

The Pursuit of Diversity: Multi-Objective Testing of Deep Reinforcement Learning Agents

Antony Bartlett^[0009–0008–2654–8556], Cynthia Liem^[0000–0002–5385–7695],
Annibale Panichella^[0000–0002–7395–3588]

Delft University of Technology, Delft, The Netherlands
`a.j.bartlett@tudelft.nl`, `c.c.s.liem@tudelft.nl`, `a.panichella@tudelft.nl`

Abstract. Testing deep reinforcement learning (DRL) agents in safety-critical domains requires discovering diverse failure scenarios. Existing tools such as **INDAGO** rely on single-objective optimization focused solely on maximizing failure counts, but this does not ensure discovered scenarios are diverse or reveal distinct error types. We introduce **INDAGO-Nexus**, a multi-objective search approach that jointly optimizes for failure likelihood and test scenario diversity using multi-objective evolutionary algorithms with multiple diversity metrics and Pareto front selection strategies. We evaluated **INDAGO-Nexus** on three DRL agents: humanoid walker, self-driving car, and parking agent. On average, **INDAGO-Nexus** discovers up to 83% and 40% more unique failures (test effectiveness) than **INDAGO** in the SDC and Parking scenarios, respectively, while reducing time-to-failure by up to 67% across all agents.

Keywords: Multi-objective search, deep reinforcement learning, surrogate models

1 Introduction

Reinforcement Learning (RL) [27], and more recently Deep Reinforcement Learning (DRL) [21], have emerged as powerful tools for automating complex tasks in dynamic and uncertain environments [22, 26]. DRL agents learn through trial-and-error interactions with their environment and are increasingly deployed in Cyber-Physical Systems (CPS), such as self-driving cars (SDCs) and humanoid robots [4], where safety, reliability, and adaptability are critical.

Ensuring the reliability of DRL-based systems remains a significant challenge [2, 29]. Unlike conventional software, DRL agents are opaque, stochastic, and highly sensitive to reward signals. Their behavior depends not only on a learned policy but also on vast environment and action spaces. As a result, edge-case scenarios—those most likely to trigger failures—are difficult to identify, and traditional testing approaches often fall short [29]. Real-world incidents involving autonomous agents [9, 12, 18] underscore the consequences of failing to detect such rare conditions during testing.

Simulation-based testing is a widely adopted technique for evaluating DRL agents, especially in safety-critical domains such as autonomous agents. It allows developers to generate and execute test scenarios in a controlled and reproducible setting, substantially reducing the time, cost, and safety risks of

real-world testing [10, 16]. A test scenario refers to a specific configuration of the environment—including initial agent state, obstacle placement, and task-specific parameters—that defines the context in which the agent is evaluated. Despite its advantages, simulation-based testing still remains resource-intensive, and exhaustive exploration of the test input space is often infeasible due to its combinatorial complexity.

To address these challenges, Biagiola and Tonella [2] proposed the use of surrogate models to guide test generation for DRL agents. Their framework, **INDAGO**, builds a surrogate model from training logs to predict whether a given test scenario is likely to cause agent failure—without executing it in simulation. **INDAGO** combines this model with two search strategies: Hill Climbing (HC) and a single-objective Genetic Algorithm (GA). Their empirical evaluation across multiple DRL tasks showed that (1) surrogate models substantially reduced simulation time, and (2) both HC and GA outperformed earlier random-search-based methods [29] in identifying more failure-inducing scenarios.

While **INDAGO** effectively increases the number of failing scenarios, it focuses solely on failure likelihood, overlooking diversity. This is a key limitation—generating more failures does not guarantee that they are distinct. Diversity is crucial in testing as it promotes broader coverage of the agent’s operational space and improves the chances of uncovering unique faults, helping to address the *curse of rarity* problem [20].

In this paper, we propose **INDAGO-Nexus**, a multi-objective search framework for DRL testing that jointly optimizes for failure probability and scenario diversity. We investigate:

RQ1: *How does multi-objective search perform in finding diverse failures?*

RQ2: *How effective is **INDAGO-Nexus** compared to state-of-the-art **INDAGO**?*

To tackle this inherently multi-objective problem, we employed multi-objective evolutionary algorithms (MOEAs). MOEAs are well-suited for this task as they efficiently explore trade-offs and produce a set of Pareto-optimal solutions [6, 14, 15]. We experimented with two diversity metrics (Euclidean distance and PCA-based clustering), two MOEAs (NSGA-II [7] and AGE-MOEA [23]), and two Pareto-front selection strategies (highest failure likelihood and knee point [31]). We benchmarked **INDAGO-Nexus** on three DRL agents: self-driving car, humanoid walker, and parking agent.

Our results indicate **INDAGO-Nexus** discovers up to 83% and 40% more unique failures than **INDAGO** in the SDC and Parking scenarios, respectively, while reducing time-to-failure by up to 67%. Knee-point selection consistently produces the best results across most configurations, while different diversity metrics show varying effectiveness depending on the scenario.

Therefore, our contributions are:

- We propose **INDAGO-Nexus**, a multi-objective approach to generate diverse test environments for DRL agents.
- We demonstrate **INDAGO-Nexus** discovers up to 83% more unique failures than **INDAGO** while reducing time-to-failure by up to 67%.

