The Journal of Systems and Software 212 (2024) 112016

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Multi-granularity coverage criteria for deep reinforcement learning systems ☆

Ying Shi, Beibei Yin *, Zheng Zheng

School of Automation Science and Electrical Engineering, Beihang University (BUAA), Beijing, China

ARTICLE INFO

Dataset link: https://github.com/stoneshadows/DRL-coverage

Keywords:
Coverage criteria
DRL systems
DRL testing
Structural elements

ABSTRACT

Deep reinforcement learning (DRL) systems are progressively being deployed in safety- and security-critical domains, such as self-driving cars and unmanned aerial vehicles, raising concerns about their trustworthiness. DRL, a integration of deep learning (DL) and reinforcement learning (RL) principles, represents a machine learning technique. Existing DL coverage criteria primarily focus on neuron coverage, overlooking the unique features of RL, thus falling short in assessing the test adequacy of DRL systems.

This paper introduces the first set of coverage criteria designed to systematically measure the elements of DRL systems exercised by test inputs. DRL elements, including states, actions, and policies, are leveraged to define coverage criteria that consider multi-levels and multi-granularities. Furthermore, these coverage criteria undergo optimization through the application of combinatorial coverage principles. State coverage is employed as feedback to guide test case selection for DRL systems. Empirical studies have been conducted to assess the performance of the proposed coverage criteria on five well-known DRL environments. The experiments demonstrate the effectiveness of these coverage criteria in detecting unexpected behaviors, highlighting that the proposed test case selection guided by state coverage serves as an effective strategy.

1. Introduction

In recent years, Deep reinforcement learning (DRL) has demonstrated significant advancements, surpassing human-level performance in diverse domains such as robotics (Gu et al., 2017), video games (Levine et al., 2016), self-driving cars (Pan et al., 2017), unmanned aerial vehicles (Li and Liu, 2021), and healthcare problem-solving (François-Lavet et al., 2018). The wide applications of DRL techniques bring new challenges to testing due to the substantial divergence in the underlying mechanisms of DRL systems compared to traditional software systems and Deep learning (DL) systems. Reinforcement learning (RL) involves an agent dynamically interacting with an environment, receiving penalties or rewards, and formulating a policy for action selection to maximize cumulative rewards (Sutton and Barto, 2018). On the other hand, DL (LeCun et al., 2015) encompasses the extraction and transformation of data features through multiple layers of nonlinear processing models. As a result, the distinctive characteristics of both RL and DL contribute to the unique complexities associated with testing DRL systems.

Coverage criteria are the measurement of testing adequacy and the evaluation method of software quality (Zhu et al., 1997). They can point out what program structure should be exercised to constitute a thorough test, provide terminating conditions for testing (e.g., coverage is achieved), and quantify the thoroughness of test inputs (Chekam et al., 2017). In the field of software testing, coverage criteria are proposed to approximate the internal behaviors of software. According to different partitions of the software structure, different test cases are selected for testing, which will have higher chances of covering diverse parts of the software structure. For example, in traditional software, statement coverage (Zhu et al., 1997) ensures that each statement in the program is executed at least once; branch coverage (Jia and Harman, 2010) attempts to cover all branches in the software (Chekam et al., 2017). The two coverage criteria are derived from the statements and branches of the software structure.

For DL testing, many coverage criteria have been proposed based on neuron coverage, the idea of which is to maximize the number of neurons activated. For example, the layer-level coverage (Ma et al., 2018a) measures the proportion of the total number of k neurons with the highest activation value on each layer in all neurons, and the t-way combination coverage (Ma et al., 2018b) is the ratio of all the neuron-activation configurations covered by test cases and the total number of neuron-activation configurations of t-way combinations of neurons. Existing DL coverage criteria depend heavily on neuron coverage but do not consider any feature of RL and therefore fail to measure the adequacy of testing DRL systems.

DRL problem is a sequential decision problem in which the agent needs to make decisions continuously. In general, the problem is modeled as the Markov decision process (MDP) (Sutton and Barto, 2018). An MDP is a 4-tuple $(S, \mathcal{A}, P, R)$, where S is the state space, $\mathcal{A}$ is

☆ Editor: Antonia Bertolino.

* Corresponding author.
E-mail addresses: shiying2017@buaa.edu.cn (Y. Shi), yinbeibei@buaa.edu.cn (B. Yin), zhengz@buaa.edu.cn (Z. Zheng).

https://doi.org/10.1016/j.jss.2024.112016
Received 29 January 2023; Received in revised form 27 February 2024; Accepted 4 March 2024
Available online 11 March 2024

the action space, P is the transition probability, and R is the reward function. The states are the observations of the environment, and the actions are the possible decisions that the agent can make.

The transition probability is the rule for transitions between states based on actions performed in the environment. The reward is the reaction of the environment to the performed action, signals for positive and negative behaviors (Sutton and Barto, 2018). Given the state s_t at time t, the agent selects an action a_t and receives a reward r_t. The environment turns into a new state s_{t+1}, and the agent selects the next action. The environment and agent interact continually until the agent reaches a terminal state. States, actions, and policies mentioned above are defined as the internal elements of DRL systems, which describe the interaction process between the agent and the environment. Thus, it is natural to design coverage criteria based on these internal elements.

DRL is trial-and-error learning in which the interaction process between the agent and environment characterizes the diverse behaviors of DRL systems, including good (high-reward) and "unexpected (low-reward) behaviors". The agent faces a large exploration space for the environment at the initial training and tends to good behaviors. The exploration space refers to the different combinations of actions and states that the agent can take, i.e., all the possibilities it can explore in the environment. At the beginning of training, the agent has limited knowledge about the environment and lacks an understanding of which actions are good or bad, so it will explore widely in the exploration space, attempting diverse actions to gain insights into the environment and optimize its policies for obtaining higher rewards. The unexpected behavior refers to a task failure or a sequence of low reward values potentially leading to task failures. We expect to comprehensively explore the entire exploration space, as this enables us to depict unexpected behaviors through their association with low-reward results.

Based on this idea, in this paper, we propose a set of coverage criteria for DRL systems, which monitor the internal elements at different levels and granularities and provide multifaceted descriptions of DRL systems. From this perspective, state-level coverage can cover the basic structure of the DRL architecture, which requires the agent to explore the state space as comprehensively as possible. Action-level coverage can cover the action selection process, which requires the agent to select each possible action at least once. Policy-level (state–action) coverage can cover a relatively complete interaction process, which requires the agent with its policy to trigger as many unexpected behaviors as possible that are generated during the interaction.

We evaluate these coverage criteria across five environments and seven algorithms. The experimental results demonstrate the effectiveness of the proposed coverage criteria in detecting unexpected behaviors. A test case for the trained agent comprises a sequence of states formed as the agent interacts with the environment, selecting actions, and transitioning to subsequent states based on feedback from the environment. The evaluation of the trained agent's performance is conducted by observing the rewards it accrues throughout this interactive process. Furthermore, the effectiveness of test case selection guided by state coverage has been verified and found to be more effective than random selection.

In summary, our main contributions to this paper are as follows:

- To the best of our knowledge, this is the first work to investigate the coverage criteria of DRL systems.
- A set of coverage criteria of DRL systems are designed by considering states, actions, and policies, which measure the degrees of coverage from dimension boundaries and quantity distributions.
- The optimization of these coverage criteria is driven by the concept of combinatorial coverage, and test case selection guided by state coverage is proposed.
- An evaluation approach for the effectiveness of these coverage criteria is proposed from two perspectives: analysis of variance (ANOVA) (Scheffe, 1999) and correlation analysis (Black et al., 2010).
- An experimental platform is established to evaluate and compare coverage criteria for DRL systems. This platform incorporates five widely recognized DRL environments and integrates seven popular DRL algorithms for comprehensive evaluation. We have added the source code into the repository to allow the repeatability of experiments.[1]

The structure of this paper is as follows: Section 2 offers a concise overview of the related work. Section 3 provides preliminaries about coverage criteria, RL architecture, examples of low reward values and unexpected behaviors, and the motivation of the proposed coverage criteria. In Section 4, a comprehensive description of our proposed coverage criteria, optimization coverage, and the process of test case selection is presented. The research questions and experimental setup are delineated in Section 5, while Section 6 is dedicated to the presentation and analysis of results. Finally, the conclusion and future work are provided in Section 7.

2. Related work

This section provides a review of coverage criteria applicable to traditional software and DL systems, as well as a survey of recent advancements in testing methodologies for DRL systems.

2.1. Coverage criteria in software testing

Coverage criteria are the objective measurement of test quality in software testing (Zhu et al., 1997). Many coverage criteria have been proposed at different levels and have been widely applied in the software industry.

There are some basic notions of coverage criteria. For example, statement coverage (Zhu et al., 1997) measures whether every statement has been exercised by testing at least once. Branch coverage (Jia and Harman, 2010) focuses on whether control transfers in the program under test have been exercised during testing. Path coverage (Zhu et al., 1997) requires that all the execution paths from the entry to its exit in the program are executed during testing. Besides, modified condition/decision coverage (Chilenski and Miller, 1994) requires that every point of entry and exit and every decision in the program are executed at least once.

In recent years, coverage criteria for the model structure of DL systems have been proposed. DeepXplore (Pei et al., 2017) pioneered to propose neuron coverage, the first coverage criteria specifically designed for DL testing. DeepGauge (Ma et al., 2018a) proposed more fined-grained neuron criteria to represent the major and corner behaviors of deep neural networks (DNNs), and presented layer-level coverage criteria that consider the top hyperactive neurons and their combinations.

Following the works around neuron coverage, some techniques introduce test case generation guided by variants coverage criteria of neuronal coverage. DeepCT (Ma et al., 2018b) adopts the idea of combinatorial testing to propose a set of coverage criteria and uses them to guide the test case generation. DeepHunter and TensorFuzz (Xie et al., 2019; Odena et al., 2019) develop coverage-guided fuzzing methods for neural networks, providing a testing framework for detecting potential problems in DNNs. NCP (Xie et al., 2022) explores the relationship between the structural coverage and the decision logic of DNNs and proposes the interpretable coverage criteria by constructing the decision structure of a DNN.

However, the coverage criteria proposed by these works are about the structural characteristics of DNN models without considering the characteristics of RL and are not suitable for DRL systems.

2.2. DRL testing

Currently, some researchers are attempting to study how to test DRL systems.

[1] https://github.com/stoneshadows/DRL-coverage

There exist several works that applied neuron coverage to DRL systems and explored its role in DRL testing. Trujillo et al. (2020) investigate whether neuron coverage could be used for DRL systems, and they concluded that neuron coverage was insufficient for the design or structure of DRL networks. Genetic algorithm of neuron coverage (GANC) (Al-Nima et al., 2021) is an approach to improving the robustness and performance of DRL networks, which uses genetic algorithms to maximize neuron coverage by generating augmented inputs. However, neuron coverage is currently applied only to DNNs, and this concept has been controversial in academia.

Some studies propose fault detection approaches of DRL systems and provide testing oracle designing strategies for test case generation. Nikanjam et al. (2021) attempt to categorize faults that occur in DRL programs and propose a fault detection method for DRL testing using static analysis and graph transformation. MDPFuzz (Pang et al., 2022) is a black-box fuzz testing framework for models solving MDPs, which forms testing oracles by checking whether the models enter abnormal and dangerous states and decides which mutated state to retain. Eniser et al. (2022) propose fuzzing strategies for test case generation and design metamorphic relations for action policies providing an oracle for the correct output. As far as we know, there are no studies that specifically designed coverage criteria for DRL testing before our paper.

3. Preliminaries and motivation

This section first introduces structural coverage criteria and then presents the RL architecture. It can be found that DRL systems have fundamental differences from both the traditional software and DL systems. Finally, the motivation to design coverage criteria from the characteristics of RL architectures for DRL systems is shown.

3.1. Structural coverage criteria

To test software quality, testers select test inputs to execute the software and then check whether the test output is consistent with the expected one, in which coverage criteria quantify how adequate the software is being tested.

