Stability-Aware Reinforcement Learning for Robust Class Integration Test Order Generation

Yanru Ding[1,2], Yanmei Zhang[1,2*], Guan Yuan[1,2*], Shujuan Jiang[1,2], Wei Dai[3], Luciano Baresi[4]

[1]School of Computer Science and Technology/School of Artificial Intelligence, China University of Mining and Technology, Xuzhou 221116, China

[2]Mine Digitization Engineering Research Center of the Ministry of Education, China University of Mining and Technology, Xuzhou 221116, China

[3]School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China

[4]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano 20133, Italy

{yrding, ymzhang, shjjiang, yuanguan, weidai}@cumt.edu.cn, luciano.baresi@polimi.it

Abstract

Generating a class integration test order (CITO) is essential to reduce the overhead of test stub construction (the primary cost in integration testing) and to ensure system reliability in complex software systems. Although reinforcement learning (RL) has shown promise in automating CITO generation, existing methods suffer from unstable policy learning and limited robustness against structural perturbations and defect injection. These challenges stem from insufficient reward shaping and the lack of reliable oracles for validation. To address these limitations, we propose LM-CITO, a stability-aware RL framework that integrates Lyapunov-guided reward shaping with semantic validation through metamorphic testing (MT). Specifically, we design a Lyapunov energy function over class dependency graphs to promote monotonic structural convergence during training, and define metamorphic relations (MRs) to verify behavioral consistency under controlled perturbations. Extensive experiments on six real-world systems demonstrate that LM-CITO consistently produces more effective policies, yielding CITOs with significantly reduced stubbing costs compared to baseline models. Furthermore, MT verifies the capability of our MRs to detect defects in 19 injected bug variants, confirming the robustness of LM-CITO under various fault-induced perturbations. These results highlight the synergy of stability guidance and MR-based validation, offering an effective, principled solution for oracle-free RL in software testing.

Introduction

Generating a class integration test order (CITO) is a core task in object-oriented integration testing, aiming to determine a cost-efficient execution sequence for testing interdependent classes while minimizing the need for test stubs. However, real-world software systems frequently exhibit cyclic dependencies that hinder topological sorting and increase the overhead of stub construction. Addressing this challenge requires optimizing test orders under complex class dependency graphs (Jiang et al. 2021). With the development of AI techniques (Zhang, Zhang, and Yuan 2024; Zhang et al. 2025a), reinforcement learning (RL) has been investigated as a means to generate CITOs (Czibula, Czibula, and Marian 2018; Ding et al. 2023a, 2025). Through interaction with the test environment, RL agents acquire integration strategies that respect dependency structures.

Despite this progress, the practical effectiveness of RL-based CITO methods remains constrained by two key challenges. First, RL in this context suffers from inherently sparse and delayed reward signals. Since meaningful feedback is often available upon completion of an entire test sequence, effective credit assignment to intermediate decisions becomes challenging. This weak reward shaping frequently results in unstable training dynamics characterized by erratic policy updates and inconsistent test orderings —especially in the presence of injected defects or structural perturbations (Klein et al. 2024). Second, the absence of a definitive oracle complicates the validation. Semantically valid input perturbations (e.g., adding a dependency) can lead to divergent but equally plausible test orders. Without a unique ground truth, reliably assessing correctness is non-trivial and undermines confidence in the learned policies (Zhang et al. 2021a). These challenges are exemplified in Figure 1.

To address these challenges, we integrate two theoretically grounded techniques that directly target the core limitations of RL-based CITO generation. First, to mitigate policy instability caused by sparse and delayed rewards, we design a Lyapunov-guided reward shaping mechanism that encourages monotonic structural convergence during training. This control-theoretic approach regularizes policy trajectories and enforces stability without relying on manual heuristics (Gajic and Qureshi 2008; Zhang et al. 2025b). Second, to address the absence of a definitive oracle, we adopt metamorphic testing (MT) (Chen, Cheung, and Yiu 1998; Xie et al. 2020) to assess the semantic consistency of generated CITOs under controlled structural perturbations. By defining domain-specific metamorphic relations (MRs) over input class diagrams, we establish an oracle-free criterion to evaluate the robustness of RL-based CITO generation.

Building on these techniques, we propose LM-CITO (Lyapunov-guided reward design and Metamorphic testing for Class Integration Test Order generation), a stability-

*These authors contributed equally.

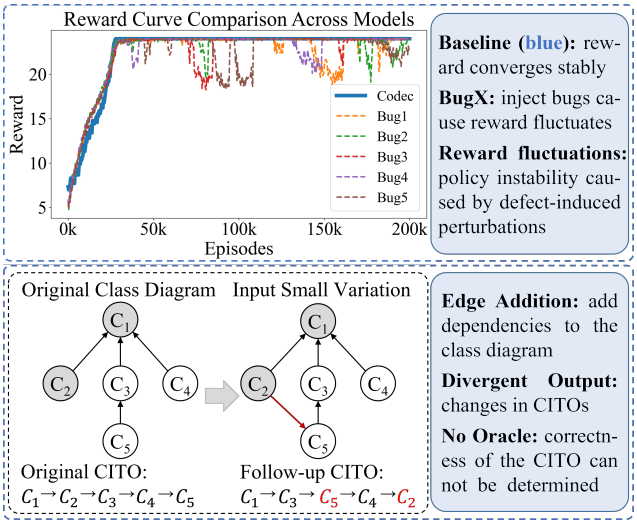

Figure 1: Challenges in RL-based CITO generation. Codec is the program under test and the original convergence curve; BugX curves represent convergence after defect injection.

aware RL framework that combines Lyapunov-guided reward shaping with MT-based semantic validation. Our key contributions are as follows:

- We introduce LM-CITO, the first CITO generation framework that leverages Lyapunov stability theory to guide policy convergence and employs MT to semantically validate robustness under structural perturbations.
- We develop a suite of domain-specific MRs that enable oracle-free verification of semantic consistency, supporting systematic evaluation against injected defects and realistic input mutations.
- We perform extensive experiments with six real-world software projects and 19 injected bug variants, showing that LM-CITO consistently achieves lower stubbing cost, higher training stability, and stronger defect detection capability than state-of-the-art baselines.