- We provide a complete replication package with implementation and experimental data¹.

2 Background and Related Work

This section describes the key background concepts used in this work and summarizes the related work.

2.1 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) extends Reinforcement Learning (RL) by using deep neural networks as function approximators to handle high-dimensional inputs (e.g., images) and continuous action spaces (e.g., steering angles in autonomous driving) [22]. Instead of explicitly representing policies or value functions, DRL learns them directly from raw observations and rewards using deep learning architectures.

This capability makes DRL particularly suitable for complex tasks in cyber-physical systems such as self-driving cars (SDCs), where the environment is dynamic and the decision space is large. In this work, we focus on *model-free* DRL algorithms, where agents learn directly from interactions without an explicit model of the environment. For example, the DRL agent for the parking scenario we consider in our case study (see Section 4) was trained with *Proximal Policy Optimization (PPO)* [24], a widely adopted policy-gradient method known for its stability and sample efficiency.

The agent’s goal is to maximize the cumulative reward it receives in the long run [27].

2.2 Black-box Testing

In black-box testing, diversity is a common strategy for exposing faults by varying inputs and observing outputs [13]. For DRL agents, we focus on input diversity—variations in environment configurations—to guide test generation without costly simulations, combined with a surrogate to estimate failure likelihood.

Related work on diversity-based testing for deep neural networks includes approaches that optimize for input diversity to improve fault detection. Aghababaeian et al. [1] proposed black-box testing techniques that use test case diversity to generate diverse test inputs for deep neural networks. While their work demonstrates the effectiveness of diversity-based approaches, their diversity metrics are primarily designed for image inputs rather than the tabular data that characterizes our MLP-based surrogate models. Our work adapts the concept of diversity-driven testing to the specific context of DRL environment configurations, which are typically represented as structured tabular data with both continuous and categorical features.

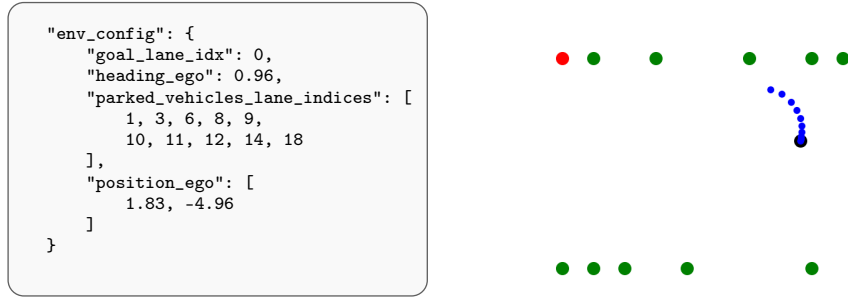


Fig. 1: Parking scenario showing (a) environment configuration details used by DRL as a starting point, and (b) the DRL agent behavior. In (b), the red dot indicates the target parking spot, green dots represent parked cars, the black dot marks the agent’s starting position, and the blue dots trace its trajectory.

2.3 Surrogates

Surrogate models are lightweight machine learning approximations that learn to predict the behavior or outcomes of computationally expensive processes based on historical data. In the context of DRL testing, these models have proven particularly effective at reducing evaluation costs [2]. Specifically, given an environment configuration (Figure 1a), a surrogate model can predict the failure probability in the range $[0, 1]$, thereby enabling efficient test generation without requiring costly simulation executions.

2.4 Testing DRL Agents

The most directly related work is by Biagiola and Tonella [2], who introduced **INDAGO**, a test generation framework that leverages surrogate models trained on DRL logs to predict whether a given test environment is likely to cause failure. These surrogates return a continuous score in $[0, 1]$, representing the estimated failure probability of an environment. **INDAGO** combines this prediction with search-based strategies, namely Hill Climbing [25] and Genetic Algorithms [11], to efficiently explore the input space.

Our work builds on **INDAGO** by introducing a second optimization objective: input diversity. This extension aims to increase the diversity of the discovered failures, thereby improving coverage of the agent’s operational space.

3 Approach

The goal of our approach is to *generate test scenarios that are both failure-inducing and diverse*, increasing the likelihood of uncovering unique faults in DRL agents. We build upon the **INDAGO** framework [2], which uses a surrogate model trained on DRL training logs to predict whether a given environment is likely to cause a failure. While **INDAGO** focuses solely on failure likelihood, we

¹ [10.6084/m9.figshare.29204687](https://figshare.com/figures-and-data/29204687)

extend it to a *multi-objective test generation framework* that also accounts for scenario diversity—crucial for uncovering unique faults. Hence, we formulate the problem of generative diverse failure-inducing test scenarios as follows:

Problem Definition 1 *Given a DRL agent A , a set of test scenarios E , and a surrogate model $s : E \rightarrow [0, 1]$ that estimates the likelihood of failure for a given e , the goal is to generate a set of test scenarios $E_{sel} \subseteq E$ that maximize the likelihood of A to fail while maximizing diversity among the selected scenarios. More formally, the goal is to find a set of environments E_{sel} that optimizes the following objectives:*

$$\begin{cases} \max O_1 = s(e) & (\text{Failure Likelihood}) \\ \max O_2 = \text{Div}(e, E_{sel}) & (\text{Input Diversity}) \end{cases} \quad (1)$$

where O_1 (Failure Likelihood) is the predicted likelihood of failure for a given environment e ; and O_2 (Input Diversity) is a measure of how diverse the selected environments are compared to the other test scenarios in E_{sel} .