Coverage criteria are widely used to guide the testing process at different levels, including unit (module) level for testing individual units, integration level for testing the combination of units, and system level for testing all units as a whole (Patton, 2006). It partitions the software program into several units that are tested separately, and then the integrated units are tested. If the testing fails at the unit level, the problem must be in individual units. If unit testing at the integration level fails, the problem will be related to the interaction between units. This testing approach is more precise than the testing at the system level.

DL systems and traditional software are fundamentally different in design: DL systems are data-driven programming paradigms, while the logic of the program in traditional software is manually specified. DNNs are the architectures of DL systems, which consist of multiple layers, and each layer contains many interconnected neurons (computation units). Neurons are connected with the neuron of the next layer, and the edge weight indicates the connection strength. Currently, coverage criteria for DL systems are mainly based on the structure of DNN models.

3.2. DRL models

DRL models are considered as software, which comprises test environments and DRL algorithms in this paper. We define a DRL system to be any software system that includes at least one DRL model. Note that some DRL systems might comprise solely a DRL model while others may have some DRL models interacting with other traditional software to produce the final output.

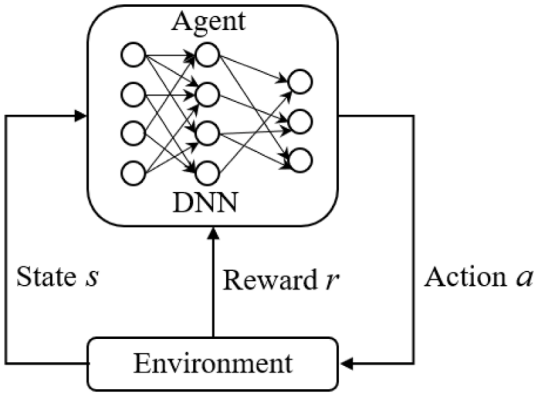

Fig. 1. The agent-environment interaction in RL.

DRL models can be expressed as an MDP, which searches for an optimal policy in an optimization mechanism for solving problems. MDP (Bellman, 1957) is a mathematical model of sequential decision problems, and its main elements consist of 1) a set of environment and agent states, 2) a set of actions of the agent, 3) policies mapping states to actions, 4) rewards that describe what the agent observes (Sutton and Barto, 2018).

Fig. 1 shows the interaction between the agent and environment. At each step t, the agent receives a state s_t and takes an action a_t from some set of possible actions according to its policy π. The policy is a mapping from perceived states of the environment to actions to be taken when in those states (Sutton and Barto, 2018), which is described by $\pi : S \to \mathcal{A}$. Then the agent receives the next state s_{t+1} and a reward r_t. The process continues until the agent reaches a terminal state, after which the process restarts. There is an inherent transition between states, where the transition probability $p\left(s_{t+1}|s_t, a_t\right)$. The return from a state is defined as the cumulative reward $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} r\left(s_i, a_i\right)$ with a discounting factor $\gamma \in [0, 1]$. The goal of the agent is to learn a policy π for action selection, which maximizes the expected cumulative reward $J = \mathbb{E}\left[R_t\right]$.

3.3. Motivation

This section initially points out the limitations of current coverage criteria and subsequently illustrates the effectiveness of the proposed coverage criteria through a practical example.

In DRL systems, when randomly assigned a test case, the primary function code is executed line by line. During this process, statement coverage can easily achieve 100% with only one test case. Nevertheless, it is evident that relying on just one test case is insufficient for adequately testing DRL systems. Meanwhile, coverage criteria for DL systems overlook the unique characteristics of RL systems, and DNN models within DRL systems only facilitate the mapping from states to actions for DRL systems. Consequently, applying these coverage criteria designed for DL systems to measure the testing adequacy of DRL systems is unreasonable. Hence, there is a pressing need to introduce relevant coverage criteria tailored for the effective testing of DRL systems.

Let us further investigate the characteristics of DRL systems, shown in Fig. 2. In this example, the environment contains four discrete states, and the agent can select three discrete actions. The total number of state–action pairs is $4 \times 3 = 12$.

For sequence 1: $s_0 \xrightarrow{a_2} s_3$, the states s_0 and s_3 are covered, and state-level coverage is 1/2. The action a_2 is covered, and action-level coverage is 1/3. The state–action pair (s_0, a_2) is covered, and policy-level (state–action) coverage is 1/12.

For sequence 2: $s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_1} s_3$, the states s_1, s_2, and s_3 are covered, and state-level coverage is 3/4. The actions a_1 and a_2 are covered, and

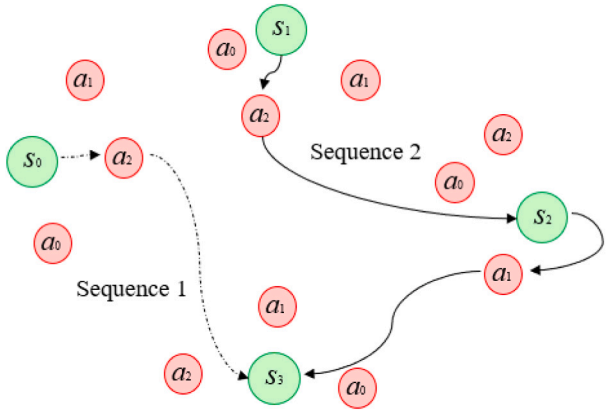

Fig. 2. Two sequences with states and actions.

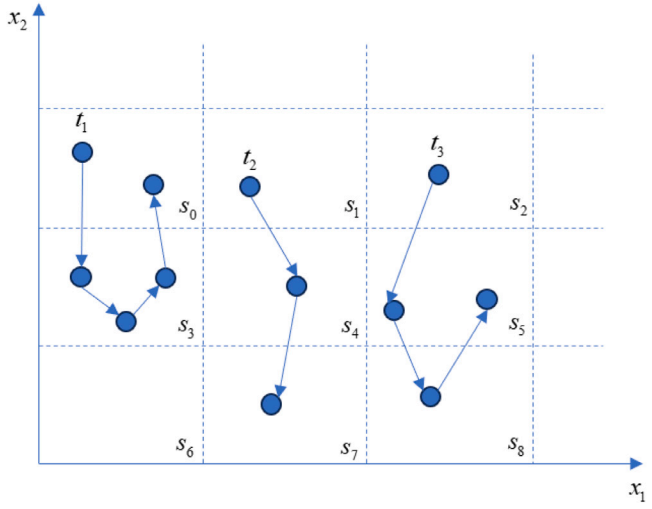

Fig. 3. The state sequences in the state space.

action-level coverage is 2/3. The state–action pairs (s_1, a_2) and (s_2, a_1) are covered, and policy-level (state–action) coverage is 1/6.

Suppose the reward for taking action a_1 is -100, and the reward for taking action a_2 is 300. The rewards of sequence 1 including only action a_2 are 300, while the rewards of sequence 2 including actions a_1 and a_2 are 200. Considering the coverage results of sequence 2 are higher than that of sequence 1, we can see that the higher the coverage results are, the more likely it covers the behaviors with low reward values.

The above example is a discrete state space, but in most environments, the state space is continuous and needs to be discretized. Fig. 3 shows three examples of discrete state sequences in the state space that have two dimensions: x_1 and x_2. The grids drawn in dashed lines represent the abstract states, i.e., s_0, s_1, …, s_9, each of which is mapped to a set of concrete states inside the corresponding grid. In this example, the state sequence t_1 is $s_0 \rightarrow s_3 \rightarrow s_0$, the state sequence t_2 is $s_1 \rightarrow s_4 \rightarrow s_7$, and the state sequence t_3 is $s_2 \rightarrow s_8 \rightarrow s_5$.

Coverage criteria based on the elements of DRL systems are good descriptions of coverage, which gauge the testing adequacy from the structure attributes of DRL systems. In the next sections, we elaborate on the proposed coverage criteria designed specifically for DRL systems.

4. Coverage criteria for testing DRL systems

In software testing, different software elements are executed by test inputs to estimate internal behaviors. For the characteristics of DRL systems, we capture three main elements, i.e., states, actions, and policies, to describe internal behaviors comprehensively.

In the following, we first introduce the definition of *major exploration space* and its acquisition algorithm. *Major exploration space* is characterized by the boundaries of state values that are acquired during the training period to approximate precise exploration space.

Specifically, S is a set of states and it is called the state space, which can be discrete or continuous. The observation of an agent is the state s the agent perceives from the environment, denoted as $(x_1, x_2, \ldots, x_n)$. Thus, the state space has n dimension and we call x_i $(i \in I = \{1, 2, \ldots, n\})$ as the state dimension. Each state dimension has the low boundary value low_{x_i} and high boundary value $high_{x_i}$, and the exploration range $[low_{x_i}, high_{x_i}]$ is denoted as X_i.

Definition 1 (*Major Exploration Space*). *Cartesian* product of the exploration range of each state dimension $\prod_{i \in I} X_i = X_1 \times X_2 \times \cdots \times X_n$ is defined as *major exploration space*. We say that a state s is in its *major exploration space* iff $s = (x_1, x_2, \ldots, x_n) \in \prod_{i \in I} X_i$.

Algorithm 1 shows how to acquire the *major exploration space*, where line 1 is the beginning of one episode and will generate a sequence of states, line 2 is the initial state in one episode, line 3 is the for loop procedure and each loop obtains a new state, line 4 is selecting an action based on the current state and policy, line 5 is the loop compares the values of each state dimension value, line 6 and line 7 indicate that if the value of state dimension x_i is greater than that of the temp state dimension, the value of the state dimension is reserved as the high boundary value, line 8 and line 9 are the same methods to determine the low boundary value of the state dimension, lines 10–14 record the temp boundary value as the base value for comparing with the next state, and line 16 and line 17 take actions to transfer from the current state to the next state and update the reward value. For each state dimension, its values are compared with that of adjacent states and higher and lower ones are recorded as boundary values until the episode ends.

Based on this, we design fine-grained coverage criteria and propose the concepts of state-level coverage, action-level coverage, and policy-level (state–action) coverage. Following these concepts, we take the CartPole-v1 environment as an example and use it to clarify these coverage criteria.

4.1. State-level coverage criteria

In this subsection, two state-level coverage criteria are introduced. SBCov emphasizes high and low boundaries of state dimension values but cannot reflect the distribution of states within boundaries. KSCov is to quantify states within boundaries to cover the state space exhaustively, but its calculation is complicated.

4.1.1. State boundary coverage (SBCov)

State boundary coverage measures the degree to which *major exploration space* has been covered by test states in the continuous space.

For each state dimension x_i, SBCov is defined as:

$$SBCov\left(S; x_i\right) = \frac{TestHigh_{x_i} - TestLow_{x_i}}{TrainHigh_{x_i} - TrainLow_{x_i}},$$

where $TestHigh_{x_i}$ and $TestLow_{x_i}$ are the high and low boundary state values in testing period, and $TrainHigh_{x_i}$ and $TrainLow_{x_i}$ are the high and low boundary state values in training period.

