railway safety regulation. Therefore, traversing

all possible paths on an actual physical system to complete comprehensive testing of train scheduling scenarios is often challenging. This section employs the ACP approach of the parallel system to study the data processing and association learning aspects of the test cases. The parallel CTC system test case generation framework is structured to utilize the NLP model for completing computational experiments of test cases. This allows for testing the function of the CTC system in combination with the subsoftware introduced by the artificial system, which saves testing time and cost.

The framework's core includes the artificial CTC system, which completely reproduces the real CTC system by simulating the train operation terminal, dispatch command terminal, application server, and simulation terminal; the CTC-BERT model is introduced, which, in combination with the historical test cases and tester experiences, learns from the real CTC test cases and feeds them back to the artificial CTC system. This process achieves predictive intelligence and improves test case efficiency, leading to more comprehensive parallel CTC system testing.

This process involved several steps, including analyzing the test case set of the actual site, user manual, dispatcher manual, CTC technical regulations, and other historical data to gain a deep understanding of the basic functional test cases and change iteration test cases, as shown in Figure 3. This figure corresponds to the "Artificial CTC System" block of Figure 1, which was built to solve the problem of the real CTC system's inability to run through all of the test cases. It facilitates the subsequent computational experiments and parallel execution of test cases in Figure 1.

These test cases are subsequently applied to the artificial CTC system created in the laboratory. Comprehensive functional tests are conducted to verify the system's ability to operate normally during historical test cases and to explore the completeness of extreme test scenarios, ensuring its functional reliability. This process tests both artificial systems and verifies the test tasks of the new version of the real system simultaneously. The historical test cases are used to complete the computational experiments on the test cases.