In the following, we describe the surrogate model, diversity metrics, and the multi-objective optimization framework.

3.1 Surrogate models

Following the prior work by Biagiola and Tonella [2], we adopt a lightweight surrogate model to estimate the failure likelihood of a test scenario without executing it. Specifically, we utilize a multi-layer perceptron (MLP) classifier trained on data collected during the DRL agent’s training episodes. The MLP architecture is well-suited for this task due to the relatively small training dataset (up to 10k labeled environments) and the low-dimensional nature of environment feature vectors (e.g., 24-dimensional in the parking scenario). The surrogate outputs $s(e) \in [0, 1]$, representing failure probability for environment e .

3.2 Diversity Metrics

We explore two diversity methods: **Euclidean distance** computes the mean distance between a new scenario and previously selected ones, and **PCA-based clustering** [8] measures distance to the nearest cluster centroid in reduced dimensionality space. Both methods handle variable-length features through one-hot encoding and zero-padding.

3.3 Multi-Objective Evolutionary Testing

To generate a diverse set of failure-inducing test scenarios, INDAGO-Nexus employs MOEAs that evolve environment configurations to simultaneously maximize (1) the likelihood of failure, as predicted by a surrogate model, and (2) diversity with respect to a growing archive of previously selected tests. We use two MOEAs: NSGA-II [7] and AGE-MOEA [23] to optimize both failure likelihood and diversity.

Algorithm 1 Multi-Objective Test Generation for DRL Agents

Require: s , classifier (surrogate model);
Require: TR , test runs;
Require: G , generations;
Require: PS , population size;
Require: CR , crossover rate;
Require: E_{train} , set of environment configurations where the DRL agent failed during training;
Ensure: E_{sel} , the archive storing a diverse set of likely-failing test scenarios.

```

1: currentTestRun  $\leftarrow 0$ ;
2: while currentTestRun  $\leq TR$  do
3:   population  $\leftarrow$  GENERATE-POPULATION( $PS$ ,  $E_{train}$ );
4:   COMPUTE-OBJECTIVES(population,  $s$ ,  $E_{sel}$ );
5:   currentGeneration  $\leftarrow 0$ ;
6:   while currentGeneration  $\leq G$  do
7:     newPop  $\leftarrow \emptyset$ ;
8:     while |newPop|  $< PS$  do
9:        $p_1 \leftarrow$  SELECTION(population)
10:       $p_2 \leftarrow$  SELECTION(population)
11:      if getRandomFloat()  $< CR$  then
12:         $o_1, o_2 \leftarrow$  crossover( $p_1, p_2$ )
13:      end if
14:       $o_1 \leftarrow$  MUTATE( $o_1$ )
15:       $o_2 \leftarrow$  MUTATE( $o_2$ )
16:      newPop  $\leftarrow$  newPop  $\cup \{o_1, o_2\}$ 
17:    end while
18:    population  $\leftarrow$  ELITISM( $\leftarrow$  newPop  $\cup$  population);
19:    if stagnation detected or currentGeneration  $\geq G$  then
20:       $E_{sel} \leftarrow E_{sel} \cup$  GET-BEST-INDIVIDUAL(population)
21:    end if
22:  end while
23: end while
24: return  $E_{sel}$ 
  
```

Algorithm 1 outlines this process. The search begins with a population of size PS . Each individual is evaluated without test execution using heuristics described in Sections 3.1 and 2.2. The algorithm proceeds for G generations. In each generation, two parents are selected (Lines 9-10), the crossover is applied with probability CR (Line 11), and both offspring are mutated (Lines 14-15). Offspring are added to the new population (Line 16), and elitism selects the best individuals for the next generation (Line 18).

To avoid stagnation, **INDAGO-Nexus** monitors hypervolume changes across generations. If improvement falls below a threshold, or we have reached the maximum number of generations G , the best individual is archived (Line 20). To explore new regions of space, the process continues until the total test runs TR is met. The resulting archive E_{sel} contains diverse, likely-failing scenarios, which are then executed on the DRL agent for validation.

While **INDAGO-Nexus** builds on established MOEAs like NSGA-II [7] and AGE-MOEA [23], it introduces several domain-specific adaptations: failure-based population seeding, surrogate-based fitness estimation, diversity measured against an evolving archive, and a stagnation-aware reset strategy for enhanced exploration. We describe these customizations in the following paragraphs.

Population Initialization. We seed the initial population with environments where the DRL agent failed during training (E_{train}), biasing the search towards failure-prone regions [2].

Selection. Selection determines which individuals are chosen for reproduction. INDAGO-Nexus uses binary tournament selection based on Pareto dominance [7], and the non-dominated sorting in particular.