Related Work

A core challenge in CITO generation is managing cyclic dependencies among classes, which hinder topological sorting and complicate sequencing. Classical graph-based methods (Kung et al. 1995b; Le Traon et al. 2000; Briand, Labiche, and Wang 2001) address this by heuristically removing weak dependencies to create acyclic subgraphs, enabling valid test orders. Extensions incorporate class similarity (Zhang et al. 2021b) and execution-aware models (Meng et al. 2022) to improve ordering precision. Zhang et al. (Zhang et al. 2021a) applied MT to evaluate the robustness of CITO algorithms under structural perturbations, designing domain-specific MRs and measuring stability via order consistency across inputs.

RL has emerged as a data-driven alternative to heuristic approaches. Early work (Czibula, Czibula, and Marian 2018) used Q-learning to reduce stubbing costs; subsequent studies incorporated class importance into reward design (Ding et al. 2023a) and improved policy efficiency through dueling A2C networks (Zhang et al. 2023). More recent efforts (Ding et al. 2025) enriched state representations with fault history for enhanced test order guidance. However, RL-based CITO methods face persistent issues, such as policy instability, due to sparse rewards and lack of reliable validation oracles (Klein et al. 2024; Zhang et al. 2021a). Our approach builds upon these foundations with a theoretically-grounded and empirically-validated solution.

Preliminaries

This section presents the background of the CITO generation problem and its evaluation indicators.

Inter-Class Dependencies

In integration testing, inter-class dependencies are the main cause of test stubs, that is, artificial components that simulate untested classes and inflate testing cost (Briand, Labiche, and Wang 2003). Dependencies are static (e.g., associations, inheritance) or dynamic (e.g., polymorphic calls) (Briand, Labiche, and Wang 2001; Le Traon et al. 2000), each posing distinct simulation challenges. Effective CITO generation requires modeling these dependencies to guide ordering strategies that minimize stub overhead.

Stubbing Cost Metrics

The CITO problem aims to minimize the stubbing cost, traditionally measured by counting the stubs: a clear but coarse metric (Kung et al. 1995a). Since breaking multiple weak dependencies may be cheaper than breaking one strong dependency, a finer metric is needed. A refined stubbing complexity metric quantifies the cost based on three normalized coupling types (Ding et al. 2023b): attribute coupling $A(i,j)$, which counts how often class j appears as a type in the attributes or parameters of class i; method coupling $M(i,j)$, which captures calls from i to j; and dynamic coupling $D(i,j)$, which accounts for polymorphic invocations from i to methods in j that are overridden in i, where j may be a superclass, abstract class, or interface. Together, these components estimate the number of elements that must be simulated in the stub. All three metrics are normalized prior to aggregation. For simplicity, we use k to represent an edge (static or dynamic) between classes i and j, and then the stubbing complexity between classes i and j can be represented by $SCplx(k)$ and computed as:

$$SCplx(k) = \sqrt{\omega_A \overline{A(k)}^2 + \omega_M \overline{M(k)}^2 + \omega_D \overline{D(k)}^2} \quad (1)$$

where ω_A, ω_M, and ω_D are the weights of the three dependencies, and their sum must be equal to 1.

Let $o = [c_1, c_2, ..., c_n]$ denote a CITO, where each c_i is a class to be tested. The overall stubbing complexity of o, denoted as $OCplx(o)$, is calculated as:

$$OCplx(o) = \sum_{k \in d(o)} SCplx(k) \quad (2)$$

where $d(o)$ denotes the set of edges in the system that are broken due to the class execution order defined by o.

Methodology

LM-CITO integrates two key modules as shown in Figure 2. First, a Lyapunov-guided reward for promoting stable class order generation. Second, a metamorphic verification process for semantic robustness checking. Policy learning is implemented using the Advantage Actor-Critic (A2C) algorithm (Mnih et al. 2016).

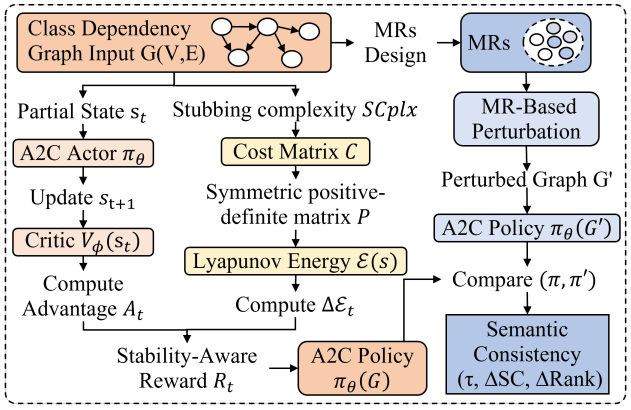

Figure 2: Overview of the LM-CITO framework.