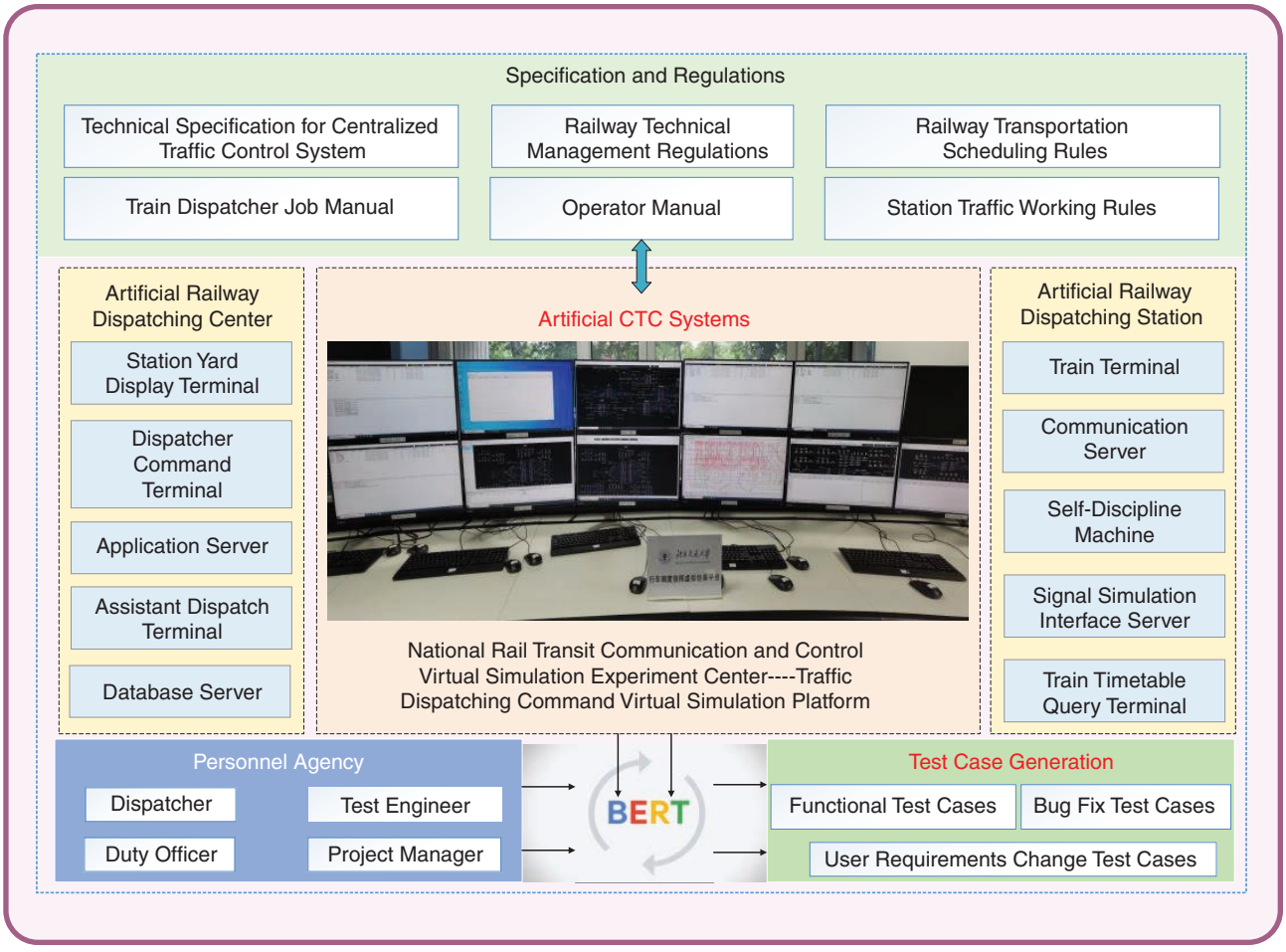



**FIG 3** Artificial CTC system and test cases.

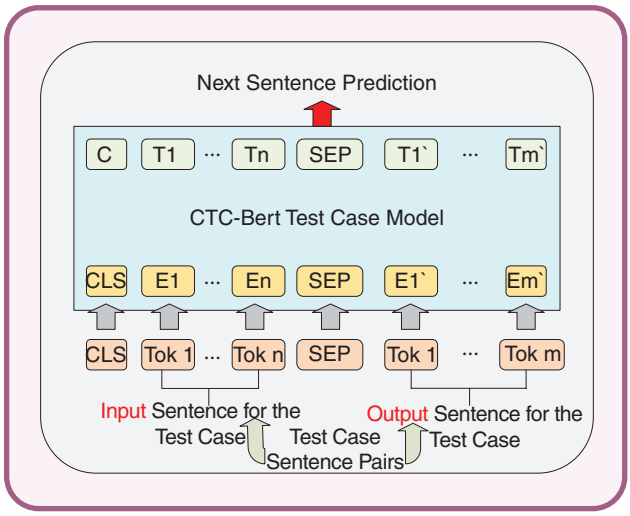



**FIG 4** CTC-BERT test case generation model.

### *CTC-BERT Model*

Processing test case data is crucial for linking parallel CTC system testing efforts. Currently, CTC system test cases use a textual sequence description paradigm. These test cases are stored as natural language in working documents, including single-step input and output, multiple operation steps, and expected results. These textual descriptions of test cases form the core of CTC system testing operations, and the focus is on how to extract and quantify these abstract test expert experiences.

BERT is a deep learning linguistic representation model. It has strong linguistic representation and feature extraction capabilities, making it particularly effective at handling long textual sequences.

The model addresses two downstream tasks that are based on the evolution of the masked language model (MLM) and next sentence prediction (NSP). The MLM randomly selects a percentage of a sentence as a token, replaces it with [MASK], and predicts the actual words of the masked [MASK] by learning their contextual content features through a classification model. The NSP learns relational features between sentences and predicts the relationship between two sentences in a pair. The purpose of NSP is to enable the pretrained language model to acquire intersentence information for logical reasoning, knowledge learning, and other subsequent tasks.

Figure 4 displays the CTC-BERT model, which is capable of parallel execution of test cases. The model serves as the fundamental component of Figure 5, facilitating the parallel execution of test cases by connecting the artificial CTC system with the real CTC system. The NSP task of BERT is used to perform sentence-to-text learning on real CTC test cases. Closed-loop feedback is employed to predict and generate new parallel test cases for the artificial CTC system. Upon completion of functional testing of the software on the artificial system, the software is fed back to the real CTC system twice to complete the parallel testing task. After functional testing of the software was conducted on the artificial system, the software was tested twice on the real CTC system to complete the parallel testing task.

### *Computational Experiments With the CTC-BERT Model*

This section models the test case library to incorporate the set of historical cases with formal testing experience into the model learning process. This, in turn, explores a method for intelligent test case generation to complete parallel execution, as shown in Figure 5. In this figure, the CTC-BERT model is used to process the real CTC test cases. The test case outputs from the model are then fed into the artificial CTC system for validation. Those test cases that pass the validation are fed back to the real CTC system to complete the testing mission of the real CTC system, which corresponds to the parallel execution block in the test case in Figure 1. The main steps of the BERT-based test case intelligent generation are as follows:

1) Standard test case assets are collected from various test scenarios to establish a test case library for each subsoftware of the CTC system.