Objective Calculation. Each individual is evaluated without test execution. The surrogate model s estimates failure likelihood, while diversity is computed with respect to the archive E_{sel} using either Euclidean distance or PCA-based clustering metrics as described in Section 3.2.

Crossover and Mutation. INDAGO-Nexus applies single-point crossover (Line 12) probabilistically based on the crossover rate CR , randomly selecting a crossover point in the feature vector and swapping segments between parents.

Mutation (Lines 14-15) is *always* applied to both offspring and uses *saliency-based feature selection* [26], which computes the gradient of the surrogate model’s output with respect to each input feature. The saliency determines which features impact the surrogate’s prediction, guiding mutation decisions. For *fixed-length features* (position, heading), we apply polynomial mutation [6] with type-specific handling (real-valued, integer, binary, categorical). For *variable-length features* (parked vehicle lists), we randomly apply removal, addition, or modification operations. A repair operator ensures validity after mutations.

Elitism. We retain the best individuals using NSGA-II’s fast non-dominated sorting with crowding distance [7] or AGE-MOEA’s adaptive survival score [23].

Stagnation and Archive Update. INDAGO-Nexus maintains an archive E_{sel} of selected scenarios. Stagnation is detected by monitoring the change in hypervolume of the Pareto front P_g at generation g relative to a reference point $\mathbf{r}$ (Equation 2). The hypervolume metric reflects both diversity and convergence by measuring the volume dominated by P_g with respect to $\mathbf{r}$:

$$|HV(P_g, \mathbf{r}) - HV(P_{g-n_{last}}, \mathbf{r})| < \text{tol} \quad (2)$$

When stagnation is detected, INDAGO-Nexus selects a representative individual from the Pareto front using two strategies: *Maximum Failure Likelihood* (highest predicted failure probability) or *Knee Point* (best trade-off between failure likelihood and diversity) [31].

4 Empirical Evaluation

Our empirical evaluation aims to answer the following research questions:

RQ1: *How does multi-objective search perform in finding diverse failures?*

RQ2: *How effective is INDAGO-Nexus compared to state-of-the-art INDAGO?*

4.1 Case Studies

We evaluate INDAGO-Nexus on three diverse DRL agents from Biagiola and Tonella [2], each representing different application domains and complexity:

Parking. The first environment we consider is the **parking** environment [19]. In this scenario, the DRL agent begins at a specific location with a defined heading direction. The agent must park in a designated goal lane while avoiding collisions with any of the already parked vehicles. An example of this environment is

illustrated in Figure 1a. The environment consists of 24 elements, including goal lane, heading, 20 parking slots, and the x, y coordinates of the starting position.

Humanoid. Next, we examine the **Humanoid** environment, one of the more challenging environments from the MuJoCo simulator [28]. In this scenario, the agent must control a bipedal robot to walk on a smooth surface within a 3D space. The environment configuration of the Humanoid robot consists of two arrays: joint position and joint velocity. The joint position array contains 24 elements representing the positions and rotations of the robot’s joints, while the joint velocity array contains the linear and angular velocities of the joints.

Self-Driving Car. The final environment we consider is the Self-Driving Car (SDC), developed using the DonkeyCar simulator [17]. In this environment, the DRL agent must navigate a car from the starting point of a track to the end without leaving track boundaries. The environment configuration consists of a list with 12 pairs, where each pair comprises a command and a value. For example, a command could be a left or right turn, with the value specifying the turn length.

4.2 Baseline

To ensure fair comparison of all approaches, we utilize the original models from the INDAGO paper. For our baseline, we employed the GA from the original INDAGO tool. This approach uses the predictive value of the surrogate model as a fitness value to gradually mutate environments in the search for failures. We executed this with the `saliency_failure` test policy, which ensures the initial population is created from previously failing environments sourced from the training data, incorporating saliency (as discussed in Section 3) during crossover and mutation. We selected this method as it demonstrated the highest rate of failure in the original study [2] against other approaches (`random search` and `hill climbing`). Due to the stochastic nature of evolutionary search, we executed the baseline 50 times, applying the process across all three agent scenarios.

4.3 Implementation and Parameter Settings

We implemented AGE-MOEA and NSGA-II using Pymoo v0.6.1 [3]. Experiments used 50 generations, population size 50, crossover rate 0.75, with stagnation-based termination ($tol = 5 \times 10^{-6}$, $n_{last} = 10$). The reference point for hypervolume calculation is set to $r = [1.2, 20.2]$, representing the upper bounds for the objectives. Experiments were conducted within Docker containers on a machine equipped with an AMD EPYC 7713 64-Core CPU (2.6 GHz), 256 threads, and an NVIDIA A40 GPU (48GB GDDR6) for tensor-based computation.

4.4 Evaluation Criteria

We measure: (1) *unique failures* via PCA+K-means clustering of execution traces, (2) *input/output diversity* using entropy across clusters, and (3) *time-to-failure (TTF)* efficiency. Statistical significance is assessed using Wilcoxon rank-sum test ($\alpha = 0.05$) and Vargha-Delaney effect size.