Lyapunov-Guided Reward Mechanism

Problem Formulation. To effectively apply RL to the generation of CITOs, we formulate CITO generation as a Markov Decision Process (MDP) (Puterman 1990), where each episode constructs a complete CITO. In step t, the agent observes a partial ordering state s_t, selects the next untested class a_t, and receives a reward R_t. The goal is to learn a policy $\pi(a|s)$ maximizing the expected cumulative reward. We adopt a position-encoded state representation (Zhang et al. 2023), with $s_1 = [-1, -1, \ldots, -1]$ indicating that no class is selected. The selection of class c updates $s[c]$ to its position in the sequence; this is repeated until all classes are ordered.

Beyond minimizing the stubbing cost, prioritizing critical classes during CITO generation is essential for early fault detection and program stability (Ding et al. 2023a; Wang et al. 2021). To balance cost-efficiency and fault prioritization, the reward function integrates three normalized components (Ding et al. 2025): (i) net revenue (NR_i), which is the net gain from selecting the class i calculated as avoided minus incurred stubbing costs; (ii) class importance (Imp_i), which is a unified metric that captures structural complexity and integration impact through influence propagation over dependencies; (iii) error probability (EP_i), that is, the estimated likelihood of faults in class i, derived from historical bug data or coverage statistics. The step-wise reward for selecting class i at position σ_i is defined as:

$$r(\sigma_i) = \begin{cases} min, & \text{if } i \text{ invalid} \\ c(NR'_i + \frac{1}{2}(Imp'_i + DP_i)), & \text{if in progress} \\ max + \frac{c}{SCplx_i}, & \text{if done} \end{cases} \quad (3)$$

where c balances the reward terms. Invalid actions incur a minimal penalty to prevent illegal selections. This formulation incentivizes the early selection of important, fault-prone, and cost-effective classes in CITO generation.

We incorporate this reward scheme into the Advantage Actor-Critic (A2C) algorithm (Mnih et al. 2016), where policy $\pi_\theta(a \mid s)$ is parameterized by the actor network and updated via the advantage function $A(s,a) = Q(s,a) - V_\phi(s)$. The value function $V_\phi(s)$ is estimated by the critic network to reduce the variance in the policy gradients. Actions are sampled from the available class candidates. This framework stabilizes policy updates by leveraging value estimates, enabling efficient learning in complex decision spaces.

The Cost Matrix–based Lyapunov Energy Function. A Lyapunov function $L(x)$ is a positive definite scalar function that decreases monotonically along system trajectories. For discrete-time systems, if $\Delta L(x) = L(x_{t+1}) - L(x_t) \leq 0$, the system is stable; if $\Delta L(x) < 0$, it is asymptotically stable (Gajic and Qureshi 2008).

Inspired by this, we define an energy function $\mathcal{E}(s_t)$ on the selection states of classes to capture structural stability. Given a class dependency graph, we construct a cost matrix $C \in \mathbb{R}^{n \times n}$, where $C_{i,j}$ denotes the stubbing complexity between classes C_i and C_j. A symmetric positive-definite matrix is derived as:

$$P = C + C^\top + \epsilon I, \quad (4)$$

where $\epsilon > 0$ ensures positive definiteness, avoids numerical singularities, and guaranties a well-defined quadratic form. The system energy at step t, is defined as:

$$\mathcal{E}(s_t) = s_t^\top P s_t, \quad (5)$$

which quantifies the structural complexity or instability encoded by the current partial order s_t. The relative change in energy between consecutive steps is defined as:

$$\Delta\mathcal{E}_t = \frac{\mathcal{E}(s_{t+1}) - \mathcal{E}(s_t)}{\mathcal{E}(s_t) + \delta}, \quad (6)$$

where δ is a small constant added for numerical stability.

In classical Lyapunov stability theory, a negative energy increase $\Delta\mathcal{E}_t < 0$ implies a stabilizing step (Gajic and Qureshi 2008). However, in our setting, state s_t represents an accumulative selection of classes, where each newly added class C_k at step $t+1$ irreversibly increases structural complexity. With $s_{t+1} = s_t + e_k$, where e_k is the one-hot vector for the selected class, the energy increment satisfies:

$$\Delta\mathcal{E}_t = \mathcal{E}(s_{t+1}) - \mathcal{E}(s_t) = 2s_t^\top P e_k + e_k^\top P e_k \geq 0, \quad (7)$$

implying energy increases monotonically with ordering.

This deviation from classical assumptions arises because our state space encodes accumulative decisions rather than freely evolving system states. Consequently, large positive increases in $\Delta\mathcal{E}_t$ reflect costly or abrupt structural changes rather than stability. To mitigate this, we incorporate a bounded hyperbolic tangent transformation of $\Delta\mathcal{E}_t$ into the reward function:

$$R_t = R_{\text{task}} + \lambda \cdot \tanh(\Delta\mathcal{E}_t) \cdot w_{\text{stab}}(t), \quad (8)$$

where λ controls the Lyapunov term's influence and $w_{\text{stab}}(t)$ is a time-dependent weight that encourages long-term stability. The function $\tanh$ suppresses excessive gradient magnitudes, stabilizing policy updates under the A2C framework.

Finally, since P is symmetric positive-definite, the energy function satisfies:

$$\mathcal{E}(s_t) > 0 \quad \text{for } s_t \neq 0, \quad \mathcal{E}(0) = 0, \tag{9}$$

which ensures positive definiteness, a necessary condition for Lyapunov functions.

In summary, although the classical condition $\Delta\mathcal{E}_t \leq 0$ for stability does not strictly hold due to the cumulative nature of our state representation, the Lyapunov-guided reward still penalizes large unstable jumps, guiding the agent toward generating smooth and structurally coherent class orderings.

Action and State Aggregation Strategies RL in CITO generation often involves large and redundant action and state spaces, which can hinder efficient exploration. To mitigate this, we propose two aggregation strategies at both action and state levels.