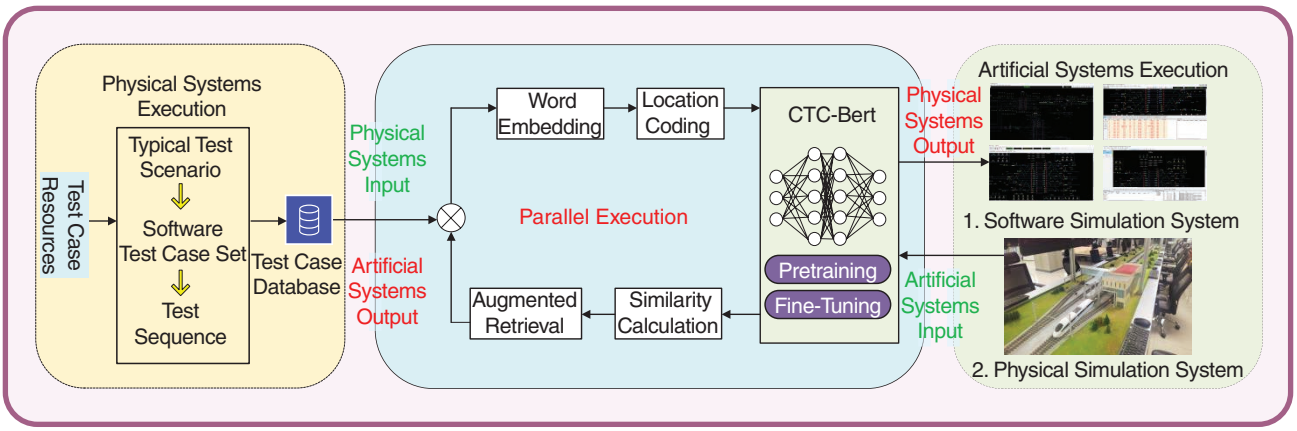



**FIG 5** Parallel test case generation framework based on CTC-BERT.

1) *Word embedding:* A multidimensional vector is used to map the input space X to the Y space. System-recognizable identifiers can be used to help the model continuously learn and train while increasing sequence correlation. The long text sequence is displayed as:

$$\begin{aligned}\text{Sents} = &<\text{CLS}> + \text{TCinput} + <\text{SEP}> + \text{TCoutput} \\ &+ <\text{EOS}> + <\text{PAD}> + <\text{PAD}> + \ldots\ldots \end{aligned} \tag{1}$$

The symbols used in this context are as follows: <CLS> marks the beginning of a sequence, <SEP> separates two test case sentences, <EOS> indicates the end of a sequence as judgment in termination, and <PAD> complements the characters that have not reached the set sentence length. $\text{TC}_{\text{input}}$ refers to the test case input text, and $\text{TC}_{\text{output}}$ refers to the test case output text.

2) *Positional encoding*: To ensure that the text sequence considers the contextual context and the position of each word within it, a positional encoding approach is employed to connect the contexts. The vocab.txt dictionary that accompanies the Chinese natural language model encodes the character-by-character position of all test case sets. As shown in Figure 6. The absolute position of the word in the utterance is represented using the sine-cosine function, and the product of the sine and cosine functions determines the relative position of the word in the utterance. The formula is as follows:

$$\text{PE}_{(\text{pos},2i)} = \sin\left(\frac{\text{pos}}{10000^{\text{li}/d_{\text{model}}}}\right) \tag{2}$$

$$\text{PE}_{(\text{pos},2i+1)} = \cos\left(\frac{\text{pos}}{10000^{\text{li}/d_{\text{model}}}}\right) \tag{3}$$

where $d_{\text{model}}$ represents the position of the word vector encoding, $i$ refers to the $i$th dimension of the word vector, and pos represents the position of the current word in the sequence.

2) *Construction of the CTC-BERT pretraining model.* The pretraining stage is used to acquire knowledge of natural language models from the test case data sequences constructed in the prequel step. The pretraining effect is further enhanced by combining an improved attention mechanism and a bidirectional long short-term memory network (Bi-LSTM).

3) *Attention mechanism*: This mechanism extracts relevant information from the input sequence through a three-stage calculation process. In the first stage, the correlation between inputs (including itself) is calculated based on the query vector query and each correlation vector key. The second stage involves the softmax normalization process, which transforms all the output values into a sum of 1 probability. Finally, the individual feature vector values are weighted and summed according to the attention weight coefficients to focus on the local features of the vector. The relationship between $Q$, $K$, and $V$ is as follows: this mechanism is the core of dealing with test cases in pairs of sequential text. The model is trained to output the best and most matching corresponding test cases.

$$\text{Attention}(Q,K,V) = \text{soft}\max\left(\frac{Qk^T}{\sqrt{d_k}}\right)V \tag{4}$$

$$a_i = \text{Soft}\max(\text{Sim}_i) = \frac{e^{\text{Sim}_i}}{\Sigma_{j=1}^{Lx} e^{\text{Sim}_j}} \tag{5}$$

$$\text{Attention}(Q,\text{Source})\Sigma_{i=1}^{Lx} a_i \times \text{Value}_i \tag{6}$$

where $Q$, $K$, and $V$ denote the query, key, and value vectors, respectively, and $\sqrt{d_k}$ is the vector dimension.