For all dimensions of states, there are two definitions of SBCov. The average value of $SBCov\left(S; x_i\right)$ is defined as:

$$SBCov1\left(S\right) = \frac{\sum_{s \in S, i=1}^{n} SBCov\left(S; x_i\right)}{n}.$$

The dimensional product of $SBCov\left(S; x_i\right)$ is defined as:

$$SBCov2\left(S\right) = \prod_{s \in S, i=1}^{n} SBCov\left(S; x_i\right).$$

Algorithm 1 Acquire the major exploration space

Input: $low_{x_i}(temp\_low_{x_i})$ and $high_{x_i}(temp\_high_{x_i})$: the low and high boundary values x_i of state dimension, $Q(s,a)$: action-value function, π: the policy is a mapping from state s to action a.
Output: $low_{x_1},\dots,low_{x_n}$ and $high_{x_1},\dots,high_{x_n}$ ▷ the major exploration region

```
for episode e = 1, M do
    Initialize state s = (x_1, x_2, ..., x_n)
    for step t = 1, T do
        Choose an action a from state s using policy π derived from Q
        for each state dimension x_i = x_1, x_n do
            if x_i >= temp_high_{x_i} then
                high_{x_i} = x_i
            else if x_i <= temp_low_{x_i} then
                low_{x_i} = x_i
            else
                high_{x_i} = temp_high_{x_i}
                low_{x_i} = temp_low_{x_i}
            end if
            records temp_low_{x_1},...,temp_low_{x_n} and temp_high_{x_1},...,temp_high_{x_n}
        end for
        Q(s,a) ← Q(s,a) + α[γ max_a Q(s',a) − Q(s,a)]
        s ← s' ▷ Take the action a to transfer from the current state s to the next state s', updating the reward value Q(s,a)
    end for
    records low_{x_1},...,low_{x_n} and high_{x_1},...,high_{x_n}
end for
```

Example 1. The state space of the CartPole-v1 environment includes four dimensions: cart position x_1, cart velocity x_2, pole angle x_3, and pole angular velocity x_4. SBCov measures the degree to which the range of states is covered.

First, find out the boundary value of each dimension of the state in training and testing periods. In our experiments,

$TrainHigh = [2.39, 3.92, 0.21, 3.02]$, where 2.39 is the high value of x_1 in training period;

$TrainLow = [-2.38, -3.64, -0.21, -2.87]$, where −2.38 is the low value of x_1 in training period;

$TestHigh = [0.89, 1.31, 0.14, 1.08]$, where 0.89 is the high value of x_1 in testing period;

$TestLow = [-0.99, -1.29, -0.13, -1.06]$, where −0.99 is the low value of x_1 in testing period.

Then, using the boundary values of the states to calculate the coverage of each dimension of the state. The ratio of the test range to the training range of dimensions is used to represent the degree to which the dimensions of states are covered, and it is calculated as follows:

$$SBCov\left(S;x_1\right) = \frac{0.89-(-0.99)}{2.39-(-2.38)} = 0.395,$$

$$SBCov\left(S;x_2\right) = \frac{1.31-(-1.29)}{3.92-(-3.64)} = 0.344,$$

$$SBCov\left(S;x_3\right) = \frac{0.14-(-0.13)}{0.21-(-0.21)} = 0.659,$$

$$SBCov\left(S;x_4\right) = \frac{1.08-(-1.06)}{3.02-(-2.87)} = 0.363.$$

Finally, calculate the average value or the product of each dimension of the state as SBCov:

$$SBCov1\left(S\right) = \frac{SBCov\left(S;x_1\right)+\cdots+SBCov\left(S;x_4\right)}{4},$$

$$SBCov2\left(S\right) = SBCov\left(S;x_1\right)\times\cdots\times SBCov\left(S;x_4\right) = 0.032.$$

4.1.2. K-multisection state coverage (KSCov)

K-multisection state coverage measures how thoroughly *major exploration space* has been covered by test states in the partition space.

To quantify each state dimension, we divide the exploration range $X_i, i \in I$ into k_i equal sections (i.e., k-multisection), denoted as $S_i^d = (d_{i1}, d_{i2}, \dots, d_{ik_i}), i \in I$. If the dimension value of the state s belongs to $d_{ij}, i \in I, j \in J = \{1,2,\dots,k_i\}$, we will say that the j-th section is covered by the state s, denoted as $S_i^d(s) \in d_{ij}$.

Cartesian product $\prod_{i\in I} S_i^d = S_1^d \times S_2^d \times \cdots \times S_n^d$ of k-multisections is defined as the partition set of *major exploration space*, denoted as S^d, which contains $|S^d| = k_1 \times k_2 \times \cdots \times k_i$ discrete states.

KSCov for dimensional product space (KSCov) is defined as:

$$KSCov\left(S\right) = \frac{|\cup_{s\in S} S^d(s)|}{|S^d|},$$

where $S^d(s) = \prod_{i\in I} S_i^d(s)$.

Example 2. KSCov measures the degree to which the interior of the state space is covered. This example follows the example in Section 4.1.1.

First, the k is set as 5 in this example, and each dimension value range of states is divided into five regions as follows:

$$S_1^5 = ([-2.38,-1.43),[-1.43,-0.47),\dots,[1.44,2.39]),$$

$$S_2^5 = ([-3.64,-2.13),[-2.13,-0.62),\dots,[2.41,3.92]),$$

$$S_3^5 = ([-0.21,-0.13),[-0.13,-0.04),\dots,[0.13,0.21]),$$

$$S_4^5 = ([-2.87,-1.69),[-1.69,-0.51),\dots,[1.84,3.02]).$$

Then, record the region in which the dimension value of each state falls. For example, given a state $s = [0.32, 2.83, 0.03, 1.56]$,

$S_1^5(s) = 0.32 \in d_{13} = [-0.47, 0.48]$ and it means the dimension x_1 of the state s falls into the 3rd region of S_1^5;

$S_2^5(s) = 2.83 \in d_{25} = [2.41, 3.92]$ and it means the dimension x_2 of the state s falls into the 5th region of S_2^5;

$S_3^5(s) = 0.03 \in d_{32} = [-0.04, 0.04]$ and it means the dimension x_3 of the state s falls into the 5th region of S_3^5;

$S_4^5(s) = 1.56 \in d_{44} = [0.66, 1.84]$ and it means the dimension x_4 of the state s falls into the 5th region of S_4^5.

Finally, $S^5(s) = [3,5,2,4]$ and it represents the position where each dimension of state s falls into the corresponding dimension region. Repeating the above steps, it is assumed that all test states in the set of states S can cover 400 regions of the state space discretization, i.e., $|\cup_{s\in S} S^d(s)| = 400$, and the number of all discretization regions is $|S^d| = 5\times5\times5\times5 = 625$. Thus, KSCov is calculated as:

$$KSCov\left(S\right) = \frac{|\cup_{s\in S} S^d(s)|}{|S^d|} = \frac{400}{625} = 0.64.$$

4.2. Action-level coverage criteria

Let $\mathcal{A}$ be a set of actions and it is called the action space, which can be discrete or continuous. Assume the action space has m dimension, each action dimension y_i ($i \in I = \{1,2,\dots,m\}$) has the low boundary value low_{y_i} and high boundary value $high_{y_i}$, and the exploration range $[low_{y_i}, high_{y_i}]$ is denoted as Y_i.

4.2.1. Action boundary coverage (ABCov)

Action boundary coverage measures the degree to which the action space in training period has been covered by test actions in the continuous space.

For each action dimension y_i, ABCov is defined as:

$$ABCov\left(\mathcal{A};y_i\right) = \frac{TestHigh_{y_i} - TestLow_{y_i}}{TrainHigh_{y_i} - TrainLow_{y_i}},$$

where $TestHigh_{y_i}$ and $TestLow_{y_i}$ are the high and low boundary action values in the testing period, and $TrainHigh_{y_i}$ and $TrainLow_{y_i}$ are the high and low boundary action values in the training period.

For all dimensions of actions, there are two definitions of ABCov. The average value of $ABCov\left(\mathcal{A}; y_i\right)$ is defined as:

$$ABCov1\left(\mathcal{A}\right) = \frac{\sum_{a\in\mathcal{A},i=1}^{m} ABCov\left(\mathcal{A}; y_i\right)}{m}.$$

The dimensional product of $ABCov\left(\mathcal{A}; y_i\right)$ is defined as:

$$ABCov2\left(\mathcal{A}\right) = \prod_{a\in\mathcal{A},i=1}^{m} ABCov\left(\mathcal{A}; y_i\right).$$

4.2.2. K-multisection action coverage (KACov)

K-multisection action coverage measures how thoroughly the action space in the training period has been covered by test states in the partition space.

To quantify each action dimension, we divide the exploration range $Y_i, i \in I$ into k_i equal sections (i.e., k-multisection), denoted as $A_i^g = (g_{i1}, g_{i2}, \dots, g_{ik_i}), i \in I$. If the dimension value of the action a belongs to $g_{ij}, i \in I, j \in J = \{1, 2, \dots, k_i\}$, we will say that the j-th section is covered by the state a, denoted as $A_i^g(a) \in g_{ij}$.

Cartesian product $\prod_{i\in I} A_i^g = A_1^g \times A_2^g \times \cdots \times A_n^g$ of k-multisections is defined as the partition set of the action space in training period, denoted as A^g, which contains $|A^g| = k_1 \times k_2 \times \cdots \times k_i$ discrete actions.

KACov for dimensional product space (KACov) is defined as:

$$KACov\left(\mathcal{A}\right) = \frac{|\cup_{a\in\mathcal{A}} A^g(a)|}{|A^g|},$$

where $A^g(a) = \prod_{i\in I} A_i^g(a)$.

Example 3. Similar to state-level coverage, action-level coverage measures the degree to which the action space is covered.

The action space of the CartPole-v1 environment is discrete and includes two dimensions: pushing the cart to the left and pushing the cart to the right. In our experiments, both actions are involved. Thus, ABCov and KACov are both 100%.

4.3. Policy-level (state–action) coverage criteria

The policy that is the mapping of states to actions in DRL environments is described by state–action pairs in this subsection. State–action coverage criteria measure the pairs of states and actions in the training period that have been covered by in the testing period in the process of action selection and state transition.

According to the definitions of KSCov and KACov, *Cartesian* product $\prod_{i=1}^{n} S_i^d \times \prod_{j=1}^{m} A_j^g$ is defined as the partition set of state–action pairs in the training period, denoted as $S^d \times A^g$.

State–action coverage (SACov) is defined as:

$$SACov\left((S, \mathcal{A})\right) = \frac{|\cup_{s\in S, a\in\mathcal{A}}(S^d(s), A^g(a))|}{|S^d \times A^g|}.$$

Example 4. SACov measures the degree to which the state–action pairs are covered. According to the examples in Section 4.1.2, each dimension value range of states is divided into five regions, and all test states can cover 400 discretization regions of the state space.

It is assumed that these test states corresponding to 2 actions cover 300 regions and states corresponding to 1 action cover 100 regions, i.e., $|\cup_{s\in S, a\in\mathcal{A}}(S^d(s), A^g(a))| = 300 \times 2 + 100 \times 1 = 700$. The number of discretization regions about state and action space is $|S^d \times A^g| = 625 \times 2 = 1250$. Thus, SACov is calculated as:

$$SACov\left((S, \mathcal{A})\right) = \frac{|\cup_{s\in S, a\in\mathcal{A}}(S^d(s), A^g(a))|}{|S^d \times A^g|} = \frac{700}{1250} = 0.56.$$

4.4. Optimized state coverage criteria

Based on the hypothesis that local state dimensions work independently, we use the idea of combinatorial coverage to optimize KSCov. Combinatorial coverage (Kuhn et al., 2013) is to cover the combinations of parameter values with relations on (discrete) input domains, which reduces the computational complexity of state space.

Applied the above concepts to DRL systems, the state space corresponds to input domains, and state dimension values correspond to parameter values. Specifically, a subset of the partition set of *major exploration space* is represented as $S^d\left\lfloor_{x_{i_1,i_2,\dots,i_l}}\right\rfloor$, in which

$$x_{i_1,i_2,\dots,i_l} = \left\{x_{i_1}, x_{i_2}, \dots, x_{i_l} | i_1 < i_2 < \cdots < i_l, i_k = 1, 2, \dots, n\right\}.$$

The subset with relations between state dimensions is denoted as R, and $\sum_{r\in R} |S^d{}_{\lfloor r\rfloor}|$ is defined as the optimized set of *major exploration space*.

Optimized state coverage criteria (OSCov) is defined as:

$$OSCov\left(S\right) = \frac{\sum_{r\in R} |\cup_{s\in S} S^d(s)_{\lfloor r\rfloor}|}{\sum_{r\in R} |S^d{}_{\lfloor r\rfloor}|}.$$

In the following, several typical relations between two state dimensions are investigated.

