Using Cooperative Co-Evolutionary Search to Generate Metamorphic Test Cases for Autonomous Driving Systems

Hossein Yousefizadeh, Shenghui Gu, Lionel C. Briand, *Fellow, IEEE*, and Ali Nasr

Abstract*—Autonomous Driving Systems (ADSs) rely on Deep Neural Networks, allowing vehicles to navigate complex, open environments. However, the unpredictability of these scenarios highlights the need for rigorous system-level testing to ensure safety, a task usually performed with a simulator in the loop. Though one important goal of such testing is to detect safety violations, there are many undesirable system behaviors, that may not immediately lead to violations, that testing should also be focusing on, thus detecting more subtle problems and enabling a finer-grained analysis. This paper introduces Cooperative Co-evolutionary MEtamorphic test Generator for Autonomous systems (CoCoMEGA), a novel automated testing framework aimed at advancing system-level safety assessments of ADSs. CoCoMEGA combines Metamorphic Testing (MT) with a search-based approach utilizing Cooperative Co-Evolutionary Algorithms (CCEA) to efficiently generate a diverse set of test cases. CoCoMEGA emphasizes the identification of test scenarios that present undesirable system behavior, that may eventually lead to safety violations, captured by Metamorphic Relations (MRs). When evaluated within the CARLA simulation environment on the Interfuser ADS, CoCoMEGA consistently outperforms baseline methods, demonstrating enhanced effectiveness and efficiency in generating severe, diverse MR violations and achieving broader exploration of the test space. Further expert assessments of these violations confirmed that most represent real safety risks, which validates their practical relevance. These results underscore CoCoMEGA as a promising, more scalable solution to the inherent challenges in ADS testing with a simulator in the loop. Future research directions may include extending the approach to additional simulation platforms, applying it to other complex systems, and exploring methods for further improving testing efficiency such as surrogate modeling.

Received 5 December 2024; revised 14 April 2025; accepted 13 May 2025. Date of publication 15 May 2025; date of current version 16 June 2025. This work was supported in part by a research grant from Huawei Technologies Canada Company Ltd., and in part by Canada Research Chair and Discovery Grant Programs of the Natural Sciences and Engineering Research Council of Canada (NSERC). Lionel C. Briand's contribution was partially funded by the Research Ireland grant 13/RC/209. Recommended for acceptance by J. Sun. *(Hossein Yousefizadeh and Shenghui Gu contributed equally to this work.) (Corresponding author: Hossein Yousefizadeh.)*

Hossein Yousefizadeh and Shenghui Gu are with the School of EECS, University of Ottawa, Ottawa, K1N 1A2, Canada (e-mail: hyous028@uottawa.ca; sgu2@uottawa.ca).

Lionel C. Briand is with the School of EECS, University of Ottawa, Ottawa, K1N 1A2, Canada, and also with the Research Ireland Lero centre, University of Limerick, V94 NYD3 Limerick, Ireland (e-mail: lbriand@uottawa.ca).

Ali Nasr is with the Waterloo Research Center of Huawei Technologies Canada Company Ltd., Waterloo, N2L 0A4, Canada (e-mail: ali.nasr2@huawei.com).

Digital Object Identifier 10.1109/TSE.2025.3570897

***Index Terms*—Autonomous driving systems, system-level testing, metamorphic testing, search-based testing, cooperative co-evolutionary algorithms, automated testing.**

I. Introduction

AUTONOMOUS Driving Systems (ADSs) have made remarkable strides and have garnered substantial research interest in recent years [1]. Typically, autonomous driving involves using different perception sensors such as cameras, LiDAR, and radar to enable a vehicle to operate without human intervention [2]. The vehicle equipped with these sensors and controlled by the ADS is referred to as the ego vehicle. It continuously gathers information about its surroundings, such as the position and movement of other vehicles, and feeds this data into the ADS to facilitate decision-making. An ADS deployed in an ego vehicle usually integrates several complex modules, including perception, prediction, planning, and control. Recent advances in Deep Neural Networks (DNNs) have further enhanced these modules, enabling ADSs to adapt their driving behavior to constantly changing and unpredictable environments [3] and numerous complex and hazardous situations in real-world scenarios [1].