(1) **Action Aggregation** groups classes that exhibit identical dependency profiles, based on the Identical Class Dependency (ICD) principle (Zhang et al. 2021b). Classes that share the same sets of incoming and outgoing dependencies are clustered and treated as a single aggregated action. This reduces the size of the action space from the original number of classes to a smaller number of ICD clusters, thereby reducing redundancy and simplifying policy learning.

(2) **State Aggregation** leverages the critic's value estimates to identify behaviorally similar states. Instead of raw state features, we monitor the critic's value function over a sliding window and aggregate states whose value fluctuations fall below a preset threshold, indicating comparable long-term outcomes. The value of the aggregated state is approximated by averaging the values of individual states'. To ensure stability, state aggregation is activated only after the training process has progressed beyond 90% of episodes, balancing convergence speed, computational stubbing cost, and final policy quality.

MT for CITO Validation

To address the oracle problem in RL-based CITO generation, we adopt MT (Chen, Cheung, and Yiu 1998) as an oracle-free validation framework. Zhang et al. (Zhang et al. 2021a) treat MRs as the foundation of an oracle-free test suite, where each MR defines a semantic-preserving transformation and a corresponding expectation. We extend this principle by pairing each MR with quantitative evaluation metrics and threshold constraints, enabling interpretable and scalable robustness validation.

In this paper, we designed 11 MRs grouped into four categories: Structural Perturbation (T1), Importance Variation (T2), Error Propagation (T3), and Path Dependency Disturbance (T4). Each MR defines a deterministic transformation MR : $ctx \mapsto ctx'$ on the abstract environment context, including the class dependency graph, importance scores, error rates, and semantic clusters. The trained RL agent produces output sequences π and π' on the original context ctx and the perturbed context ctx', respectively. Behavioral differences between π and π' are evaluated against semantic expectations summarized in Table 1.

The last column of Table 1 summarizes the expected behavioral outcomes under perturbation. These expectations determine whether the observed deviation under perturbation is acceptable. All indicators are normalized, with lower values indicating semantic consistency and higher values reflecting deviation. The formal definitions of these metrics are provided below.

(1) **Kendall Tau Distance** is denoted as $\tau(\pi, \pi')$ and measures the ranking divergence between the original order π and the perturbed order π':

$$\tau(\pi, \pi') = \frac{2 \cdot \text{Discounted Pairs}}{n(n-1)} \times 100\%, \tag{10}$$

where n is the number of classes and *Discounted Pairs* denotes the number of pairs of classes whose relative order is reversed. A higher τ indicates greater output instability.

(2) **Cost Perturbation** quantifies the difference in execution costs using two metrics: stub complexity change $\Delta OCplx$ and stub count change $\Delta Stubs$:

$$\Delta OCplx = \frac{|OCplx(\pi') - OCplx(\pi)|}{\max(OCplx(\pi), \epsilon)}, \tag{11}$$

$$\Delta Stubs = \frac{|Stub(\pi') - Stub(\pi)|}{\max(Stub(\pi), \epsilon)}. \tag{12}$$

Here, $OCplx(\cdot)$ denotes the total complexity of the required stubs, and $Stub(\cdot)$ is the number of stubs to be generated. A small constant $\epsilon = 10^{-8}$ avoids division by zero.

(3) **Positional Perturbation** tracks ranking shifts of critical class subsets include important and error-prone classes:

$$\Delta\text{ImpPos} = \frac{1}{|\mathcal{I}|} \sum_{C_i \in \mathcal{I}} |rank(\pi', C_i) - rank(\pi, C_i)|, \tag{13}$$

$$\Delta\text{ErrPos} = \frac{1}{|\mathcal{E}|} \sum_{C_i \in \mathcal{E}} |rank(\pi', C_i) - rank(\pi, C_i)|, \tag{14}$$

where $\mathcal{I}$ denotes the top 50% important classes based on the RL agent's attention, $\mathcal{E}$ represents the error-prone classes identified by fault profiles, and $\text{rank}(\pi', C_i)$ gives the position of class C_i in the perturbed order π'.

(4) **Reward Perturbation** measures the difference in reward between the perturbed order π' and the original π:

$$\Delta R = R(\pi') - R(\pi). \tag{15}$$

This examines whether the reward change aligns with the intended semantic effect of MR ΔR.

High values of these metrics indicate significant behavioral divergence between original and perturbed outputs. To assess semantic acceptability, these deviations are compared against thresholds δ_{kendall}, δ_{imp}, δ_{reward}, and δ_{error}, calibrated for each MR type, typically ranging from 0.2 to 0.6. Together, these MR-metric pairs establish a principled, oracle-free validation framework that enables structured, reward-aware, and ranking-sensitive robustness evaluation of RL-based CITO generation.

Experiments

Experimental Setup and Evaluation Metrics

Subject Programs. We evaluated LM-CITO on six Defects4J programs (Just, Jalali, and Ernst 2014), selected