4.4.1. Single state dimension coverage

State dimensions are independent, and there are no relations between state dimensions.

In this case, the relations of single state dimensions are presented as $R = \left\{\left\{x_1\right\}, \left\{x_2\right\}, \dots, \left\{x_n\right\}\right\}$. The total number of sections to be covered is $\sum_{r\in R} |S^d{}_{\lfloor r\rfloor}| = |S_1^d| + |S_2^d| + \cdots + |S_n^d| = k_1 + k_2 + \cdots + k_i$, which reduces the computational complexity of coverage criteria compared with $k_1 \times k_2 \times \cdots \times k_i$ for KSCov.

4.4.2. Pairwise state dimension coverage

Two state dimensions have different combinations, and there is relations between two state dimensions.

In this case, three combination patterns are considered: *link pairwise*, *cross pairwise*, and *mutual pairwise*.

The relations of *link pairwise* for state dimensions are presented as $R = \left\{\left\{x_1, x_2\right\}, \left\{x_3, x_4\right\}, \left\{x_5, x_6\right\}, \dots\right\}$. The total number of sections to be covered is $\sum_{r\in R} |S^d{}_{\lfloor r\rfloor}| = |S_1^d \times S_2^d| + |S_3^d \times S_4^d| + |S_5^d \times S_6^d| + \cdots = k_1 \times k_2 + k_3 \times k_4 + k_5 \times k_6 + \cdots$.

The relations of *cross pairwise* for state dimensions are presented as $R = \left\{\left\{x_1, x_2\right\}, \left\{x_2, x_3\right\}, \left\{x_3, x_4\right\}, \dots\right\}$. The total number of sections to be covered is $\sum_{r\in R} |S^d{}_{\lfloor r\rfloor}| = |S_1^d \times S_2^d| + |S_2^d \times S_3^d| + |S_3^d \times S_4^d| + \cdots = k_1 \times k_2 + k_2 \times k_3 + k_3 \times k_4 + \cdots$.

The relations of *mutual pairwise* for state dimensions are presented as

$R = \left\{\left\{x_1, x_2\right\}, \left\{x_1, x_3\right\}, \dots, \left\{x_2, x_3\right\}, \left\{x_2, x_4\right\}, \dots\right\}$. The total number of sections to be covered is $\sum_{r\in R} |S^d{}_{\lfloor r\rfloor}| = \sum_{1\le i<j\le n} |S_i^d \times S_j^d| = \sum_{1\le i<j\le n} k_i \times k_j$.

In addition to considering the relations on single and pairwise dimensions, it is possible to generalize the typical relations to the case where more dimensions interact.

Example 5. Following the setup of KSCov, determine the position of each state s in the discretized state space $S^d(s)$. Assume there are two states and their corresponding positions are $S^d(s_1) = [1, 5, 2, 4]$ and $S^d(s_2) = [2, 3, 2, 4]$, the OSCov of two states is calculated as follows:

For single state dimension coverage, $\sum_{r\in R} |\cup_{s\in S} S^d(s)_{\lfloor r\rfloor}| = |\{\{1\}, \{2\}\}| + |\{\{5\}, \{3\}\}| + |\{\{2\}\}| + |\{\{4\}\}| = 2 + 2 + 1 + 1 = 6$, and $\sum_{r\in R} |S^d{}_{\lfloor r\rfloor}| = |S_1^d| + |S_2^d| + |S_3^d| + |S_4^d| = 5 + 5 + 5 + 5 = 20$.

$$Single\left(s_1, s_2\right) = \frac{\sum_{r\in R} |\cup_{s\in S} S^d(s)_{\lfloor r\rfloor}|}{\sum_{r\in R} |S^d{}_{\lfloor r\rfloor}|} = \frac{6}{20} = 0.3.$$

For link pairwise state dimension coverage, $\sum_{r\in R}|\cup_{s\in S}S^d(s)_{\lfloor r\rfloor}| = |\{\{1,5\},\{2,3\}\}| + |\{\{2,4\}\}| = 2+1 = 3$, and $\sum_{r\in R}|S^d{}_{\lfloor r\rfloor}| = |S_1^d \times S_2^d| + |S_3^d \times S_4^d| = 5\times 5 + 5\times 5 = 50$.

$$Link\left(s_1,s_2\right) = \frac{\sum_{r\in R}|\cup_{s\in S}S^d(s)_{\lfloor r\rfloor}|}{\sum_{r\in R}|S^d{}_{\lfloor r\rfloor}|} = \frac{3}{50} = 0.06.$$

For cross pairwise state dimension coverage, $\sum_{r\in R}|\cup_{s\in S}S^d(s)_{\lfloor r\rfloor}| = |\{\{1,5\},\{2,3\}\}| + |\{\{5,2\},\{3,2\}\}| + |\{\{2,4\}\}| = 2+2+1 = 5$, and $\sum_{r\in R}|S^d{}_{\lfloor r\rfloor}| = |S_1^d \times S_2^d| + |S_2^d \times S_3^d| + |S_3^d \times S_4^d| = 3\times 5\times 5 = 75$.

$$Cross\left(s_1,s_2\right) = \frac{\sum_{r\in R}|\cup_{s\in S}S^d(s)_{\lfloor r\rfloor}|}{\sum_{r\in R}|S^d{}_{\lfloor r\rfloor}|} = \frac{5}{75} = 6.67E-2.$$

For mutual pairwise state dimension coverage, $\sum_{r\in R}|\cup_{s\in S}S^d(s)_{\lfloor r\rfloor}| = |\{\{1,5\},\{2,3\}\}|+|\{\{1,2\},\{2,2\}\}|+|\{\{1,4\},\{2,4\}\}|+|\{\{5,2\},\{3,2\}\}|+|\{\{5,4\},\{3,4\}\}|+|\{\{2,4\}\}| = 11$, and $\sum_{r\in R}|S^d{}_{\lfloor r\rfloor}| = \sum_{1\le i<j\le 4}|S_i^d \times S_j^d| = C_4^2\times 5\times 5 = 150$.

$$Mutual\left(s_1,s_2\right) = \frac{\sum_{r\in R}|\cup_{s\in S}S^d(s)_{\lfloor r\rfloor}|}{\sum_{r\in R}|S^d{}_{\lfloor r\rfloor}|} = \frac{11}{150} = 7.33E-2.$$

4.5. State coverage guiding test case selection

One of the applications of coverage criteria is to provide a guideline for test case selection, which aims to select a subset of test cases to represent the whole test set to improve testing efficiency. In DRL testing, we selected state sequences with a large difference to promote the diversity of coverage for the state space.

4.5.1. Similarity metrics

two similarity metrics are proposed to select state sequences as diverse as possible in the direction of increasing coverage, and the differences in two state sequences are measured by occurrence states and adjacent state pairs.

State occurrence similarity metric. The basic idea is to count the number of states in each state sequence, and calculate Jaccard indices of occurrence states in two episodes as follows:

$$J\_state(x,y) = \frac{\left|s_x \cap s_y\right|}{\left|s_x \cup s_y\right|},$$

where s_x and s_y represent the occurrence states of two state sequences x and y, respectively. The metric measures the similarity of states in two state sequences.

State transition similarity metric. The basic idea is to record adjacent state pairs in each state sequence and calculate Jaccard indices of state pairs in two episodes as follows:

$$J\_transition(x,y) = \frac{\left|t_x \cap t_y\right|}{\left|t_x \cup t_y\right|},$$

where t_x and t_y represent state pairs of two state sequences x and y, respectively. The metric measures the similarity degree of state pairs in two state sequences.

For achieving the maximum coverage of the state space, the above two metrics are used to select state sequences with a large difference, and the following examples show how they calculate the similarity of two state sequences.

4.5.2. Examples of calculating two similarity metrics

Suppose state sequence x is $s_0 \to s_1 \to s_2 \to s_3$, where s_0 is the initial state; state sequence y is $s_1 \to s_2 \to s_3 \to s_1 \to s_4 \to s_5$, where s_1 is the initial state. *State occurrence similarity* of the two sequences can be calculated as:

$$J\_state(x,y) = \frac{\left|s_x \cap s_y\right|}{\left|s_x \cup s_y\right|} = \frac{1}{2},$$

where $s_x \cap s_y = \{s_1, s_2, s_3\}$ and $s_x \cup s_y = \{s_0, s_1, s_2, s_3, s_4, s_5\}$.

The set of state pairs in sequence x is $\{(s_0,s_1),(s_1,s_2),(s_2,s_3)\}$ and the set of state pairs sequence y is $\{(s_1,s_2),(s_2,s_3),(s_3,s_1),(s_1,s_4),(s_4,s_5)\}$. *State transition similarity* of the two sequences can be calculated as:

$$J\_transition(x,y) = \frac{\left|t_x \cap t_y\right|}{\left|t_x \cup t_y\right|} = \frac{1}{3},$$

where $t_x \cap t_y = \{(s_1,s_2),(s_2,s_3)\}$ and $t_x \cup t_y = \{(s_0,s_1),(s_1,s_2),(s_2,s_3),(s_3,s_1),(s_1,s_4),(s_4,s_5)\}$.

The two metrics measure the similarity of state sequences from occurrence states and state transition, which is conducive to selecting more diverse state sequences.

4.5.3. The steps of test case selection

Based on two similarity metrics, state coverage guiding test case selection can cover state occurrence and adjacent state pairs in state sequences, and the following four steps show the process of test case selection in DRL systems.

Step 1. Discretize the state space and record all the states and adjacent state pairs in each episode to describe the coverage of state sequences.

Step 2. Compare state coverage of the current state sequence with that of the previous state sequence and calculate state occurrence and state transition similarity metrics according to data recorded in *Step 1*.

Step 3. Calculate two similarity metrics of state sequences and then record state sequences with a similarity value of 0 and their rewards. Complete the specified number of test episodes and obtain all state sequences with a similarity value of 0.

Step 4. Calculate the average rewards of all state sequences saved in *Step 3* and compare them with the average rewards of state sequences selected by random selection in the same number of test episodes.

We will illustrate by experiments that test case selection based on two similarity metrics is more effective than random selection in unexpected behavior detection.

5. Experimental setup

In this section, we introduce the research questions and describe the test environments and algorithms. Additionally, we employ a set of analysis methods to gauge the effectiveness of the proposed coverage criteria.

5.1. Research questions

We conduct an empirical study to investigate the proposed coverage criteria in relation to the following three research questions:

- **RQ1:** *How effective are the proposed coverage criteria in detecting unexpected behaviors?* To answer this question, we perform experiments across five environments and with seven DRL algorithms to assess the effectiveness of the proposed coverage criteria.
- **RQ2:** *What are the relations between the number of episodes with low reward values and the results of coverage criteria?* To answer this question, we initially introduce examples of low reward values and unexpected behaviors. Subsequently, we quantitatively analyze the relations between low reward values and the results of the proposed coverage criteria.
- **RQ3:** *How effective are test case selection strategies guided by state coverage for detecting unexpected behaviors?* To answer this question, we compare the proposed test case selection with the random selection.

Table 1
The introduction of five environments.