In spite of their achievements, it remains crucial that ADSs undergo comprehensive system-level testing to ensure their safe functioning, particularly in uncommon situations where pedestrians or vehicles behave unpredictably [4], [5]. System-level testing refers to evaluating ADSs as a whole—making the system interact with its environment through realistic scenarios over a period of time—rather than focusing on individual modules such as perception or decision-making in isolation, based on single inputs. Although developers and manufacturers strive to mitigate potential hazards throughout the development cycle, accidents involving ADSs regularly occur [6], [7]. Though fatal crashes are rare, non-fatal accidents are more common [8], highlighting the ongoing challenges in ensuring the safety and reliability of ADSs [9].

In this paper, our focus is on validating the safety of the ADSs, aiming to assess whether the system behaves appropriately under realistic and challenging scenarios. In particular, we aim to identify subtle behavioral anomalies that may not result in immediate safety violations (e.g., collisions or traffic rule infractions) but reflect latent flaws in the system that could eventually compromise safety, especially in edge cases. Thus, this type of validation focuses on testing the external aspects of system behavior (e.g., safety), rather than evaluating the system's ability to function (e.g., failure rate measurement), which is a different objective and is typically addressed in *reliability testing* [10].

The system-level testing of ADSs is particularly challenging due to the huge and diverse set of hazards they face. Recent testing techniques primarily focus on approaches based on safety metrics [11], [12], which quantify how close an ADS is to dangerous situations (e.g., collisions), as well as the assessment of traffic rule violations. These metrics can be calculated by analyzing system behavior during simulations and defined as formal specifications like Signal Temporal Logic [13]. Temporal metrics such as Time-to-Collision [14] and Worst-Time-to-Collision [15], assess the ego vehicle's proximity to potential collisions over time. Other metrics, such as Time Headway and Time-to-React [16], estimate how much time the vehicle has to respond to potential hazards. While safety metrics provide a useful quantitative framework for detecting violations of known safety properties, they may not fully capture subtle or indirect unsafe system behavior that may eventually lead to violations [12], which we refer to as unsafe behavior hereafter. For example, an ADS may unnecessarily apply excessive braking in low-risk scenarios, such as slowing down abruptly when another vehicle is far ahead. Although this behavior does not immediately compromise safety, it can lead to negative consequences, including passenger discomfort, reduced fuel efficiency, and potentially even rear-end collisions. Such issues emphasize the need for testing methods that explore such scenarios that are not entailing immediate safety risks but are still critical for evaluating the ADS.

However, existing software testing approaches lack the complexity and adaptability required to address the intricacies of ADSs [17], [18]. A key challenge lies in specifying test oracles for ADSs, as they operate in the open contexts that constitute road traffic, where expected system behavior cannot always be precisely predicted or defined in advance. Indeed, ADSs face an infinite number of potential traffic scenarios, making it nearly impossible to determine expected outcomes for every situation [19], [20]. The challenge is further compounded by the widespread adoption of DNNs in ADSs. While DNNs enable sophisticated decision-making and adaptability, they lack transparency and formal specifications, making their behavior difficult to evaluate [21]. For example, determining whether an ADS has behaved correctly when encountering a pedestrian in foggy conditions is heavily influenced by environmental factors and sensor limitations. The lack of clear criteria for evaluating these decisions highlights the difficulty of defining reliable test oracles. Another challenge stems from the open nature of the environment in which ADSs operate, where new and unexpected scenarios can continuously arise. This openness means that test cases can never be exhaustive, as it is infeasible to cover all possible conditions in such a complex and dynamic domain. Therefore, it is reasonable to conclude that getting absolute proof of ADS safety and reliability is unattainable [22]. However, the effective and efficient identification of unsafe behavior emerges as an important and significant challenge.

To address these challenges, we propose *Cooperative Co-evolutionary MEtamorphic test Generator for Autonomous systems (CoCoMEGA)*, an effective and efficient automated test case generation method for system-level testing of ADSs. Specifically, we combine Metamorphic Testing (MT) and a search-based testing approach using Cooperative Co-Evolutionary Algorithm (CCEA) [23] to automatically generate a comprehensive and diverse set of system test cases that can effectively identify potential unsafe behavior in the target ADS. CCEA is used to efficiently explore the vast and complex input space of ADSs, increasing the likelihood of uncovering unsafe behavior. MT, in turn, provides a mechanism to automatically detect unsafe behavior, through the violation of Metamorphic Relations (MRs), such as verifying that the ego vehicle's steering angle remains consistent when weather conditions change.