Type	MR ID	Description	Operation	Expected Judgment
T1	MR1.1	Add edges between important classes	Increase cohesion	$\Delta OCplx \geq 0$, $\Delta Stubs \geq 0$
	MR1.2	Add unrelated class	Noise robustness	$\Delta OCplx \geq 0$, $\Delta Stubs \geq 0$
	MR1.3	Merge similar classes	Interface compatibility	$\tau(\pi, \pi') < \delta_{\text{kendall}}$
	MR1.4	Add cycle edges	Handle cyclic dependencies	$\Delta\text{ImpPos} < \delta_{\text{imp}}$, $\tau(\pi, \pi') < \delta_{\text{kendall}}$
T2	MR2.1	Boost low-importance class	Rank responsiveness	$\tau(\pi, \pi') < \delta_{\text{kendall}}$
	MR2.2	Scale importance-based reward	Reward consistency	$\Delta I \cdot \Delta R \geq 0$, $\lvert\Delta R\rvert < \delta_{\text{reward}}$
T3	MR3.1	Connect high-error class	Fault propagation	$\tau(\pi, \pi') < \delta_{\text{kendall}}$, $\Delta R < \delta_{\text{reward}}$
	MR3.2	Extreme error injection	Fault tolerance	$\Delta\text{ErrPos} > \delta_{\text{error}}$ or $\tau(\pi, \pi') > \delta_{\text{kendall}}$
T4	MR4.1	Random edge mutation	Path disruption	$\tau(\pi, \pi') > \delta_{\text{kendall}}$, $\Delta OCplx \geq 0$
	MR4.2	Node reordering	Index invariance	$\tau(\pi, \pi') > \delta_{\text{kendall}}$
	MR4.3	Connect super node	Centralization robustness	$\Delta OCplx \geq 0$

Table 1: Categorization and operational expectations of MRs.

based on: (1) availability of faulty class annotations for semantic validation, (2) diversity in class scale (from 26 to 171 classes) and dependency complexity (1–229 cycles), and (3) coverage of both acyclic and cyclic integration structures. Table 2 summarizes the key statistics of these programs.

Programs	Classes	Deps	Cycles	Faulty
Codec	26	59	1	7
JacksonXml	48	68	3	4
Compress	66	117	1	16
Gson	137	447	8	3
Lang	170	245	155	21
Jxpath	171	628	229	19

Table 2: Subject programs overview.

Training Setup. All models adopt the A2C algorithm with $\alpha = 0.0001$, $\gamma = 0.8$, and $200,000$ episodes. We build on stable-baselines3 (Raffin et al. 2021), integrating task-specific reward shaping and Lyapunov stability terms.

Competitors. We compare LM-CITO with DAO-CITO, a variant of CD-CITO (Ding et al. 2025) enhanced with action and state aggregation. DAO-CITO serves as our baseline, allowing a controlled evaluation of Lyapunov-guided reward and MT-based validation.

CITO Evaluation Metrics. We evaluate CITO quality from two perspectives: (1) *Stubbing cost* as shown in Section **Stubbing Cost Metrics**, including stub count ($Stubs$) and overall stubbing complexity ($OCplx$); (2) *Class prioritization*, measuring how early critical classes appear —and lower values are better. Specifically:

$$\text{AR}_{50\%} = \frac{1}{n/2} \sum_{i=1}^{n/2} \text{Rank}(T, C_i), \tag{16}$$

$$\text{AER} = \frac{1}{m} \sum_{i=1}^{m} \text{rank}(T, E_i), \tag{17}$$

where C_i are the top 50% important classes, E_i are error-prone classes, and T is the predicted test order.

MT Evaluation Metrics. To assess robustness, we adopt the mutation score (MS) (Zhang et al. 2021a), defined as:

$$MS = \frac{N_k}{N_m - N_e}, \tag{18}$$

where N_k is the number of mutants killed, N_m is the total number of mutants, and N_e the number of equivalent ones. A mutant is considered killed if the post-perturbation policy π' exhibits significant deviations in one or more semantic consistency metrics $\psi(\pi')$, including Kendall Tau distance, cost perturbation, positional perturbation, or reward perturbation, beyond predefined thresholds.

Perturbations and Experimental Scale. To evaluate robustness, we consider two types of perturbation: (1) *Bug-injected models*, where 19 training bugs (Zhang et al. 2025b) simulate five types of faults including: input processing, network functions, A2C core logic, policy structure, and loss configuration. (2) *MRs* as defined in Table 1, which apply semantically consistent structural transformations to the environment to verify that the behavior of the RL agent remains stable or changes according to expected semantic patterns.

Each system is tested with 21 models (1 LM-CITO, 1 DAO-CITO as baseline, 19 buggy variants), 11 MRs, 10 mutation rounds per MR, and 30 CITOs per round. This yields approximately 69,300 test cases per system.

Main Results and Discussion

Comparative Robustness Analysis of LM-CITO. To assess the influence of the Lyapunov mechanism on CITO generation, we compare three methods: CD-CITO (the state-of-the-art RL-based method) (Ding et al. 2025), DAO-CITO (CD-CITO enhanced with dual aggregation of action and state), and LM-CITO. Each experiment on the six programs was repeated 30 times, and the averages were reported.

Table 3 reports the stubbing cost metrics $Stubs$ and $OCplx$, and the critical class metrics $AR_{50\%}$ and AER. LM-CITO achieves the lowest values for $Stubs$ in 3 out of 6 programs and the lowest value for $OCplx$ in 5 out of 6, demonstrating its effectiveness in reducing the cost of stubbing during CITO generation. This also reflects that multiple simpler stubs may incur lower overall structural com-

Programs	Method	Stubs	OCplx	$AR_{50\%}$	AER
Codec	CD-CITO	7	0.48	15.38	10.86
	DAO-CITO	15	0.38	**14.62**	**9.86**
	LM-CITO	**6**	**0.3**	15.59	14.44
JacksonXml	CD-CITO	9	0.71	31.04	29.28
	DAO-CITO	7	0.49	**28.19**	**27.56**
	LM-CITO	**4**	**0.17**	29.19	35.1
Compress	CD-CITO	**33**	2.52	66.06	21.06
	DAO-CITO	49	1.23	65.39	**18**
	LM-CITO	46	**1.1**	**62.72**	57.88
Gson	CD-CITO	**8**	0.47	18.42	31.25
	DAO-CITO	**8**	0.47	17.17	**15.25**
	LM-CITO	**8**	**0.31**	**16.12**	17.27
Lang	CD-CITO	**42**	1.68	78.82	**62.2**
	DAO-CITO	67	1.99	**76.4**	64.88
	LM-CITO	53	**1.3**	77.28	82.31
Jxpath	CD-CITO	**58**	**2.45**	**70.32**	**64.73**
	DAO-CITO	112	3.48	74.33	81.49
	LM-CITO	104	2.77	71.13	90.74