Env.	Introduction	State space	Action space	Reward function
CP	A pole is attached by an un-actuated joint to a cart and moves along a frictionless track.	4d continuous	2d discrete	reward = +1 for each time step, including the termination step
MC	A car is on a one-dimensional track positioned between two mountains, and the goal is to drive up the mountain on the right.	2d continuous	3d discrete	reward = 0 if the car reaches the goal position −1 for each time step if the position is less than 0.5
PD	The pendulum starts in a random position, and the goal is to swing it up to stay upright.	3d continuous	1d continuous	reward = $-(angle^2 + 0.1 \times (angular\_velocity)^2 + 0.001 \times action^2)$
LL	If the lander moves from the top of the screen to the landing pad, it will get rewards. If the lander moves away from the landing pad, it will lose rewards.	8d continuous	4d discrete	reward = +100 for safe landing on the landing pad −100 for crashes +10 for each leg contact with the landing pad
BW	If the robot moves forward, it will get rewards. If the robot falls, it will lose rewards. The more optimal the agent will get better scores.	24d continuous	4d continuous	reward = +300 for forward movement −100 if the robot falls

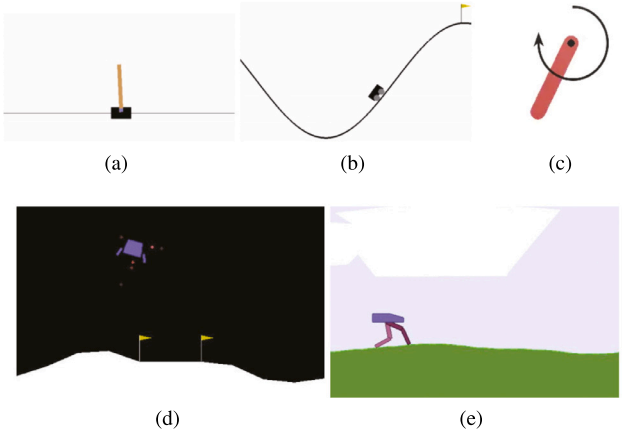

Fig. 4. Examples of control task environments: (a) CartPole-v1 (CP), (b) MountainCar-v0 (MC), (c) Pendulum-v0 (PD), (d) LunarLander-v2 (LL), (e) BipedalWalker-v3 (BW).

Table 2
The information of DRL models, including average rewards, average time, and the number of episodes.

ID	Env.	Alg.	Ave. rewards	Time	Episodes
1	CP	DQN	322.41	3 h/3 min	800/50
2		Duel	353.71	2 h/1 h	
3		D3QN	403.94	6 h/4 min	
4		PPO	487.68	1 min/6 min	
5	MC	DQN	−109.33	2.5 h/2 min	1500/100
6		Duel	−103.92	3 h/1.5 min	
7		D3QN	−105.48	18 h/1 min	
8	PD	DDPG	−621.08	14 min/1.5 min	600/50
9		TD3	−149.79	15 min/1.5 min	
10		SAC	−186.75	20 min/2 min	
11	LL	DQN	188.39	7 h/11 min	1500/100
12		Duel	150.54	8 h/8 min	
13		D3QN	149.96	1.6days/5 min	
14		AC	177.61	11 min/9 min	
15		TD3	227.93	2 h/3 min	
16		PPO	90.78	16 min/40 min	
17	BW	SAC	269.49	5 h/12 min	1500/100
18		TD3	281.27	2 h/10 min	
		PPO	278.22	1 h/8.5 min	

5.2. Test environments and test algorithms

To validate the effectiveness of our proposed coverage criteria, we select five well-known environments and employ seven DRL algorithms for evaluation. Fig. 4 shows the examples of five OpenAI Gym (Brockman et al., 2016) control task environments: CartPole-v1 (CP), MountainCar-v0 (MC), Pendulum-v0 (PD), LunarLander-v2 (LL), and BipedalWalker-v3 (BW).

Table 1 shows the description about environments, including state space, action space, and reward functions. For example, the introduction of CP is that a pole is attached by an un-actuated joint to a cart and moves along a frictionless track. CP has a 4-dimensional continuous state space, including cart position, cart velocity, pole angle, and pole angular velocity, and it has a 2-dimensional discrete action space, including pushing the cart to the left and pushing the cart to the right. Regarding the reward function, CP exhibits a uniform reward, where the reward remains the same for each time step. In contrast, MC exhibits a nonuniform reward, with distinctions across different time steps. The same discussion holds for the reward functions of PD, LL, and BW.

Table 2 shows seven popular DRL algorithms, including DQN (Mnih et al., 2015), DuelingDQN (Wang et al., 2016), D3QN (Fang et al., 2019), DDPG (Lillicrap et al., 2015), TD3 (Fujimoto et al., 2018), PPO (Schulman et al., 2017), and SAC (Haarnoja et al., 2018). The column labeled Ave. rewards of Table 2 presents the results of average rewards. There are ten independent runs are done and each run includes multiple episodes, then reports the average rewards. For each episode, the state and action values of environments are recorded and compared to obtain the ranges of each state and action dimension. The last column presents the number of training and testing episodes in the test process, which are justified by our many experiments. If the agent's performance reaches or exceeds the preset average reward and does not improve further, it may be training enough indication that it has learned the optimal or near-optimal policy. For example, when the number of episodes is 800 on CP, the learning curve does not improve further. Thus, we set 800 as the number of episodes for each run on CP.

5.3. Effectiveness analysis methods

The effectiveness of the proposed coverage criteria is evaluated from two perspectives: ANOVA and correlation analysis.

ANOVA. An effective coverage criterion should demonstrate discriminative capabilities across various DRL models, implying statistically significant differences in the results of the criterion when applied to different DRL models.

To assess these differences, the ANOVA is conducted to compare the mean of coverage results for each DRL model. The assumptions underlying the ANOVA are verified using the Shapiro–Wilk test (Shapiro and Wilk, 1965) for normality of data and Levene's test (Brown and Forsythe, 1974) for homogeneity of variance.

The F-statistic is a measure of the ratio of between-model variability to within-model variability in the context of ANOVA. A larger F-statistic signifies a more substantial between-model difference in relation to within-model differences. This statistic is commonly employed to assess whether there are statistically significant differences in means among multiple DRL models. Similarly, the Kruskal–Wallis statistic (Kruskal and Wallis, 1952) is utilized as a non-parametric alternative to ANOVA, particularly when the assumptions of normality and homogeneity of variances are not satisfied. A larger Kruskal–Wallis statistic indicates an increased likelihood that there are differences in medians among the models.

Interpreting the results involves considering the p-value, which reflects the probability of obtaining the observed results under the assumption that the null hypothesis is true. If the p-value is below the significance level of 0.05, the null hypothesis is rejected, suggesting sufficient evidence to conclude that at least one model's mean (or median in the case of the Kruskal–Wallis test) differs from the others. A p-value below the stricter significance level of 0.01 provides stronger evidence, indicating a more robust indication of a significant difference among the model means.

Correlation Analysis. An effective coverage criterion is considered successful when higher coverage values correlate with an increased likelihood of capturing unexpected behaviors through test cases. Consequently, the results of effective coverage criteria exhibit a negative correlation with the rewards of corresponding model.

To quantify this correlation, Spearman correlation is employed, as it is less restrictive on the data, more robust than Pearson correlation, and provides a comprehensive evaluation of the correlation.

$$Spearman\ correlation_s = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)},$$

where d_i represents the differences between the ranks of the corresponding data points, and n is the number of data points.

The correlation coefficient, ranging from -1 to 1, signifies the strength and direction of the relationship between variables. A value of -1 indicates a perfect negative relationship (one variable increases as the other decreases).

6. Experimental results and analysis

This section first discusses the effectiveness of the proposed coverage criteria on each environment and then provides a quantitative analysis of unexpected behaviors. Finally, the effectiveness of our test case selection guided by coverage is verified.

6.1. The results and analysis of state-level coverage

In the following, the results about ANOVA and correlation analysis pertaining to SBCov, KSCov (k = 10), SACov, and OSCov are analyzed and discussed.

ANOVA results. Table Table 3 demonstrates whether the ANOVA assumptions are satisfied, including normality and homoscedasticity of the coverage result data. It also presents the F-statistic values, associated p-value1, Kruskal–Wallis statistic values, and corresponding p-value2.

For example, considering the results for SBCov1 on MC, the assumptions of normality and homoscedasticity are satisfied, and the F-statistic value of 7.031 implies the presence of significant differences in the means of coverage values among DQN, Duel, and D3QN. The low p-value1 (0.003) provides evidence against the null hypothesis, indicating that these differences are statistically significant. For the SBCov2 results on PD, the normality assumption is not satisfied. As a result, the Kruskal–Wallis statistic value of 11.533 is more robust in this context. The associated p-value2 (0.003), being less than 0.01, indicates statistically significant differences across different DRL models.

Analogous to the above analysis, it is evident in Table Table 3 that each of the proposed coverage criteria exhibits significant differences when applied to different DRL models in most cases. This observation indicates that the proposed coverage criteria exhibit discriminative capabilities across various DRL models from an ANOVA perspective.

Correlation analysis results. Table Table 4 shows the results of Spearman correlation between coverage and rewards for each model considering the values of each run. Taking each coverage result of CP as an example, for SBCov1 on DQN, DQN, Duel, and D3QN, the Spearman correlation values are -0.468, -0.176, -0.166, and -0.576, which is a negative correlation between SBCov1 results and rewards. Table Table 4 reveals that the proposed coverage criteria exhibit a negative correlation between coverage values and rewards for each DRL model. This observation suggests that high coverage results correspond to low rewards and a higher likelihood of unexpected behavior detection, demonstrating the effectiveness of the proposed coverage criteria from the perspective of correlation analysis.

By employing both ANOVA and correlation analysis, a comprehensive evaluation of the proposed coverage criteria is conducted. This dual approach allows for a more nuanced understanding of how these criteria perform across different DRL models and their potential relationships with the rewards for each DRL model. Such a multi-faceted evaluation is crucial for gaining a holistic perspective on the effectiveness of coverage criteria in diverse scenarios.

In addition, within each environment, different state dimensions exhibit diverse coverage results. For each environment, each state dimension has different coverage results. The SBCov results for each state dimension on CP are depicted in Fig. 5. The horizontal axis of Fig. 5 denotes the number of experimental runs, with each run encompassing 800 episodes. Meanwhile, the vertical axis represents the exploration range of each state dimension, where the green and red regions correspond to the exploration ranges during training and testing periods, respectively. Each subplot in Fig. 5 illustrates the extent to which the range of testing states covers the range of training states for specific state dimensions. For example, the state dimension *pole angle* attains a higher SBCov result compared to other state dimensions, registering values of 0.645 in DQN, 0.769 in Duel, 0.787 in D3QN, and 0.8 in PPO. This implies that the state dimension *pole angle* achieves a satisfactory level of coverage.

6.2. The results and analysis of action-level coverage

The coverage can be further considered from the action perspective, an analogy to state-level coverage, action-level coverage criteria include ABCov, KACov, and OACov.

From Table 1, we can observe that the action space of CP has two discrete dimensions, the action space of MC has three discrete dimensions, and the action space of LL has four discrete dimensions. All of these discrete actions are covered in the experiments. Thus, it is easy to achieve 100% coverage in environments with discrete actions.

For environments with continuous actions, the action space of PD has only one continuous dimension, and it is also easy to achieve 100% action-level coverage. In this experiment, BW with four continuous actions is taken as an example to verify the effectiveness of action-level coverage. The results about ANOVA and correlation analysis pertaining to ABCov, KACov (k = 10), and OACov results are analyzed and discussed.

ANOVA results. Table Table 5 demonstrates whether the ANOVA assumptions are satisfied, including normality and homoscedasticity of the coverage result data. It also presents the F-statistic values, associated p-value1, Kruskal–Wallis statistic values, and corresponding p-value2.

For example, considering the results for ABCov1 on BW, the assumptions of normality and homoscedasticity are not satisfied, and the F-statistic value of 113.134 and the Kruskal–Wallis statistic value of 25.302 imply the significant differences in the means of coverage values

Table 3
The SBCov, KSCov (k = 10), and OSCov results of ANOVA.