4) Improved multihead attention mechanism: The multihead attention mechanism divides the transformer model into multiple subspaces, each of which can focus on learning the necessary information. $Q$, $K$, and $V$ are projected through $h$ different linear transformations, where $h$ is the number of heads in the model, and the results of different attention mechanisms are combined. The formula is as follows:

$$\text{Multi}(Q,k,v) = \text{Con}(\text{head}_1,\ldots,\text{head}_h)W^0 \tag{7}$$

$$\text{head}_1 = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \tag{8}$$

| sents | labels |
|---|---|
| ('输入错误代码或密码点击确认按钮，进行交接班操作', '提示“用户代号或密码错误，请重新输入”') | 1 |
| ('行调台下达通过计划', '通过计划已下达') | 1 |
| ('删除唯一管理员用户', '提示“用户代号或密码错误，请重新输入”') | 0 |
| ('在新增界面确认密码与密码不一致', '通过计划已下达') | 0 |

| label | sent |
|---|---|
| 1 | [1, 6777, 1051, 7225, 6422, 801, 4766, 2766, 2160, 4766, 4151, 1134, 4796, 6365, 2896, 7169, 8018, 6816, 6115, 763, 2964, 4402, |
| 1 | [1, 6115, 6438, 1372, 672, 6803, 6852, 6808, 6363, 1147, 2, 1, 6852, 6808, 6363, 1147, 2341, 672, 6803, 2, 0, 0, 0, 0, 0, 0, 0, |
| 0 | [1, 1154, 7364, 1540, 665, 5046, 4409, 1441, 4494, 2781, 2, 1, 2984, 4844, 4494, 2781, 801, 1378, 2766, 2160, 4766, 7225, 6422, |
| 0 | [1, 1756, 3167, 1866, 4512, 7475, 4796, 6365, 2160, 4766, 674, 2160, 4766, 673, 665, 5630, 2, 1, 6852, 6808, 6363, 1147, 2341, |

FIG 6 CTC Chinese test case set and positional coding.

The enhanced BERT model no longer requires the outputs of $K$ and $V$. Instead, it outputs the highest score after passing through the linear and softmax layers. The improved BERT model is encoded with the BERT large model, which has 24 encoder layers, significantly more than the six layers of the ordinary transformer model. Using more encoders within a cost-controllable range can lead to better training results.

5) Bi-LSTM is a type of recurrent neural network that considers both past and future information of input sequences at each time step. This approach improves the model's ability to capture long-term dependencies. A simple expression of Bi-LSTM can be obtained by splicing forward and backward hidden states.

$$h_t^{\text{Bilstm}} = [h_t^{\text{forward}}; h_t^{\text{backward}}] \tag{9}$$

where $h_t^{\text{Bilstm}}$ represents the output of the bidirectional LSTM at time step $t$, $h_t^{\text{forward}}$ represents the hidden state of the forward LSTM at time step t, and $h_t^{\text{backward}}$ represents the hidden state of the reverse LSTM at time step $t$. The semicolon denotes the splicing operation, which joins the forward and reverse hidden states at a particular time step by dimension to form a richer representation that captures past and future contextual information.

3) *Test case generation*. Test cases are generated intelligently using pretraining results from the previous CTC-BERT model. The tester is provided with test case inputs for expected operations, and the test case output sequence for a given scenario is generated after sufficiently learning the BERT model from external a priori knowledge and similarity computations of the test text. This process improves the test case generation process.

Table 2. Test case for CTC train operation terminal.

| Module Name | No. of TCs |
|---|---|
| 1. User management function test | 21 |
| 2. Functional test of reporting form interface | 125 |
| 3. Functional test of dispatching command management | 212 |
| 4. Station display function test | 113 |
| 5. Functional test of traveling voucher | 69 |
| 6. Train management functionality test | 70 |
| 7. Functional test of confirm query | 88 |
| 8. Control command function test | 119 |
| 9. Approach sequence function test | 289 |
| 10. Shunting operation order function test | 12 |
| 11. Dual-machine synchronization function test | 22 |
| 12. Voice alarm function test | 9 |
| Total | 1,149 |

## Parallel Execution of CTC System Test Cases

This section demonstrates an experiment on the parallel execution of test cases by applying a parallel CTC system on the Guizhou High-Speed Railway. The Guikai Intercity Railway is an important part of the Guiyang municipal express railway network of "one ring, one shot, and two links" and is also the first intercity passenger dedicated line in Guizhou Province in China. The case study focuses on Guiyang 1 of the Guiyang-Kaiyang Intercity Railway, which comprises six stations from Guiyang East Station to Kaiyang Station. The test case study is based on this example.

To generate parallel CTC system test cases, the laboratory creates an artificial CTC system environment and selects several stations in Guiyang 1 for core testing, including train reception and dispatch.

In this section, we first define the test objects: the dispatch command terminal software and the train operation terminal software, which operate in two main subsoftware. The basic functional test cases are taken from the "Artificial CTC Test Case" block in Figure 1. Table 1 corresponds to the dispatch command terminal, which has 11 modules, and each module contains several test cases, with a total of 1,593 items. Table 2 corresponds to the test cases of the train operation terminal; there are 12 modules and each module corresponds to different functional test scenarios, for a total of 1,149 test cases. All of the test cases are rich in textual data with expert testing experience, describing how to test the CTC system. The total number of test cases in Tables 1 and 2 were converted into long text pairs recognizable by the BERT model, and there were more than 8,000 test case sentence pairs, which were used as input data for the BERT model to execute the test cases in parallel execution.

The experiment's code was written in Python and divided into three subtasks: data preparation, model training, and test case generation. These steps corresponded to the preprocessing of test data, pretraining of the model, and generation for test cases.