4.5 Failure Detection and Diversity Assessment

To assess the effectiveness of our multi-objective search strategies (**RQ1**), we executed both **NSGA-II** and **AGE-MOEA** across all combinations of diversity metrics and Pareto front selection strategies.

We focused on three aspects: the number of unique failures discovered, the diversity of these failures, and the efficiency in detecting them. We define a failure as a test scenario in which the DRL agent violates its task specification (e.g., crashing, falling, or missing the goal). To measure *unique failures*, we cluster the execution traces (i.e., output trajectories) of failing tests using Principal Component Analysis (PCA) followed by K-means. Each resulting cluster corresponds to a distinct failure type, providing a behavior-grounded notion of uniqueness. Since trajectory lengths vary, we normalize them via zero-padding. For the Parking and SDC scenarios, we track the agent’s position over time, while in the Humanoid scenario, we monitor the robot’s vertical movement to detect falls.

To determine the optimal number of clusters K^* , we apply silhouette analysis and increase K only if the silhouette score improves by at least 20%, reducing the effect of noisy improvements. This clustering method is used both to compute output diversity and to count unique failures. It is important to distinguish this post hoc clustering from the PCA-based metric used within the search itself (Section 3.2). The former relies on test execution results while the latter (our objective) relies on the input features (i.e., without running the tests).

We also evaluate the diversity of failures from both input and output perspectives. *Output diversity* is measured using the same clustering of output trajectories used to identify unique failures. For *input diversity*, we cluster the failing environment configurations using PCA and K-means, providing insight into the structural variety of the input scenarios, regardless of the agent’s behavior. In both cases, we calculate two diversity metrics: *unique failures*, which reflects how many clusters are populated, and *entropy*, which quantifies how evenly failures are distributed across clusters. We follow the definitions provided in the **INDAGO** framework [2].

To assess *test efficiency*, we compute the average time required to find the first failing test case. This metric reflects how quickly each approach exposes failures. The same metrics are then used in the comparison between **INDAGO-Nexus** and the baseline **INDAGO** (**RQ2**). To assess statistical significance, we apply the Wilcoxon rank-sum test [5] with a confidence level of $\alpha = 0.05$, and we report the Vargha–Delaney $\hat{A}_{12}$ statistics for the effect size [30].

5 Results

This section presents the results of our study, addressing in turn each research question introduced earlier.

5.1 Configuration Comparison (RQ1)

Table 1 outlines the performance of each **INDAGO-Nexus** configuration across the three case studies:

Table 1: Performance metrics across different configurations for all three agents, displaying median and Interquartile Range (IQR) across 50 runs. (Gray cells indicate top values for each agent and approach.)

(a) Parking Agent					
Configuration	Failures		Entropy		TTF
	#Total	#Unique	Input	Output	
AGE-MOEA					
PCA/Knee	10 (3.75)	7 (1.00)	69.05 (19.19)	62.34 (15.62)	106.12 (26.89)
PCA/ O_1	7 (3.00)	4 (2.00)	24.79 (41.62)	39.28 (18.25)	108.13 (26.74)
Euclidean/Knee	10 (3.75)	7 (2.75)	63.95 (18.20)	61.98 (16.33)	123.27 (27.51)
Euclidean/ O_1	7 (2.00)	4 (1.00)	11.72 (37.64)	39.85 (14.35)	123.73 (26.27)
NSGA-II					
PCA/Knee	10 (3.00)	7 (2.00)	73.00 (20.13)	63.64 (9.77)	109.19 (23.00)
PCA/ O_1	7 (2.75)	4 (1.75)	10.26 (42.01)	41.05 (14.81)	108.12 (25.46)
Euclidean/Knee	11 (3.00)	6 (1.00)	64.12 (14.18)	60.59 (10.26)	120.34 (25.98)
Euclidean/ O_1	7 (2.00)	3 (1.00)	0.00 (39.71)	36.12 (15.04)	120.70 (27.56)
(b) Humanoid Agent					
Configuration	Failures		Entropy		TTF
	#Total	#Unique	Input	Output	
AGE-MOEA					
PCA/Knee	-	-	-	-	-
PCA/ O_1	2 (1.75)	1 (0.75)	10.65 (34.60)	0.00 (0.00)	78.94 (27.12)
Euclidean/Knee	1 (2.00)	1 (1.00)	0.00 (38.69)	0.00 (0.00)	96.06 (29.95)
Euclidean/ O_1	2 (2.00)	1 (1.00)	0.00 (35.59)	0.00 (0.00)	102.53 (27.58)
NSGA-II					
PCA/Knee	-	-	-	-	-
PCA/ O_1	2 (1.75)	1 (1.00)	0.00 (32.88)	0.00 (0.00)	79.32 (28.42)
Euclidean/Knee	2 (2.00)	1 (1.00)	0.00 (40.90)	0.00 (0.00)	90.94 (27.65)
Euclidean/ O_1	2 (2.00)	1 (0.75)	0.00 (28.95)	0.00 (0.00)	95.70 (23.58)
(c) Self-Driving Car (SDC)Agent					
Configuration	Failures		Entropy		TTF
	#Total	#Unique	Input	Output	
AGE-MOEA					
PCA/Knee	31 (5.75)	5 (6.00)	56.97 (12.51)	63.03 (75.90)	1578.16 (130.69)
PCA/ O_1	36 (3.75)	6 (6.00)	71.96 (12.36)	77.34 (86.13)	1092.10 (49.04)
Euclidean/Knee	31 (5.00)	5.5 (6.00)	57.06 (18.83)	71.90 (79.67)	1072.26 (43.36)
Euclidean/ O_1	50 (1.00)	6 (6.00)	74.70 (5.84)	64.99 (75.69)	2435.78 (274.89)
NSGA-II					
PCA/Knee	30 (4.00)	5 (6.00)	54.50 (18.05)	62.75 (74.53)	1521.15 (102.29)
PCA/ O_1	34 (5.75)	6 (6.75)	70.97 (13.84)	77.00 (85.31)	1087.94 (59.30)
Euclidean/Knee	30 (4.75)	5 (6.00)	56.12 (9.07)	71.07 (80.27)	1073.21 (45.60)
Euclidean/ O_1	50 (1.00)	6 (6.00)	72.19 (6.10)	66.78 (74.89)	2588.43 (270.45)