Env.	ANOVA	SBCov1	SBCov2	KSCov	SACov	Single	Link	Cross	Mutual
CP	Normality	✗	✓	✗	✓	✗	✓	✓	✓
	Homoscedasticity	✓	✓	✓	✓	✗	✓	✓	✓
	F-statistic	1.806	1.776	14.554	19.279	2.868	4.626	3.545	3.69
	P-value1	0.164	0.169	0.000(**)	0.000(**)	0.050(*)	0.008(**)	0.024(*)	0.021(*)
	Kruskal–Wallis statistic	4.408	5.004	19.52	23.32	4.26	9.707	7.855	7.486
	P-value2	0.221	0.172	0.000(**)	0.000(**)	0.235	0.021(*)	0.049(*)	0.058
MC	Normality	✓	✓	✓	✓	✓	✓	✓	✓
	Homoscedasticity	✓	✓	✓	✓	✓	✓	✓	✓
	F-statistic	7.031	6.963	6.264	7.641	9.744	6.264	6.264	6.264
	P-value1	0.003(**)	0.004(**)	0.006(**)	0.002(**)	0.001(**)	0.006(**)	0.006(**)	0.006(**)
	Kruskal–Wallis statistic	9.95	10.602	9.302	12.398	12.347	9.302	9.302	9.302
	P-value2	0.007(**)	0.005(**)	0.010(**)	0.002(**)	0.002(**)	0.010(**)	0.010(**)	0.010(**)
PD	Normality	✗	✗	✗	✓	✗	✗	✓	✗
	Homoscedasticity	✓	✓	✗	✓	✓	✓	✓	✓
	F-statistic	130.441	111.65	36.207	457.796	2.131	2.5	56.476	42.845
	P-value1	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.153	0.116	0.000(**)	0.000(**)
	Kruskal–Wallis statistic	10	11.533	11.826	15.158	4.235	4.25	11.569	11.74
	P-value2	0.007(**)	0.003(**)	0.003(**)	0.001(**)	0.12	0.119	0.003(**)	0.003(**)
LL	Normality	✓	✗	✗	✗	✓	✗	✗	✗
	Homoscedasticity	✓	✓	✓	✗	✓	✓	✓	✓
	F-statistic	14.625	1.345	24.712	35.243	14.827	20.904	19.067	16.488
	P-value1	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)
	Kruskal–Wallis statistic	32.392	31.565	34.194	37.355	30.984	30.046	30.259	31.773
	P-value2	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)
BW	Normality	✗	✗	✓	✓	✓	✓	✓	✓
	Homoscedasticity	✓	✓	✗	✗	✓	✓	✓	✓
	F-statistic	34.622	9.135	421.686	449.318	4.096	5.979	49.128	11.792
	P-value1	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.021(*)	0.007(**)	0.000(**)	0.001(**)
	Kruskal–Wallis statistic	39.992	23.489	90.084	101.897	9.938	14.077	27.012	19.96
	P-value2	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.007(**)	0.001(**)	0.000(**)	0.000(**)

Note: ✓*indicates the assumption is satisfied and* ✗ *indicates the assumption is not satisfied. Significance levels:* $p < 0.05(*)$, $p < 0.01(**)$.

Table 4
The SBCov, KSCov (k = 10), and OSCov results and their respective Spearman correlations with rewards.

Env.	Alg.	SBCov1	SBCov2	KSCov	SACov	Single	Link	Cross	Mutual
CP	DQN	−0.468	−0.456	−0.273	−0.432	−0.372	−0.383	−0.358	−0.358
	Duel	−0.176	−0.176	−0.624	−0.6	−0.294	−0.515	−0.47	−0.527
	D3QN	−0.166	−0.166	−0.117	−0.092	−0.006	−0.062	−0.031	−0.031
	PPO	−0.576	−0.709	−0.486	−0.515	−0.591	−0.598	−0.413	−0.669
MC	DQN	−0.018	−0.018	−0.006	−0.067	−0.019	−0.006	−0.006	−0.006
	Duel	−0.018	−0.018	−0.025	−0.067	−0.069	−0.025	−0.025	−0.025
	D3QN	−0.032	−0.032	−0.032	−0.032	−0.136	−0.032	−0.032	−0.032
PD	DDPG	−0.395	−0.395	−0.029	−0.6	−0.257	−0.257	−0.257	−0.029
	SAC	−0.086	−0.086	−0.294	−0.657	–	–	−0.609	−0.676
	TD3	−0.353	−0.319	−0.725	−0.371	−0.068	−0.068	−0.235	−0.696
LL	DQN	−0.358	−0.545	−0.261	−0.236	−0.182	−0.498	−0.527	−0.358
	Duel	−0.821	−0.9	−0.6	−0.1	−0.7	−0.667	−0.9	−0.9
	D3QN	−0.127	−0.349	−0.772	−0.772	−0.182	−0.637	−0.81	−0.38
	PPO	−0.152	−0.01	−0.224	−0.285	−0.031	−0.049	−0.024	−0.079
	TD3	−0.061	−0.152	−0.067	−0.648	−0.209	−0.11	−0.152	−0.34
BW	PPO	−0.287	−0.214	−0.085	−0.329	−0.683	−0.789	−0.801	−0.695
	SAC	−0.257	−0.2	−0.6	−0.943	−0.657	−0.943	−0.829	−0.829
	TD3	−0.259	−0.484	−0.21	−0.37	−0.407	−0.469	−0.505	−0.556

among PPO, SAC, and TD3. The low p-value1 and p-value2 provide evidence against the null hypothesis, indicating that these differences are statistically significant.

Analogous to the above analysis, it is evident in Table Table 5 that each of the proposed coverage criteria exhibits significant differences when applied to different DRL models. This observation indicates that the proposed coverage criteria exhibit discriminative capabilities across various DRL models from an ANOVA perspective.

Correlation analysis results. Table Table 6 shows the results of Spearman correlation between coverage and rewards for each model considering the values of each run. For example, considering ABCov1 on PPO and SAC, the Spearman correlation values are −0.037 and −0.169, which is a negative correlation between SBCov1 results and rewards. For ABCov1 on TD3, the correlation coefficient cannot be calculated since the coverage results are the same value for each model considering the values of each run.

Table Table 6 reveals that the proposed coverage criteria exhibit a negative correlation between coverage values and rewards for each DRL model. This observation suggests that high coverage results correspond to low rewards and a higher likelihood of unexpected behavior detection, demonstrating the effectiveness of the proposed coverage criteria from the perspective of correlation analysis.

6.3. The results and analysis of state–action coverage

It follows from the definition of SACov that SACov results depend on the number of state–action pairs. SACov results have similar conclusions with KSCov results when the k value is certain since the number

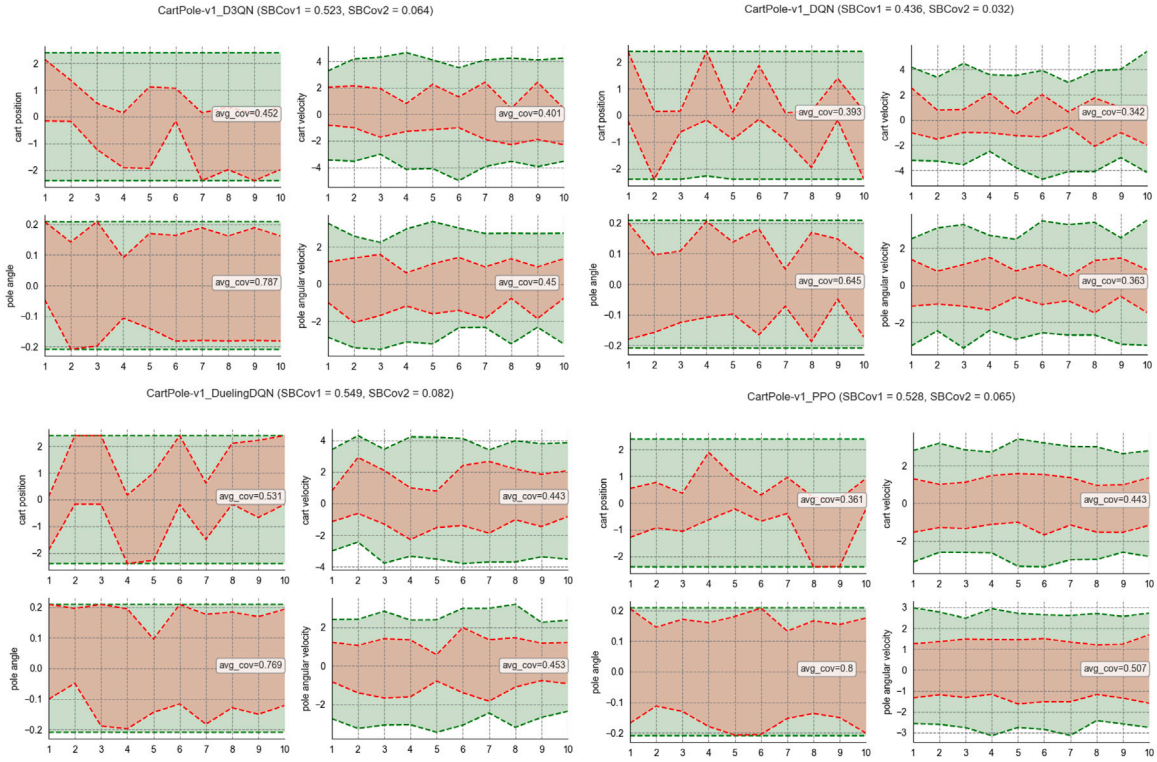

Fig. 5. The SBCov results of every state dimension for four models on CP. The horizontal axis represents the number of runs, and the vertical axis represents the exploration range of each state dimension, where the green and red regions are the exploration range of training and testing periods, respectively.

Table 5
The ABCov, KACov (k = 10), and OACov results of ANOVA.

Env.	ANOVA	ABCov1	ABCov2	KACov (k = 10)	Single	Link	Cross	Mutual
BW	Normality	✗	✗	✗	✗	✗	✗	✗
	Homoscedasticity	✗	✗	✗	✓	✓	✓	✗
	F-statistic	113.134	242.07	216.16	449.318	166.252	133.351	177.623
	P-value1	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)
	Kruskal–Wallis statistic	25.302	26.852	24.624	27.992	25.081	25.074	24.384
	P-value2	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)	0.000(**)

Note: ✓*indicates the assumption is satisfied and* ✗ *indicates the assumption is not satisfied. Significance levels:* $p < 0.01(**)$.

Table 6
The ABCov, KACov (k = 10), and OACov results and their respective Spearman correlations with average rewards.

Env.	Alg.	ABCov1	ABCov2	KACov(k = 10)	Single	Link	Cross	Mutual
BW	PPO	−0.037	−0.037	−0.354	−0.045	−0.346	−0.301	−0.351
	SAC	−0.169	−0.086	−0.486	–	–	–	–
	TD3	–	–	−0.235	–	−0.533	−0.533	−0.354

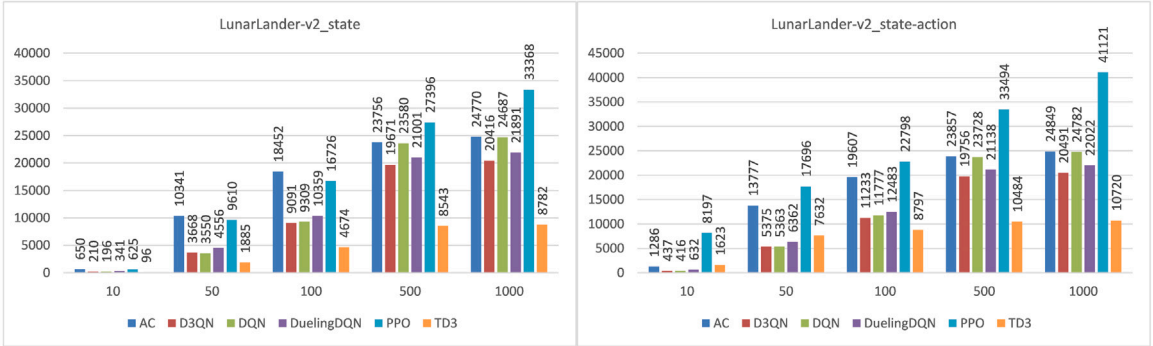

Fig. 6. The number of states and state–action pairs with different settings of k values.

Table 7
The low reward values and unexpected behaviors about the environments.