### *Data Preparation*

More than 8,000 test cases of Chinese version were extracted and integrated into the test sequence. The natural language sequences of the test input and output sentence pairs were merged. Sentence pairs that were relevant were marked with 1, while irrelevant pairs were marked with 0. The sentence pairs were then encoded character by character using word embedding and positional encoding. Figure 6 displays the integration method for input and output sentence pairs in the Chinese version test case, including the marking of identifiers and the positional coding of the integrated long Chinese sentences. Table 3 displays the English version of the test cases corresponding to the Chinese version of Figure 6.

### *Model Training*

After processing the test case data, the BERT model was used to address the issue of contextual key information. This was accomplished by separating the encoder from the transformer model. The BERT model encodes the word embedding and the position of the text through an attention mechanism to test the logical relationship between the input and the output.

> The purpose of NSP is to enable the pretrained language model to acquire intersentence information for logical reasoning, knowledge learning, and other subsequent tasks.

The model was trained using the BERT comparison experiment, which involved training four different BERT models: the ordinary BERT model, the BERT model with an added LSTM layer neural network, the BERT model with an improved attention mechanism, and the BERT model with an improved attention mechanism of Bi-LSTM. The models were evaluated based on their accuracy and loss function to compare their advantages and disadvantages. The text compares the advantages and disadvantages of different models through experiments.

The evaluation indices are calculated based on the comparison experiments of the four models. The loss function measures the difference between the predicted and real results of the model, while the accuracy rate refers to the proportion of correctly predicted samples in the model to the total number of samples. Its calculation formula is as follows:

$$L_{\text{nsp}} = -\frac{1}{N}\Sigma_i y_i \log P(y_i) + (1 - y_i)\log(1 - P(y_i)) \quad (10)$$

$$\text{Acc} = \frac{N_{\text{correct}}}{N_{\text{total}}}. \quad (11)$$

In the loss function, $N$ is the number of training samples, $P(y_i)$, is the probability that the model predicts $y_i$ to be adjacent; in the accuracy, $N_{\text{correct}}$ is the number of test sequences correctly predicted, and $N_{\text{total}}$ is the total number of samples.

Figure 7 demonstrates that the BERT model of the Bi-LSTM improved attention mechanism is the most effective among the four comparative experiments. The subsequent code in the "Test Case Generation" section will use this model for parallel test case generation for predictive software development.

Table 3. English test cases corresponding to Figure 6.

| Sentence Pair | Labels |
|---|---|
| "Enter error code for handover", "Error, please re-enter" | 1(relevant) |
| "Train schedules are issued by the Dispatch Command Terminal", "Enter the user management interface" | 0(irrelevant) |
| ..... | ..... |

### *Test Case Generation*

PyQt5 was used in the PyCharm compiler software for software development. The software supports the prediction

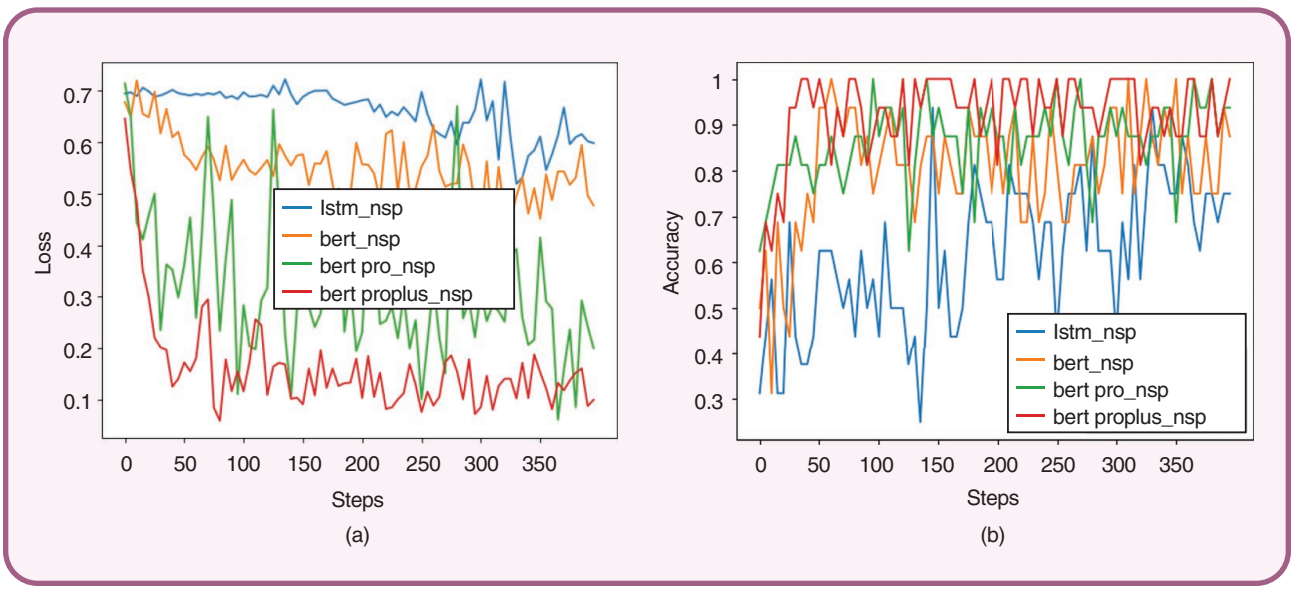



FIG 7 The CTC-BERT model test case generation effect diagram.

of parallel CTC system test case generation and provides five optimal output results for testers to choose from. It also has the ability to add, delete, change, and check test cases, which can improve testing efficiency and reduce testing time for subsequent parallel CTC testing.