Choice of MOEA: AGE-MOEA generally outperforms NSGA-II in unique failures discovered. In Parking, AGE-MOEA achieves 4-7 unique failures compared to NSGA-II's 3-7. Both perform similarly in SDC, while Humanoid proves

challenging for both algorithms. AGE-MOEA’s adaptive survival score mechanism appears better suited for the multi-objective DRL testing problem compared to NSGA-II’s crowding distance approach.

Diversity Metrics: PCA-based diversity consistently outperforms Euclidean distance across all scenarios. In Parking, PCA configurations achieve substantially higher input entropy (24.79-69.05) compared to Euclidean (11.72-63.95), demonstrating PCA’s superior ability to capture meaningful relationships in high-dimensional feature spaces. The Euclidean approach occasionally achieves slightly higher total failure counts, revealing an important trade-off between failure quantity and diversity quality. This suggests that while Euclidean distance may find failures faster in certain regions, PCA-based diversity better explores the full failure space.

Pareto Selection Strategy: Knee-point selection consistently yields superior diversity compared to the extreme O_1 point across all scenarios. The knee point represents a balanced trade-off between objectives, achieving higher unique failures and entropy scores.

Different configurations perform best in different scenarios. Overall, AGE-MOEA tends to produce more unique and diverse failures than NSGA-II. Pareto knee-point selection consistently yields better results than the extreme O_1 point. However, the choice of configuration can significantly impact performance, highlighting the need for adaptive strategies that consider the specific context and objectives.

5.2 Comparison with INDAGO (RQ2)

Table 2 presents a direct comparison between INDAGO-Nexus (AGE-MOEA *Euclidean/Knee* configuration) and the state-of-the-art INDAGO approach. The results demonstrate significant improvements across multiple metrics:

Unique Failure Discovery: INDAGO-Nexus consistently finds more unique failures than INDAGO. In the Parking scenario, INDAGO-Nexus discovers 7 unique failures compared to INDAGO’s 5 (+40% improvement). For the SDC scenario, INDAGO-Nexus finds 5.5 unique failures versus INDAGO’s 3 (+83% improvement). The Humanoid scenario shows comparable performance between both approaches, which aligns with our observation that this scenario is inherently challenging for all testing methods.

Diversity Enhancement: The multi-objective approach significantly improves both input and output diversity. In the Parking case, INDAGO-Nexus achieves 63.95 input entropy compared to INDAGO’s 0.00, and 61.98 output entropy versus 49.37. This demonstrates that explicitly optimizing for diversity as an objective leads to more varied test scenarios and failure modes. The diversity improvements are particularly important for comprehensive testing, as they increase the likelihood of discovering edge cases and systematic weaknesses.

Efficiency Gains: INDAGO-Nexus demonstrates superior efficiency in finding failures across all scenarios. Time-to-failure improvements range from 18% (Parking: 123.27 vs 150.53) to 67% (SDC: 1072.26 vs 3251.97), with Humanoid

Table 2: Comparison between INDAGO-Nexus and INDAGO across different agents, displaying median and Interquartile Range (IQR) across 50 runs. Statistical significance markers for the Wilcoxon rank-sum test: * p -value<0.05, ** p -value<0.01, *** p -value<0.001.

(a) Parking Agent					
Approach	Failures		Entropy		TTF
	#Total	#Unique	Input	Output	
INDAGO-Nexus	10 (3.75)	7 (2.75)**	63.95 (18.20)***	61.98 (16.33)***	123.27 (27.51)***
INDAGO	14.5 (4.00)	5 (2.00)	0.00 (35.72)	49.37 (12.56)	150.53 (0.03)
(b) Humanoid Agent					
Approach	Failures		Entropy		TTF
	#Total	#Unique	Input	Output	
INDAGO-Nexus	1 (2.00)	1 (1.00)	0.00 (38.69)	0.00 (0.00)	96.06 (29.95)***
INDAGO	2 (2.75)	1 (1.00)	28.90 (39.31)	0.00 (50.89)	150.50 (0.01)
(c) Self-Driving Car (SDC) Agent					
Approach	Failures		Entropy		TTF
	#Total	#Unique	Input	Output	
INDAGO-Nexus	31 (5.00)***	5.50 (6.00)**	57.06 (18.83)	71.90 (79.67)***	1072.26 (43.36)***
INDAGO	17.00 (5.75)	3.00 (3.00)	61.72 (21.59)	34.36 (48.71)	3251.97 (0.35)