Env.	Alg.	Min values	Low reward values			Max values	Unexpected behaviors
			Below 3%	Below 10%	Below 20%		
CP	DQN	9	23.7	58.1	107.2	500	Failure to balance the pole
	Duel	9	23.7	58.1	107.2	500	Unstable actions
	D3QN	8	22.8	57.2	106.4	500	
	PPO	8	22.8	57.2	106.4	500	
MC	DQN	−200	−196.5	−188.3	−176.6	−83	Failure to reach the goal
	Duel	−200	−196.5	−188.3	−176.6	−83	Slow progress
	D3QN	−200	−196.5	−188.3	−176.6	−83	
PD	DDPG	−1687.8	−1637.2	−1519.1	−1350.5	−1.2	Failure to balance the pendulum
	SAC	−1861.0	−1805.2	−1674.9	−1488.9	−0.3	High energy
	TD3	−1868.7	−1812.6	−1681.8	−1495.0	−0.1	
LL	DQN	−5674.4	−5494.5	−5074.8	−4475.2	321.2	Crash or off-screen
	Duel	−2592.5	−2504.7	−2299.9	−2007.3	333.2	Fuel consumption
	D3QN	−2611	−2523.0	−2317.6	−2024.3	322.5	Incomplete landings
	PPO	−616.3	−591.1	−532.3	−448.4	223.4	
	TD3	−1877.6	−1813.2	−1662.7	−1447.7	271.9	
	AC	−651.9	−624.1	−559.3	−466.8	273.8	
BW	PPO	−189.1	−174.3	−139.7	−90.2	305.3	Falling or loss of balance
	SAC	−196.4	−180.8	−144.4	−92.5	323.0	High energy consumption
	TD3	−242.2	−225.6	−186.7	−131.1	313.3	Movement outside the bounds

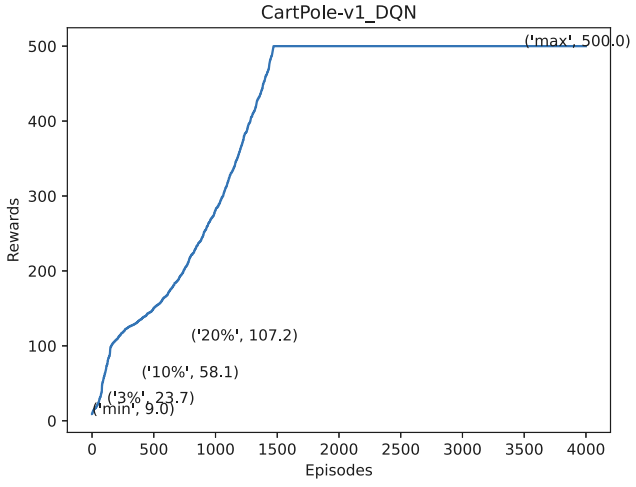

Fig. 7. Statistics of rewards about a DQN agent.

Fig. 8. Examples of unexpected behaviors related to low reward values.

of states is much larger than that of discrete actions. *It shows that SACov criteria are effective in detecting unexpected behaviors.*

Taking LL as an example, Fig. 6 shows the numbers of testing states and state–action pairs with different settings of k values. For example, state quantities with k value 50 are 16 times larger than those with k value 10 on AC, and they tend to be close when k values are 500 and 1000. It illustrates that state quantities increase with k values becoming larger and tend to be close when k values are larger enough.

Fig. 6 also shows the number of state–action pairs is more than that of states for every model, but none of them more than two times. However, if the agent can take all four discrete actions in a certain state, the state–action pairs should be four times that of states. That is, the number of state–action pairs is not m times that of states, where m represents the number of discrete actions in the action space. There are two possible reasons for this result: states limit the choice of actions, and accumulated experience does not require all actions as the transfer condition of the states.

6.4. A quantitative analysis of unexpected behaviors

Firstly, the specific method to determine low reward values is given, and examples of low reward values and unexpected behaviors are listed. Then, the relations between low reward values and coverage criteria are analyzed.

Examples of low reward values and unexpected behaviors. Taking CP as an example, a DQN agent was implemented to perform the DRL task, and the performance of the agent is evaluated by cumulated rewards per episode. Fig. 7 shows the ranking of episodic rewards from low to high over 4000 episodes. The ('10%', 58.1) represents the 10% corresponding to the episodic reward of 58.1 in 4000 episodes.

The approach for identifying low reward values relies on their closeness to the minimum of rewards, represented by 3%, 10%, and 20% in this paper, which consistently applies across other values in the experiment. It creates meaningful categories that represent different performance levels and provides a finer granularity in assessing the agent's performance, which is sufficient to evaluate the effectiveness of the proposed coverage criteria. Specifically, we consider 3%, 10%, and 20% above the minimum value as thresholds. The values respectively below these thresholds are defined as low reward values, i.e., $\min + (\max - \min) \times 3\%$, $\min + (\max - \min) \times 10\%$, and $\min + (\max - \min) \times 20\%$.

Table 7 shows more details about the low reward values of other experimental environments. The DQN agent can obtain over 400 scores on average and even obtains 500 scores in the most optimistic scenarios. However, it cannot be ignored that the agent performs poorly and gets low reward values in some cases. For example, the score is lower than 107.2 with a percentage of 20% and lower than 23.7 with a percentage of 3%, which are called low reward values.

Fig. 8 shows two unexpected behaviors on CP related to low reward values: 1. *Failure to balance the pole:* when the pole falls or the agent is unable to balance the pole within the specified time steps; 2. *Unstable Actions:* taking actions that lead to instability or cause the pole to deviate significantly from the vertical position. Next, we give a detailed explanation of the unexpected behaviors of other four environments.

– Two unexpected behaviors on MC: 1. *Failure to reach the goal:* If the agent fails to reach the flag within a certain number of time

Table 8
Analysis of low reward values and coverage criteria in every episode.

Env.	Alg.	Below 3%	Below10%	Below20%	SBCov1	KSCov(%)	SACov(%)	Single	Link	Cross	Mutual
CP	DQN	16	31	56	0.56	1.99	2.97	0.7	0.345	0.393	0.332
		11	18	37	0.499	1.11	1.56	0.6	0.245	0.28	0.238
		9	16	26	0.451	0.63	0.88	0.55	0.175	0.187	0.175
	Duel	13	29	44	0.62	1.76	2.57	0.7	0.35	0.367	0.31
		13	22	42	0.556	1.68	2.52	0.675	0.295	0.32	0.278
		8	18	34	0.535	1.07	1.51	0.6	0.265	0.283	0.248
	D3QN	41	51	61	0.625	2.15	3.1	0.725	0.375	0.413	0.358
		25	37	55	0.537	1.51	2.17	0.625	0.31	0.32	0.27
		11	29	54	0.504	1.17	1.74	0.6	0.25	0.283	0.253
	PPO	254	453	509	0.564	3.04	4.9	0.65	0.345	0.377	0.33
		251	448	502	0.558	2.32	3.96	0.625	0.345	0.393	0.293
		204	384	425	0.471	1.72	2.83	0.55	0.235	0.257	0.213
MC	DQN	208	220	256	0.938	0.68	0.89	0.95	0.68	0.68	0.68
		102	108	127	0.844	0.51	0.71	0.9	0.51	0.51	0.51
		97	102	118	0.809	0.48	0.66	0.9	0.48	0.48	0.48
	Duel	406	414	439	0.873	0.57	0.69	0.9	0.57	0.57	0.57
		207	217	242	0.753	0.39	0.55	0.85	0.39	0.39	0.39
		163	169	196	0.698	0.3	0.44	0.75	0.3	0.3	0.3
	D3QN	195	201	217	0.772	0.41	0.53	0.85	0.41	0.41	0.41
		173	184	213	0.739	0.37	0.46	0.8	0.37	0.37	0.37
		111	117	136	0.669	0.25	0.3	0.75	0.25	0.25	0.25
PD	DDPG	0	3	5	1	1.95	3.75	1	0.28	0.625	0.69
		0	2	3	0.993	1.77	3.38	1	0.28	0.61	0.68
		0	2	2	0.977	1.65	3.21	1	0.28	0.595	0.633
	SAC	1	2	5	0.998	1.91	6.42	1	0.28	0.63	0.697
		0	2	5	0.995	1.89	6.36	1	0.28	0.625	0.683
		0	2	4	0.99	1.91	6.08	1	0.28	0.62	0.683
	TD3	2	17	80	0.945	1.58	2.97	1	0.28	0.56	0.607
		2	15	80	0.939	1.55	2.76	0.967	0.28	0.555	0.607
		1	10	79	0.927	1.49	2.61	0.933	0.28	0.53	0.573
LL	DQN	1	1	1	0.494	2.73E−4	5.38E−4	0.35	0.075	0.074	0.093
		0	0	1	0.474	2.06E−4	4.29E−4	0.325	0.072	0.067	0.084
		0	0	0	0.42	1.17E−4	2.91E−4	0.287	0.067	0.063	0.068
	Duel	1	1	2	0.644	5.42E−4	9.62E−4	0.487	0.155	0.14	0.152
		0	0	1	0.576	3.15E−4	6.97E−4	0.438	0.13	0.113	0.115
		0	0	0	0.556	3.13E−4	6.83E−4	0.413	0.13	0.111	0.113
	D3QN	0	0	1	0.538	2.51E−4	5.2E−4	0.4	0.11	0.107	0.108
		0	0	0	0.509	2.5E−4	5.01E−4	0.375	0.107	0.101	0.102
		0	0	0	0.48	1.73E−4	4.15E−4	0.35	0.105	0.096	0.09
	PPO	2	2	7	0.665	7.33E−4	1.63E−3	0.55	0.253	0.249	0.199
		1	2	6	0.599	6.94E−4	1.67E−3	0.463	0.215	0.207	0.16
		0	0	3	0.592	3.99E−4	1.01E−3	0.45	0.198	0.181	0.142
	TD3	1	2	4	0.505	2.15E−4	1.92E−3	0.388	0.145	0.134	0.104
		1	1	2	0.432	1.3E−4	1.61E−3	0.325	0.098	0.091	0.079
		1	1	1	0.415	0.76E−4	1.38E−3	0.287	0.072	0.069	0.065
BW	PPO	1	1	96	0.877	7.67E−18	1.01E−17	0.808	0.325	0.339	0.383
		0	1	93	0.845	6.97E−18	8.57E−18	0.671	0.295	0.302	0.315
		0	0	85	0.835	6.14E−18	7.51E−18	0.658	0.282	0.29	0.304
	SAC	3	10	369	0.884	1.47E−18	1.85E−18	0.817	0.319	0.338	0.398
		2	9	219	0.851	1.41E−18	1.85E−18	0.679	0.279	0.301	0.314
		1	5	200	0.85	1.3E−18	1.65E−18	0.675	0.263	0.283	0.306
	TD3	2	7	112	0.803	1.54E−18	2.7E−18	0.646	0.212	0.223	0.227
		1	2	45	0.774	1.3E−18	1.56E−18	0.546	0.213	0.218	0.21
		0	1	22	0.761	1.03E−18	1.35E−18	0.525	0.205	0.216	0.199

steps or episodes, it receives a negative reward, typically 0. 2. *Slow progress:* If the agent makes slow progress or struggles to overcome the gravitational pull, it may receive low positive rewards, but still less than the maximum reward.

– Two unexpected behaviors on PD: 1. *Failure to balance the pendulum:* If the agent fails to balance the pendulum within a certain time limit, it receives a low negative reward. 2. *High energy consumption:* The agent may receive low negative rewards for applying high torque or energy to the pendulum. This discourages the agent from making overly aggressive or inefficient movements that waste energy.

– Three unexpected behaviors on LL: 1. *Crash or off-screen:* If the spacecraft crashes or goes off the screen, it receives a low negative reward. This penalizes unsafe landings or actions that lead to the spacecraft's destruction. 2. *Fuel consumption:* The agent may receive low negative rewards for excessive fuel usage. 3. *Incomplete landings:* If the spacecraft touches down on the ground but fails to land safely on the designated landing pad, it receives a low negative reward.