This tool has been proven in real-world test projects and can effectively improve test case generation.

## Conclusion

This article presents the ACP method-based parallel system theory for modeling and testing large and complex railway systems in CTC. The BERT model is used for intelligent test case generation. The virtual train dispatching and command platform of the CTC is modeled based on LAN communication technology, and an artificial system is structured. Using the BERT model and incorporating historical test case data and formal testers' experience, we thoroughly examine extreme test scenarios. This enables us to establish a comprehensive test case library and continuously optimize the computational experimental aspects of test cases. Software for intelligent test case generation has been developed to assist testers in conducting more efficient functional tests and completing parallel execution of test cases under an on-site manual parallel CTC system environment. This is based on the results of CTC-BERT's computational experiments.

Parallel CTC systems can generate test cases for functional testing with high accuracy, full coverage, and greater speed than traditional manual testing. This approach is particularly effective for testing extreme chance scenarios. The proposed method can be extended to test case generation tasks for complex systems in other fields. This extension supports the intelligent generation of parallel test cases for test case reports, product functional specifications, and other data in different fields. This approach effectively improves testing efficiency and guarantees testing accuracy and high coverage.

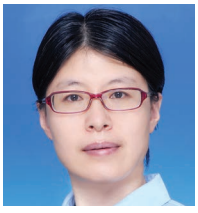
## Acknowledgment

The research in the article was jointly supported by the Technological R&D Program of China State Railway Group Co., Ltd (L2022X002), National Key R&D Program of China (2022YFB4300500), and the Beijing Natural Science Foundation (L211004).

## About the Authors

***Yuantao Jiao*** (22120232@bjtu.edu.cn) earned his B.S. degree in rail transportation signal and control from Dalian Jiaotong University, Dalian, China. He is working toward an M.S. degree in the School of Automation and Intelligence, Beijing Jiaotong University, Beijing 100044, China. His research interests include intelligent transportation systems, intelligent testing method, and parallel intelligence.

***Jian Wang*** (wangj@bjtu.edu.cn) earned his Ph.D. degree in traffic information engineering and control from Beijing Jiaotong University, Beijing, China in 2007. He is a professor and the vice dean of the School of Automation and Intelligence, Beijing Jiaotong University, Beijing 100044, China, where he is also a member of the Beijing Engineering Research Center of EMC and GNSS Technology for Rail Transportation. His research interests include satellite navigation, railway signaling, train control, information security, and intelligent transportation systems. He is a Member of IEEE.



***Runmei Li*** (rmli@bjtu.edu.cn) earned her Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China. She is a professor and the vice dean of the School of Automation and Intelligence, Beijing Jiaotong University, Beijing 100044, China. Her research interests include cooperative vehicle-road system, traffic data mining, and type-2 fuzzy sets theory and application.

***Fei-Yue Wang*** (feiyue.wang@ia.ac.cn) earned his Ph.D. degree in computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA. He is with the National Key Laboratory for Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. His current research focuses on methods and applications for parallel intelligence, social computing, and knowledge automation. He is the president of the IEEE Council on RFID and the vice-president of the IEEE Systems, Man, and Cybernetics Society. He is a Fellow of IEEE.

***Hongtao Zhao*** (zhaoht@rails.cn) earned his Ph.D. degree in traffic information engineering and control from the China Academy of Railway Sciences. He is an associate researcher with the Signal and Communication Research Institute, China Academy of Railway Sciences, Beijing 100081, China. His research interests include intelligent train scheduling and collaborative control of trains in complex scenarios.

***Gang Xiong*** (gang.xiong@ia.ac.cn) earned his Ph.D. degree in control science and engineering from Shanghai Jiao Tong University, Shanghai, China. He is a research scientist with the State Key Laboratory of Management and Control for Complex Systems, Beijing Engineering Research Center for Intelligent Systems and Technology, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China. He is also the Deputy Director of the Beijing Engineering Research Center for Intelligent Systems and Technology and the Cloud Computing Center, Chinese Academy of Sciences, Beijing 100190, China. His research interests include parallel control and management, modeling and optimization of complex systems, cloud computing and big data, intelligent manufacturing, and intelligent transportation systems. He is a Senior Member of IEEE.

## References


[1] F.-Y. Wang, "Parallel systems for control and management of complex systems," *J. Control Decis.*, vol. 19, no. 5, pp. 485–489, 2004.