showing 36% improvement (96.06 vs 150.50). These efficiency gains suggest that the multi-objective approach not only finds more diverse failures but does so more quickly than single-objective methods. The efficiency improvements are statistically significant (Wilcoxon test, $p < 0.001$) with large effect sizes across all scenarios (Vargha-Delaney $\hat{A}_{12} > 0.9$) for TTF.

Statistical Significance: All reported improvements show statistical significance at $\alpha = 0.05$ level using the Wilcoxon rank-sum test. Effect sizes calculated using Vargha-Delaney $\hat{A}_{12}$ statistics indicate substantial practical differences, particularly for efficiency metrics where INDAGO-Nexus consistently achieves large effect sizes ($\hat{A}_{12} > 0.7$), indicating that INDAGO-Nexus significantly outperforms INDAGO in key performance areas.

INDAGO-Nexus finds more unique failures and higher input/output diversity in Parking and SDC. Instead, it matches INDAGO in the challenging Humanoid scenario. Finally, INDAGO-Nexus finds unique failures faster across all DRL agents.

6 Threats To Validity

Internal Validity: Our experimental design addresses several potential threats through careful control of variables and statistical rigor. We ensure fair comparison by running all algorithms under identical conditions with the same hardware and time constraints. Each experiment is repeated 50 times to account for the

stochastic nature of evolutionary algorithms, and we apply appropriate statistical tests (Wilcoxon rank-sum) with effect size measurements (Vargha-Delaney $\hat{A}_{12}$) to assess practical significance. The clustering approach for measuring unique failures uses established silhouette analysis with conservative thresholds (20% improvement requirement) to avoid over-segmentation due to noise.

External Validity: Our evaluation covers three diverse DRL domains with different state/action spaces and failure modes. However, generalizability to other DRL applications, training algorithms beyond PPO, or different architectures requires further investigation.

Construct Validity: The choice of evaluation metrics reflects established practices in the DRL testing literature, particularly building on the INDAGO framework. Our clustering-based approach to measuring unique failures provides a behavior-grounded assessment of diversity that captures meaningful differences in agent behavior rather than superficial input variations. The entropy-based diversity measures provide quantitative assessments that complement the qualitative notion of test case variety.

Conclusion Validity: The statistical methods employed (non-parametric tests, effect size calculations) are appropriate for the experimental design and address the non-normal distribution of performance metrics commonly observed in evolutionary computation experiments. The large number of experimental repetitions (50 per configuration) provides sufficient statistical power to detect meaningful differences between approaches. Furthermore, a small number of SDC configurations were executed on a smaller hardware setup (AMD Ryzen Threadripper 3970X 32-Core Processor, 64GB memory, NVIDIA GeForce RTX 3080 10GB). Although less powerful than the main setup, the results were considerably faster. However, other validation metrics remain consistent.

7 Conclusion and Future Work

This paper introduced INDAGO-Nexus, a multi-objective search approach for generating diverse failures in DRL environments. Unlike prior methods that focus solely on inducing agent failures, INDAGO-Nexus simultaneously optimizes for failure likelihood and scenario diversity, leading to a more comprehensive exploration of failure modes. Our empirical evaluation across three DRL environments—Parking, Humanoid, and Self-Driving Car (SDC)—shows that the multi-objective INDAGO-Nexus consistently outperforms the single-objective baseline INDAGO in terms of unique failures discovered, diversity (measured via input and output entropy), and search efficiency.

We found that the effectiveness of INDAGO-Nexus varies across scenarios, with AGE-MOEA generally achieving more unique and diverse failures than NSGA-II. Both the choice of diversity metric (Euclidean vs. PCA) and the Pareto selection strategy (knee point vs. extreme O_1) significantly influence the outcomes. In particular, knee-point selection consistently led to better results across most configurations. Nonetheless, the Humanoid scenario remains particularly challenging, exposing limitations in current diversity metrics and search robustness.

Future Work: Several research directions emerge from this work. First, exploring many-objective optimization with additional objectives such as test case complexity, execution cost, or coverage metrics could further improve testing effectiveness. Second, investigating adaptive diversity metrics that automatically adjust based on the characteristics of discovered failures could enhance the approach’s generalizability. Third, extending the framework to other DRL training algorithms (beyond PPO) and neural network architectures would broaden its applicability. Finally, developing theoretical foundations for understanding when and why multi-objective approaches outperform single-objective methods in testing contexts would provide valuable insights for the testing community.

INDAGO-Nexus introduces multi-objective search for generating diverse DRL failures, consistently outperforming INDAGO in unique failures and efficiency. AGE-MOEA with knee-point selection proves most effective, though Humanoid scenarios remain challenging. Future work includes many-objective extensions, improved surrogate models, and broader domain evaluation.