Table 3: Comparison of testing cost and sorting metrics.

plexity than a single entangled one; hence, we emphasize $OCplx$ as the most indicative cost measure. The results reveal a trade-off: LM-CITO excels at reducing $OCplx$ but shows less consistent gains in $AR_{50\%}$, with only two values outperformed by DAO-CITO. Stubbing cost reduction relies mainly on modeling complex dependencies, while AER depends on historical fault data, which LM-CITO does not explicitly optimize. For more balanced metric performance, DAO-CITO may be preferred. Prioritizing stub cost reduction, the Lyapunov-guided reward of LM-CITO focuses on overall stubbing complexity over ranking.

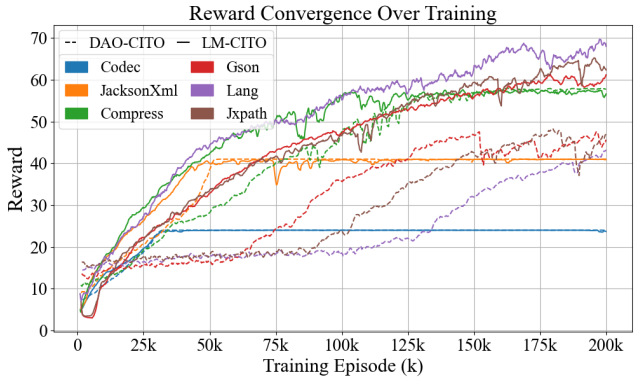

Figure 3: Comparison of reward convergence.

The reward convergence curves in Figure 3 show that LM-CITO (solid lines) consistently achieves faster and more stable improvements than DAO-CITO (dashed lines) on Compress, Gson, Lang, and Jxpath, indicating increased training efficiency and reduced policy variance. For Codec and JacksonXml, both methods converge similarly, likely due to simpler dependency structures limiting the impact of Lyapunov guidance. In general, LM-CITO's integration of structural stability yields more robust and efficient optimization.

Crucially, LM-CITO's Lyapunov reward incorporates the cost matrix as a negative guidance signal, effectively steering policy optimization towards minimizing OCplx. This shaping also promotes more efficient and stable training convergence, improving the speed and stability of the learning process, and demonstrating the advantage of LM-CITO in enhancing training dynamics for CITO generation.

Robustness of LM-CITO under MT. To evaluate the robustness of LM-CITO under the perturbations induced by 11 MRs, we first group the 19 bugs into five defect categories: input processing, network functions, A2C core logic, policy structure, and loss configuration, and then measure the MS of the MRs across these bugs. The results are shown in Table 4 as (mean, standard deviation). A higher mean indicates stronger fault detection, while a smaller deviation reflects more consistent performance across defect types.

In Table 4, MR1.3, MR1.4, MR2.2, and MR3.1 consistently achieve the highest MSs with scores around 0.8, demonstrating strong and broad fault detection, especially under structural merging, cycle injection, reward scaling, and fault propagation, critical scenarios in RL testing. In contrast, MR3.2, MR4.1, and MR4.2 yield lower MSs ranging from 0.2 to 0.3, indicating weaker responsiveness to perturbations such as extreme random faults or class reordering. Despite standard deviations mostly below 0.4, overall trends remain stable in all bug categories.

MRs	Input	Network	A2C	Policy	Loss
MR1.1	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)
MR1.2	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)	(0.5, 0.3)	(0.4, 0.3)
MR1.3	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)
MR1.4	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)
MR2.1	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)
MR2.2	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)
MR3.1	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)	(0.8, 0.4)
MR3.2	(0.2, 0.4)	(0.2, 0.4)	(0.2, 0.4)	(0.2, 0.4)	(0.2, 0.4)
MR4.1	(0.2, 0.4)	(0.2, 0.4)	(0.2, 0.4)	(0.2, 0.4)	(0.2, 0.4)
MR4.2	(0.3, 0.4)	(0.3, 0.4)	(0.3, 0.4)	(0.3, 0.4)	(0.3, 0.4)
MR4.3	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)	(0.4, 0.2)

Table 4: MSs of MRs under defect categories.

Figure 4 complements this analysis by presenting a heat map of the mean MSs in three variants of the model: injected bug models, DAO-CITO, and LM-CITO. Darker cells indicate greater MR sensitivity. For Codec, all three models are vulnerable to MR3.2 and MR4.1, revealing sensitivity to fault propagation and stochastic path disruptions. Both DAO-CITO and LM-CITO exhibit limited resistance to MR4.2, suggesting insufficient index invariance. LM-CITO shows slightly reduced variance, indicating marginal but incomplete robustness against T4-type perturbations. For JacksonXml and Compress, the bug models respond mainly to structural and reward-related MRs