[2] J. Chen, Y. Zhang, S. Teng, Y. Chen, H. Zhang, and F.-Y. Wang, "ACP-based energy-efficient schemes for sustainable intelligent transportation systems," *IEEE Trans. Intell. Veh.*, vol. 8, no. 5, pp. 3224–3227, May 2023, doi: 10.1109/TIV.2023.3269527.

[3] T. Liu, Y. Xing, X. Tang, H. Wang, H. Yu, and F.-Y. Wang, "Cyber-physical-social system for parallel driving: From Concept to Application," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 1, pp. 59–69, Spring 2021, doi: 10.1109/MITS.2020.3014079.

[4] Y. Lv, Y. Chen, L. Li, and F.-Y. Wang, "Generative adversarial networks for parallel transportation systems," *IEEE Intell. Transp. Syst. Mag.*, vol. 10, no. 3, pp. 4–10, Fall 2018, doi: 10.1109/MITS.2018.2842249.

[5] J. Wang, X. Wang, Y. Guo, and F.-Y. Wang, "A parallel medical diagnostic and treatment system for chronic diseases," in *Proc. Chin. Automat. Congr. (CAC)*, Shanghai, China, 2020, pp. 7412–7416, doi: 10.1109/CAC51589.2020.9327701.

[6] T. Shen et al., "Parallel medical imaging: An ACP-based approach for intelligent medical image recognition with small samples," in *Proc. IEEE 1st Int. Conf. Digit. Twins Parallel Intell. (DTPI)*, Beijing, China, 2021, pp. 226–229, doi: 10.1109/DTPI52967.2021.9540120.

[7] Y. Sun, S. Ding, Z. Li, Y. Ren, K. Sheng, and Y. Yang, "Research on human reliability of the high-speed railway intelligent dispatching centralized traffic control system," in *Proc. IEEE 7th Int. Conf. Intell. Transp. Eng. (ICITE)*, Beijing, China, 2022, pp. 111–116, doi: 10.1109/ICITE56321.2022.10101442.

[8] G. Feng and X. Wei, "Research on function test method of intelligent high-speed railway centralized traffic control system," in *Proc. 36th Youth Acad. Annu. Conf. Chin. Assoc. Automat. (YAC)*, Nanchang, China, 2021, pp. 104–108, doi: 10.1109/YAC53711.2021.9486513.

[9] F. Zhu, S. Li, F. Li, and Y. Cheng, "Test case study of high-speed railway train control system for typical operation scenarios," in *Proc. Australian New Zealand Control Conf. (ANZCC)*, Gold Coast, Australia, 2022, pp. 161–165, doi: 10.1109/ANZCC56036.2022.9966951.

[10] F.-Y. Wang, "Toward a revolution in transportation operations: AI for complex systems," *IEEE Intell. Syst.*, vol. 23, no. 6, pp. 8–13, Nov./Dec. 2008.

[11] C. Wang, F. Pastore, A. Goknil, and L. C. Briand, "Automatic generation of acceptance test cases from use case specifications: An NLP-based approach," *IEEE Trans. Softw. Eng.*, vol. 48, no. 2, pp. 585–616, Feb. 2022, doi: 10.1109/TSE.2020.2998503.

[12] K. Liu, L. Li, Y. Lv, D. Cao, Z. Liu, and L. Chen, "Parallel intelligence for smart mobility in cyberphysical social system-defined metaverses: A report on the international parallel driving alliance," *IEEE Intell. Transp. Syst. Mag.*, vol. 14, no. 6, pp. 18–25, Nov./Dec. 2022, doi: 10.1109/MITS.2022.3202825.

[13] X. Dong et al., "A parallel transportation management and control system for bus rapid transit using the ACP approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2569–2574, Sep. 2017, doi: 10.1109/TITS.2016.2645783.

[14] L. Vlacic, "The 2009-2020 Chinese Program on self driving and parallel testing [Editor's Column]," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 2, pp. 3–247, Summer 2021, doi: 10.1109/MITS.2021.3065863.

[15] H. Dong et al., "Parallel intelligent systems for integrated high-speed railway operation control and dynamic scheduling," *IEEE Trans. Cybern.*, vol. 48, no. 12, pp. 3381–3389, Dec. 2018, doi: 10.1109/TCYB.2018.2852772.

[16] W. Xu et al., "Parallel testing for centralized traffic control systems of intelligent railways," *IEEE Trans. Intell. Veh.*, vol. 8, no. 9, pp. 4249–4262, Sep. 2023, doi: 10.1109/TIV.2023.3305543.

[17] J. Zhang, "Research on CTC simulation technology of high-speed railway," *Railway Signalling Commun. Eng.*, vol. 16, pp. 14–17, Feb. 2019.

[18] Z. Li, W. Xu, T. Zhang, T. Wang, C. Miao, and J. Duan, "Research and application of automated test for background program of railway CTC system," in *Proc. China Automat. Congr. (CAC)*, Beijing, China, 2021, pp. 1975–1978, doi: 10.1109/CAC53003.2021.9727320.