– Three unexpected behaviors on BW: 1. *Falling or loss of balance:* If the walker falls or loses balance, it receives a low negative reward. 2. *High energy consumption:* The agent may receive low negative rewards for using excessive energy to move. 3. *Movement outside the bounds:* If the walker moves outside the designated environment bounds or falls into hazardous areas, it receives a low negative reward.

Table 9
The results about the proposed strategies for test case selection.

k	Alg.	LL Env.				CP Env.			
		State	Transition	Random	Num.	State	Transition	Random	Num.
10	D3QN	**−15.56**	**−15.56**	−10.44	3	**352.16**	**352.16**	373.83	1
	DQN	166.8	**117.32**	175.94	2	**326.33**	357.08	384.83	1
	Dueling	**25.01**	**25.01**	45.25	2	234.83	**225.08**	251.91	1
	PPO	**84.54**	**84.54**	115.07	3	**429.66**	**429.66**	487.83	1
50	D3QN	**−179.13**	−145.12	104.29	10	352.16	350.91	**344.66**	2
	DQN	**112.58**	119.94	164.49	6	**326.33**	372.45	374.58	2
	Dueling	**30.01**	42.13	42.19	10	**218.52**	245.99	277.79	4
	PPO	54.9	**52.14**	90.53	23	**401.25**	447.77	485.22	3
100	D3QN	**−126.1**	−103.82	21.26	21	350.58	345.84	**338.54**	3
	DQN	**94.21**	95.33	145.13	14	369.75	361.67	**361.35**	3
	Dueling	41.66	**13.1**	71.16	34	269.64	265.95	**244.39**	8
	PPO	**52.53**	56.11	92.23	51	**418.23**	449.2	479.25	10
500	D3QN	**16.93**	31.42	62.08	81	**345.02**	354.06	358.59	50
	DQN	**143.5**	143.86	153.47	77	**337.85**	340.95	362.95	37
	Dueling	55.48	**54.16**	70.74	86	252.37	**250.28**	262.39	67
	PPO	**65.99**	67.3	87.04	70	**476.05**	477.55	477.51	99
1000	D3QN	**42.61**	44.38	55.22	89	**352.87**	357.78	359.58	68
	DQN	**148.76**	149.07	155.57	86	**343.09**	345.9	363.54	55
	Dueling	**60.42**	62.96	68.03	93	**253.48**	255.95	256.35	92
	PPO	**68.07**	69.67	91.87	73	477.69	477.69	477.69	100

The relations between low reward values and coverage criteria. For each DRL task, we conducted three runs of experiments, ensuring a decrease in the number of episodes with low reward values. The relations between the number of episodes with low reward values and the results of coverage criteria are presented in Table 8.

Taking the results for CP on DQN as an example, the number of episodes with low reward values below 3% is 16 in the first row, surpassing the counts in the second and third rows. Correspondingly, the results for SBCov, KBCov, SACov, Single, Link, Cross, and Mutual at this point are 0.56, 1.99, 2.97, 0.7, 0.345, 0.393, and 0.332, respectively. Importantly, these values are higher than those in the second and third rows. The conclusions drawn from episodes with low reward values below 10% and 20% yield similar findings.

The results reveal that in each experiment, the more the number of episodes with low reward values is, the greater the results of coverage criteria will be. That is, the proposed coverage criteria can effectively act as the measurement of the fact that test cases with high coverage can cover more unexpected behaviors.

6.5. The results of test case selection

Table 9 shows the average rewards of state sequences selected by the proposed test case selection and random selection strategies with different k values for CP and LL on D3QN, DQN, DuelingDQN, and PPO. The column labeled State represents the test case selection based on the state occurrence similarity metric, the column labeled Transition represents the test case selection based on the state transition similarity metric, and the column labeled Random represents random selection strategies. The column labeled Num. represents the number of selected state sequences in 100 episodes.

For the column labeled Transition, when the k value is 10, the average rewards of state sequences selected is 117.32 for LL on DQN, which is lower than that selected by the other two strategies. The number of selected state sequences with a similarity value of 0 is 2, and more state sequences can be selected with larger k values.

The data in bold is the lowest reward values of the three selection strategies in the same settings. From Table 9, we can observe that the average rewards of state sequences selected by the proposed test case selection are lower than those selected by random selection in most cases. It illustrates that two proposed test case selections can select state sequences with lower average rewards, and they are more effective than random selection.

7. Conclusions and future work

Coverage criteria as the primary consideration of software testing are necessary for the trustworthiness of software systems. To the best of our knowledge, we contribute the first exploration of the idea in coverage criteria for DRL systems considering the boundaries and quantities of states, actions, and policies to evaluate the testing adequacy. Furthermore, these coverage criteria are optimized by the idea of combinatorial coverage. Experiments demonstrate that the proposed coverage criteria are effective in unexpected behavior detection. Besides, state coverage provides a guideline for test case selection that makes DRL testing more effective than random selection.

In the near future, the proposed coverage criteria for different types of DRL systems will be performed. Based on it, the scenarios of utilizing the coverage criteria will be further presented.

CRediT authorship contribution statement

Ying Shi: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Beibei Yin:** Writing – review & editing, Visualization, Methodology, Conceptualization. **Zheng Zheng:** Writing – review & editing, Supervision, Methodology, Investigation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The code is added into the repository to allow the repeatability of experiments: https://github.com/stoneshadows/DRL-coverage.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62372021.

References

Al-Nima, R.R.O., Han, T., Al-Sumaidaee, S.A.M., Chen, T., Woo, W.L., 2021. Robustness and performance of deep reinforcement learning. Appl. Soft Comput. 105, 107295.

Bellman, R., 1957. A Markovian decision process. J. Math. Mech. 6 (5), 679–684.

Black, W.C., Babin, B.J., Anderson, R.E., 2010. Multivariate Data Analysis: A Global Perspective. Pearson.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W., 2016. Openai gym. arXiv preprint arXiv:1606.01540.

Brown, M.B., Forsythe, A.B., 1974. Robust tests for the equality of variances. J. Amer. Statist. Assoc. 69 (346), 364–367.

Chekam, T.T., Papadakis, M., Le Traon, Y., Harman, M., 2017. An empirical study on mutation, statement and branch coverage fault revelation that avoids the unreliable clean program assumption. In: 2017 IEEE/ACM 39th International Conference on Software Engineering. ICSE, IEEE, pp. 597–608.

Chilenski, J.J., Miller, S.P., 1994. Applicability of modified condition/decision coverage to software testing. Softw. Eng. J. 9 (5), 193–200.

Eniser, H.F., Gros, T.P., Wüstholz, V., Hoffmann, J., Christakis, M., 2022. Metamorphic relations via relaxations: An approach to obtain oracles for action-policy testing.

Fang, S., Chen, F., Liu, H., 2019. Dueling double deep Q-network for adaptive traffic signal control with low exhaust emissions in a single intersection. In: IOP Conference Series: Materials Science and Engineering, vol. 612, (no. 5), IOP Publishing, 052039.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M.G., Pineau, J., 2018. An introduction to deep reinforcement learning. arXiv preprint arXiv:1811.12560.

Fujimoto, S., Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. PMLR, pp. 1587–1596.

Gu, S., Holly, E., Lillicrap, T., Levine, S., 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: 2017 IEEE International Conference on Robotics and Automation. ICRA, IEEE, pp. 3389–3396.

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning. PMLR, pp. 1861–1870.

Jia, Y., Harman, M., 2010. An analysis and survey of the development of mutation testing. IEEE Trans. Softw. Eng. 37 (5), 649–678.

Kruskal, W.H., Wallis, W.A., 1952. Use of ranks in one-criterion variance analysis. J. Amer. Statist. Assoc. 47 (260), 583–621.

Kuhn, D.R., Mendoza, I.D., Kacker, R.N., Lei, Y., 2013. Combinatorial coverage measurement concepts and applications. In: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops. IEEE, pp. 352–361.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

Levine, S., Finn, C., Darrell, T., Abbeel, P., 2016. End-to-end training of deep visuomotor policies. J. Mach. Learn. Res. 17 (1), 1334–1373.

Li, J., Liu, Y., 2021. Deep reinforcement learning based adaptive real-time path planning for UAV. In: 2021 8th International Conference on Dependable Systems and their Applications. DSA, IEEE, pp. 522–530.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B., Chen, C., Su, T., Li, L., Liu, Y., et al., 2018a. DeepGauge: Multi-granularity testing criteria for deep learning systems. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. pp. 120–131.

Ma, L., Zhang, F., Xue, M., Li, B., Liu, Y., Zhao, J., Wang, Y., 2018b. Combinatorial testing for deep learning systems. arXiv preprint arXiv:1806.07723.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.

Nikanjam, A., Morovati, M.M., Khomh, F., Braiek, H.B., 2021. Faults in deep reinforcement learning programs: A taxonomy and a detection approach. arXiv preprint arXiv:2101.00135.

Odena, A., Olsson, C., Andersen, D., Goodfellow, I., 2019. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In: International Conference on Machine Learning. PMLR, pp. 4901–4911.

Pan, X., You, Y., Wang, Z., Lu, C., 2017. Virtual to real reinforcement learning for autonomous driving. arXiv preprint arXiv:1704.03952.

Pang, Q., Yuan, Y., Wang, S., 2022. Mdpfuzz: testing models solving Markov decision processes. In: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 378–390.

Patton, R., 2006. Software Testing. Pearson Education India.

Pei, K., Cao, Y., Yang, J., Jana, S., 2017. Deepxplore: Automated whitebox testing of deep learning systems. In: Proceedings of the 26th Symposium on Operating Systems Principles. pp. 1–18.

Scheffe, H., 1999. The Analysis of Variance, vol. 72, John Wiley & Sons.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Shapiro, S.S., Wilk, M.B., 1965. An analysis of variance test for normality (complete samples). Biometrika 52 (3/4), 591–611.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT Press.

Trujillo, M., Linares-Vásquez, M., Escobar-Velásquez, C., Dusparic, I., Cardozo, N., 2020. Does neuron coverage matter for deep reinforcement learning? A preliminary study. In: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops. pp. 215–220.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N., 2016. Dueling network architectures for deep reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 1995–2003.

Xie, X., Li, T., Wang, J., Ma, L., Guo, Q., Juefei-Xu, F., Liu, Y., 2022. NPC: N euron P ath C overage via characterizing decision logic of deep neural networks. ACM Trans. Softw. Eng. Methodol. 31 (3), 1–27.

Xie, X., Ma, L., Juefei-Xu, F., Xue, M., Chen, H., Liu, Y., Zhao, J., Li, B., Yin, J., See, S., 2019. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In: Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 146–157.

Zhu, H., Hall, P.A., May, J.H., 1997. Software unit test coverage and adequacy. ACM Comput. Surv. 29 (4), 366–427.

Ying Shi received an M.S. degree from the School of Mathematics and System Sciences at Beihang University (BUAA) in China. She is now a Ph.D. Student in School of Automation Science and Electrical Engineering, Beihang University. Her main research interests include software testing and software reliability.

Beibei Yin received a Ph.D. degree in control science and engineering from Beihang University (BUAA), Beijing, China, in 2010. She was a Research Scholar with the Department of Electrical and Computer Engineering, Duke University, Durham, NC, USA, in 2015. She is currently a Lecturer in Automation Science at Beihang University. Her main research interests include software testing, software reliability, and software cybernetics.

Zheng Zheng received a Ph.D. degree in computer software and theory from the Chinese Academy of Sciences, Beijing, China, in 2006. He is currently a Full Professor in Control Science and Engineering at the School of Automation Science and Electrical Engineering, Beihang University (BUAA), Beijing, China. In 2014, he was with the Department of Electrical and Computer Engineering at Duke University, working as a Research Scholar. His research interests include software dependability, unmanned aerial vehicle path planning, artificial intelligence applications, and software fault localization.