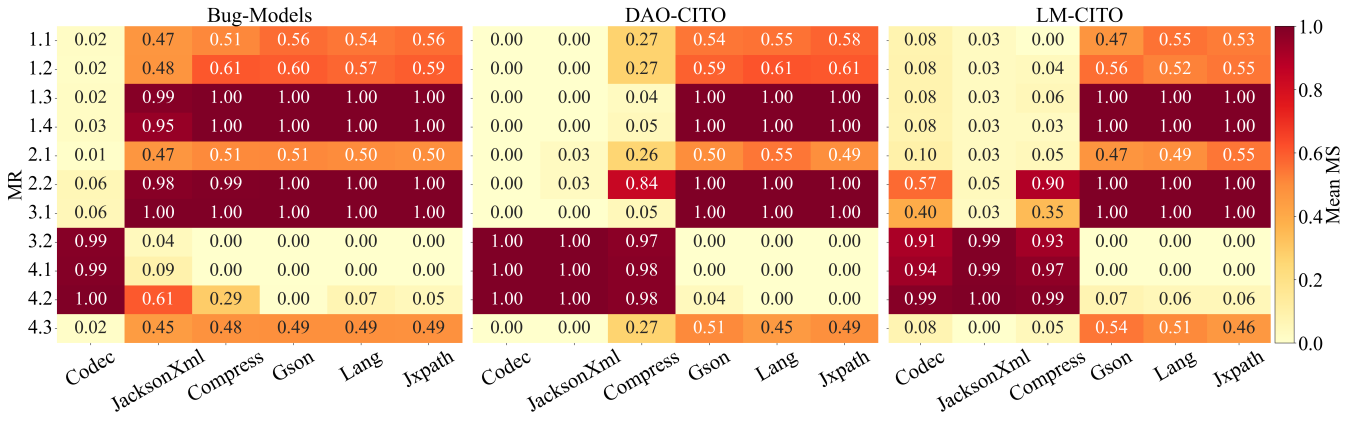

Figure 4: MS heatmaps of different models. Higher values and darker shading imply greater mutation detection by each MR across all models. Conversely, for DAO-CITO and LM-CITO, lower values and lighter shades suggest enhanced robustness under the corresponding metamorphic perturbations.

such as MR1.3, MR1.4, MR2.2, and MR3.1, whereas DAO-CITO and LM-CITO primarily react to MR3.2, MR4.1, and MR4.2. This divergence reveals a critical limitation of learning-based models in capturing logic-preserving variations, especially under metamorphic conditions of types T1-T3. For larger systems such as Gson, Lang, and Jxpath, MS differences among models converge. However, LM-CITO consistently outperforms DAO-CITO in MR1.1-MR3.1 and MR4.2-MR4.3, showing a greater detection of mutations and improved semantic stability, particularly in the cohesion, reward perturbation and fault aggregation tests.

We conducted pairwise statistical comparisons among bug-injected models, DAO-CITO, and LM-CITO. Table 5 reports the effect sizes and the corresponding P-values using the Mann-Whitney U test in the six programs.

Programs	Bug vs DAO		Bug vs LM		LM vs DAO	
	P	Eff.	P	Eff.	P	Eff.
Codec	0.12	0.28	0.00	-0.59	0.02	0.62
JacksonXml	0.01	0.47	0.01	0.48	0.05	0.27
Compress	0.11	0.28	0.08	0.31	0.32	-0.19
Gson	0.91	-0.02	0.96	0.01	0.89	-0.03
Lang	0.91	-0.02	1.00	0.00	0.89	-0.01
Jxpath	0.99	0.00	0.84	0.04	0.75	-0.04

Table 5: Effect sizes (Eff.) and p-values (P) for CITO comparisons. Eff. quantifies the magnitude of group differences.

Significant differences are observed in Codec and JacksonXml, where LM-CITO outperforms DAO-CITO with effect sizes of $\delta = 0.62$ and $\delta = 0.27$, respectively; both comparisons are statistically significant, with p-values below 0.05. Moreover, LM-CITO detects faults that DAO-CITO fails to capture. This is especially evident in Codec, where the comparison between the bug-injected model and LM-CITO yields a large negative effect size of $\delta = -0.59$ and a highly significant p-value below 0.001, indicating the heightened sensitivity of LM-CITO to injected anomalies.

By contrast, in larger and more stable programs such as Gson, Lang, and Jxpath, the performance differences between the two methods are negligible, suggesting that both converge under highly regularized environments. In general, these findings suggest that input-sensitive MRs are particularly effective for simpler programs, while structure-aware MRs are essential in more complex systems. The energy-based design of LM-CITO improves robustness and reduces false positives, making it a promising approach to adaptive and scalable defect detection.

Our results reveal that different metamorphic perturbations expose distinct defect types: input and reward scaling perturbations detect environment- and policy-related faults, while structural and path-based disturbances reveal architectural and propagation errors. Thus, a diverse MR suite is vital for thorough fault detection. DAO-CITO performs well with simpler input and stochastic faults, offering efficiency in small programs. LM-CITO shows stronger robustness against complex structural and semantic perturbations due to its energy-based design, making it suitable for highly structured programs. These findings provide practical guidance on the selection of testing methods that are aligned with the complexity of the system and the nature of the defect.

Conclusion

This work presents LM-CITO, a stability-aware RL framework for CITO generation. By incorporating a Lyapunov-guided reward into A2C training, LM-CITO promotes structural convergence and stable learning. We further design a domain-specific MT suite to validate behavioral consistency under perturbations, supporting oracle-free semantic verification. Experimental results on six real-world systems and 19 injected bug models show that LM-CITO reduces stub-related testing costs and improves convergence compared to existing methods. Although DAO-CITO remains effective in lightweight environments, LM-CITO offers superior robustness in complex systems. Together, Lyapunov-based optimization and MR-based validation provide a principled and interpretable solution for RL-based software testing.

Acknowledgments

This work is partially supported by the Fundamental Research Funds for the Xuzhou Science and Technology Project under grant No. KC22047; Xuzhou K&D Program under grant No. KC23296; the National Natural Science Foundation of China under grant No. 71774159, 61673384; the Postdoctoral Foundation of China under grant No. 2021T140707; and China Scholarship Council.