[19] R. Tang et al., "A literature review of artificial intelligence applications in railway systems," *Transp. Res. C, Emerg. Technol.*, vol. 140, Jul. 2022, Art. no. 103679, doi: 10.1016/j.trc.2022.103679.

[20] X. Dai et al., "Dynamic scheduling, operation control and their integration in high-speed railways: A review of recent research," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 13,994–14,010, Sep. 2022, doi: 10.1109/TITS.2021.3131202.

[21] F. Kitahara, K. Kera, and K. Bekki, "Autonomous decentralized traffic management system," in *Proc. Int. Workshop Auton. Decentralized Syst. (Cat. No. 00EX449)*, Chengdu, China, 2000, pp. 87–91, doi: 10.1109/IWADS.2000.880891.

[22] W. Xu, N. Zhang, and T. Wang, "Design and realization of a TDCS/CTC-based train reception and departure operation control-management integrated system," in *Proc. Chinese Automat. Congr. (CAC)*, Jinan, China, 2017, pp. 6311–6317, doi: 10.1109/CAC.2017.8243915.

[23] H. Dong B. Ning, B. Cai, and Z. Hou, "Automatic train control system development and simulation for high-speed railways," *IEEE Circuits Syst. Mag.*, vol. 10, no. 2, pp. 6–18, 2nd Quart. 2010, doi: 10.1109/MCAS.2010.936782.

[24] J. Lv, W. Lu, T. Wang, and B. Wei, "The search-based mutation testing of the Chinese train control system level 3 on board a train control system," *IEEE Intell. Transp. Syst. Mag.*, vol. 14, no. 5, pp. 41–58, Sep./Oct. 2022, doi: 10.1109/MITS.2021.3069900.

[25] H. Felbinger, F. Wotawa, and M. Nica, "Adapting unit tests by generating combinatorial test data," in *Proc. IEEE Int. Conf. Softw. Testing, Verification Validation Workshops (ICSTW)*, Västerås, Sweden, 2018, pp. 352–355, doi: 10.1109/ICSTW.2018.00072.

[26] A. Sharif, D. Marijan, and M. Liaaen, "DeepOrder: Deep learning for test case prioritization in continuous integration testing," in *Proc. IEEE Int. Conf. Softw. Maintenance Evolution (ICSME)*, Luxembourg, Germany, 2021, pp. 525–534, doi: 10.1109/ICSME52107.2021.00053.

[27] H. Zheng, J. Feng, W. Miao, and G. Pu, "Generating test cases from requirements: A case study in railway control system domain," in *Proc. Int. Symp. Theor. Aspects Softw. Eng. (TASE)*, Shanghai, China, 2021, pp. 183–190, doi: 10.1109/TASE52547.2021.00029.

[28] J.-G. Hwang, J.-H. Baek, H.-J. Jo, and K.-M. Lee, "Software black-box testing tool for railway signaling system by real interface," in *Proc. 13th Int. Conf. Control, Automat. Syst. (ICCAS)*, Gwangju, Korea (South), 2013, pp. 508–511.

[29] M. H. Alkawaz and A. Silvarajoo, "A survey on test case prioritization and optimization techniques in software regression testing," in *Proc. IEEE 7th Conf. Syst., Process Control (ICSPC)*, Melaka, Malaysia, 2019, pp. 59–64, doi: 10.1109/ICSPC47137.2019.9068003.

[30] W. Xu, "Design and implementation of FZy-CTC dispatching command release system for Han Huang line of Beijing Railway Bureau," in *Proc. IEEE Int. Conf. Service Operations Logistics, Inform. (SOLI)*, Beijing, China, 2016, pp. 261–266, doi: 10.1109/SOLI.2016.7551698.

[31] X. Wei, Z. Tao, S. Pengfei, W. Tao, L. Zhi, and G. Feng, "Design and Implementation of Key Data Automatic Test system for Decentralized Autonomous Computer," in *Proc. China Automat. Congr. (CAC)*, Beijing, China, 2021, pp. 8375–8379, doi: 10.1109/CAC53003.2021.9727332.

[32] S. Wang, Q. Shang, Z. Ling, D. Liu, and X. Chen, "A method of automatic test case generation based on CT-LSSVM algorithm in FAO systems," in *Proc. IEEE 5th Int. Conf. Intell. Transp. Eng. (ICITE)*, Beijing, China, 2020, pp. 259–264, doi: 10.1109/ICITE50838.2020.9231435.

[33] S. Wang, Q. Shang, Q. Fang, and F. Fang, "Research on automated testing method of railway signaling system," in *Proc. IEEE 5th Int. Conf. Intell. Transp. Eng. (ICITE)*, Beijing, China, 2020, pp. 165–169, doi: 10.1109/ICITE50838.2020.9231379.

[34] K. Chen, J. Lv, Z. Luo, T. Tang, and S. Gao, "Complete test suites generation of CTCS -3 target speed monitor based on combinatorial testing," in *Proc. Chin. Control Conf. (CCC)*, Guangzhou, China, 2019, pp. 7126–7131, doi: 10.23919/ChiCC.2019.8866000.


ITS