References

1. Aghababaeian, Z., et al.: Black-box testing of deep neural networks through test case diversity. *IEEE Transactions on Software Engineering* **49**(5), 3182–3204 (May 2023). <https://doi.org/10.1109/tse.2023.3243522>, <http://dx.doi.org/10.1109/TSE.2023.3243522>
2. Biagiola, M., Tonella, P.: Testing of deep reinforcement learning agents with surrogate models. *ACM Trans. Softw. Eng. Methodol.* **33**(3) (Mar 2024). <https://doi.org/10.1145/3631970>, <https://doi.org/10.1145/3631970>
3. Blank, J., Deb, K.: pymoo: Multi-objective optimization in python. *IEEE Access* **8**, 89497–89509 (2020)
4. Bloom, J., et al.: Decentralized multi-agent reinforcement learning with global state prediction (2023), <https://arxiv.org/abs/2306.12926>
5. Conover, W.J.: *Practical nonparametric statistics*, vol. 350. John Wiley & Sons (1998)
6. Deb, K.: Multi-objective optimisation using evolutionary algorithms: an introduction. pp. 3–34. Springer (2011)
7. Deb, K., et al.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation* **6**(2), 182–197 (2002)
8. Ding, C., He, X.: K-means clustering via principal component analysis. p. 29. *ICML ’04*, Association for Computing Machinery, New York, NY, USA (2004). <https://doi.org/10.1145/1015330.1015408>, <https://doi.org/10.1145/1015330.1015408>
9. Ghoshal, A.: Why did this humanoid robot go nuts and nearly injure its handlers? *New Atlas* (05 2025), <https://newatlas.com/ai-humanoids/humanoid-robot-nearly-injures-handlers-unitree/>
10. Giubilato, R., et al.: Simulation framework for mobile robots in planetary-like environments (2020), <https://arxiv.org/abs/2006.00057>
11. Goldberg, D.E., Holland, J.H.: Genetic algorithms and machine learning. *Machine Learning* **3**(2), 95–99 (1988). <https://doi.org/10.1023/A:1022602019183>, <https://doi.org/10.1023/A:1022602019183>
12. Guardian, T.: Us investigates 2.4m tesla self-driving vehicles after reported collisions. *The Guardian* (10 2024), <https://www.theguardian.com/technology/2024/oct/18/tesla-self-driving-car-investigation>

13. Guidotti, R., et al.: A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* **51**(5), 1–42 (2018)
14. Haq, F.U., et al.: Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization. pp. 811–822 (2022)
15. Ishibuchi, H., et al.: Evolutionary many-objective optimization: A short review. pp. 2419–2426. *IEEE* (2008)
16. Kaur, P., et al.: A survey on simulators for testing self-driving cars (2021), <https://arxiv.org/abs/2101.05337>
17. Kramer, T., contributors: Self driving car sandbox. <https://github.com/tawnkramer/sdsandbox> (2021), gitHub repository
18. Lambert, F.: Tesla full self-driving veers off road, flips car in scary crash driver couldn't prevent. *Electrek* (05 2025), <https://electrek.co/2025/05/23/tesla-full-self-driving-veers-off-road-flips-car-scary-crash-driver-couldnt-prevent/>
19. Leurent, E.: An environment for autonomous driving decision-making. <https://github.com/eleurent/highway-env> (2018)
20. Liu, H.X., Feng, S.: "curse of rarity" for autonomous vehicles (2022), <https://arxiv.org/abs/2207.02749>
21. Mnih, V., et al.: Playing atari with deep reinforcement learning (2013), <https://arxiv.org/abs/1312.5602>
22. Mnih, V., et al.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015)
23. Panichella, A.: An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization. p. 595–603. *GECCO '19*, Association for Computing Machinery, New York, NY, USA (2019). <https://doi.org/10.1145/3321707.3321839>, <https://doi.org/10.1145/3321707.3321839>
24. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. *CoRR* **abs/1707.06347** (2017), <http://arxiv.org/abs/1707.06347>
25. Selman, B., Gomes, C.P.: Hill-climbing Search. John Wiley & Sons, Ltd (2006). <https://doi.org/https://doi.org/10.1002/0470018860.s00015>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/0470018860.s00015>
26. Simonyan, K., et al.: Deep inside convolutional networks: Visualising image classification models and saliency maps (2014), <https://arxiv.org/abs/1312.6034>
27. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, second edn. (2018), <http://incompleteideas.net/book/the-book-2nd.html>
28. Todorov, E., et al.: Mujoco: A physics engine for model-based control. 2012 *IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 5026–5033 (2012), <https://api.semanticscholar.org/CorpusID:5230692>
29. Uesato, J., et al.: Rigorous agent evaluation: An adversarial approach to uncover catastrophic failures (2018), <https://arxiv.org/abs/1812.01647>
30. Vargha, A., Delaney, H.D.: A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics* **25**(2), 101–132 (2000)
31. Zhang, X., et al.: A knee point-driven evolutionary algorithm for many-objective optimization. *IEEE transactions on evolutionary computation* **19**(6), 761–776 (2014)