References

Briand, L. C.; Labiche, Y.; and Wang, Y. 2001. Revisiting strategies for ordering class integration testing in the presence of dependency cycles. In *Proc. of the International Symposium on Software Reliability Engineering*, 287–296.

Chen, T. Y.; Cheung, S. C.; and Yiu, S. M. 1998. Metamorphic testing: a new approach for generating next test cases. *Technical Report HKUST-CS98-01*.

Czibula, G.; Czibula, I. G.; and Marian, Z. 2018. An effective approach for determining the class integration test order using reinforcement learning. *Applied Soft Computing*, 65: 517–530.

Ding, Y.; Zhang, Y.; Yuan, G.; Jiang, S.; Dai, W.; and Baresi, L. 2025. Optimizing class integration testing with criticality-driven test order generation. In *Proc. of the IEEE International Conference on Software Analysis, Evolution and Reengineering*, 276–286.

Ding, Y.; Zhang, Y.; Yuan, G.; Jiang, S.; Dai, W.; and Zhang, Y. 2023a. Integration test order generation based on reinforcement learning considering class importance. *Journal of Systems and Software*, 205: 111823.

Ding, Y.; Zhang, Y.; Yuan, G.; Li, Y.; Jiang, S.; and Dai, W. 2023b. A reinforcement learning method for generating class integration test orders considering dynamic couplings. In *Proc. of the International Conference on Neural Information Processing*, 95–107.

Gajic, Z.; and Qureshi, M. T. J. 2008. *Lyapunov matrix equation in system stability and control*. Courier Corporation.

Jiang, S.; Zhang, M.; Zhang, Y.; Wang, R.; Yu, Q.; and Keung, J. W. 2021. An integration test order strategy to consider control coupling. *IEEE Transactions on Software Engineering*, 47(7): 1350–1367.

Just, R.; Jalali, D.; and Ernst, M. D. 2014. Defects4J: A database of existing faults to enable controlled testing studies for Java programs. In *Proc. of the International Symposium on Software Testing and Analysis*, 437–440.

Klein, T.; Miklautz, L.; Sidak, K.; Plant, C.; and Tschiatschek, S. 2024. Plasticity loss in deep reinforcement learning: A survey. *arXiv preprint arXiv:2411.04832*.

Kung, D.; Gao, J.; Hsia, P.; Toyoshima, Y.; and Chen, C. 1995a. A test strategy for object-oriented programs. In *Proc. of the International Computer Software and Applications Conference*, 239–244.

Kung, D. C.; Gao, J.; Hsia, P.; Lin, J.; and Toyoshima, Y. 1995b. Class firewall, test order, and regression testing of object-oriented programs. *Journal of Object-Oriented Programming*, 8(2): 51–65.

Le Traon, Y.; Jéron, T.; Jézéquel, J.-M.; and Morel, P. 2000. Efficient object-oriented integration and regression testing. *IEEE Transactions on Reliability*, 49(1): 12–25.

Meng, F.; Wang, Y.; Yu, H.; and Zhu, Z. 2022. Devising optimal integration test orders using cost–benefit analysis. *Frontiers of Information Technology & Electronic Engineering*, 23(5): 692–714.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *Proc. of the International Conference on Machine Learning*, 1928–1937.

Puterman, M. L. 1990. Markov decision processes. *Handbooks in Operations Research and Management Science*, 2: 331–434.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1): 12348–12355.

Briand, L. C.; Labiche, Y.; and Wang, Y. 2003. An investigation of graph-based class integration test order strategies. *IEEE Transactions on Software Engineering*, 29(7): 594–607.

Wang, Z.; You, H.; Chen, J.; Zhang, Y.; Dong, X.; and Zhang, W. 2021. Prioritizing test inputs for deep neural networks via mutation analysis. In *Proc. of the International Conference on Software Engineering*, 397–409.

Xie, X.; Zhang, Z.; Chen, T. Y.; Liu, Y.; Poon, P.-L.; and Xu, B. 2020. METTLE: A metamorphic testing approach to assessing and validating unsupervised machine learning systems. *IEEE Transactions on Reliability*, 69(4): 1293–1322.

Zhang, G.; Yuan, G.; Cheng, D.; Liu, L.; Li, J.; and Zhang, S. 2025a. Mitigating propensity bias of large language models for recommender systems. *ACM Transactions on Information Systems*, 43(6): 1–26.

Zhang, G.; Zhang, S.; and Yuan, G. 2024. Bayesian graph local extrema convolution with long-tail strategy for misinformation detection. *ACM Transactions on Knowledge Discovery from Data*, 18(4): 1–21.

Zhang, M.; Keung, J. W.; Chen, T. Y.; and Xiao, Y. 2021a. Validating class integration test order generation systems with metamorphic testing. *Information and Software Technology*, 132: 106507.

Zhang, M.; Keung, J. W.; Xiao, Y.; and Kabir, M. A. 2021b. Evaluating the effects of similar-class combination on class integration test order generation. *Information and Software Technology*, 129: 106438.

Zhang, S.; Song, H.; Wang, Q.; Shen, H.; and Pei, Y. 2025b. A test oracle for reinforcement learning software based on lyapunov stability control theory. In *Proc. of the International Conference on Software Engineering*, 611–611.

Zhang, Y.; Zhang, Y.; Zhang, Z.; Jiang, S.; Ding, Y.; and Yuan, G. 2023. Generation method of class integration test order based on deep reinforcement learning. *Acta Electronica Sinica*, 51(2): 455.