This approach allows us to test ADS more comprehensively, going beyond the sole reliance on safety properties and aligning with established industrial standards such as ISO 21448's Safety of the Intended Functionality (SOTIF) [24], which emphasizes safety in the presence of functional or performance limitations. Indeed, MRs allow us to capture conditions where a system may behave undesirably despite being fault-free, a core concern of SOTIF. Another key advantage of MT is that it does not require a complete specification to derive useful MRs, which is often unrealistic, particularly for ADSs. Furthermore, given the complexity of ADS scenarios, with their large number of parameters and wide range of values, the search process and convergence to safety violations can be slow. MT expedites this convergence by selectively mutating only the parameters related to MRs. By leveraging MRs, we can detect subtle, unsafe behaviors that may not immediately violate safety properties but are eventually expected to lead to problems, thus providing a more thorough assessment of system. To ensure practical applicability, our method is compatible with simulation environments that support standard formats such as OpenScenario [25].

We have evaluated *CoCoMEGA* on the CARLA simulator using the INTERFUSER ADS. Our evaluation results demonstrate that *CoCoMEGA* has significantly greater effectiveness and efficiency than baseline methods in identifying unsafe behavior and exploring diverse regions of the search space. Its advantage is consistent across a wide range of search budgets and settings. This superior performance highlights *CoCoMEGA* as a promising and scalable solution for testing ADSs.

The major contributions of this paper can be summarized as follows:

- We proposed *CoCoMEGA*, the first automated testing method that combines MT and CCEA to enable the effective and efficient system-level testing of ADSs.
- We applied *CoCoMEGA* to a complex case study utilizing an industry-grade simulator and a high-performing ADS incorporating DNN modules.
- We evaluated the effectiveness and efficiency of *CoCoMEGA* through large scale experiments and comparisons with baseline methods, demonstrating its ability to identify a significant number of test cases that violate the MRs and thus exhibit unsafe behavior.

Note that while we have dedicated considerable effort to comprehensively review the literature, identify relevant MRs and validate them with experts, this is not intended to be a contribution of this paper. Instead, this compilation of MRs is simply used to support our experimental evaluation on a rich and diverse set of MRs that were defined independently from our study, within feasible computational bounds. Fully developing, validating, and integrating a more comprehensive set of MRs would require collaboration among many domain experts, which is beyond the scope of this study.

The remainder of this paper is structured as follows. Section II outlines the challenges and defines the problem in detail. Section III introduces related research works and existing gaps. Section IV describes *CoCoMEGA* in detail. Section V presents an empirical evaluation of the proposed approach. Section VI discusses the results and threats to validity. Section VII concludes the paper.

II. Problem and Challenges

In this section, we define the research problem and discuss its associated challenges.

A. Problem Definition

The primary problem to be addressed in this research is the development of an effective and efficient system-level testing methodology for ADSs to identify potential unsafe behavior, and thus determine when ADSs can be trusted under diverse and unpredictable scenarios and conditions. This approach focuses on the ADS interactions with the environment within realistic scenarios, rather than testing individual modules in isolation. To address this problem, we need to tackle the following specific issues.

- **Scalability and coverage.** Real-world system testing is unable to exhaustively cover the vast space of possible execution scenarios. However, in the context of critical systems, such as ADSs, identifying unsafe scenarios is crucial for system assurance, even when they are unlikely.
- **Oracle problem.** Defining precise criteria to determine acceptable behavior in ADSs, particularly for complex tasks like perception and decision-making, is challenging. The ever-changing and unpredictable nature of real-world traffic scenarios makes it impossible to enumerate all potential outcomes of ADSs and define corresponding expectations. For instance, consider a situation where a pedestrian is standing at a crosswalk or a traffic light turns yellow as the ego vehicle approaches at high speed. Deciding whether to stop abruptly, proceed, or adjust speed depends on various factors such as weather conditions and nearby vehicles. Such complexities make it challenging to develop effective test oracles, potentially making system testing ineffective.
- **Computational costs and efficiency.** The computational expense associated with high-fidelity simulations presents a significant challenge, effectively limiting system testing in practice and the ability to rapidly iterate on system improvements.

Given the above, there is a critical need for innovative testing methodologies that can effectively and efficiently test ADSs across a wide range of scenarios and conditions to identify unsafe behavior. The integration of MT with advanced search techniques offers a promising solution, allowing for the systematic and efficient exploration of scenario spaces and the detection of undesirable system behaviors that do not necessarily and systematically lead to safety property violations but are indicative of potential safety problems.

The primary issue tackled in this work is the development of a system-level testing framework, assuming an environment simulator in the loop, that combines MT with advanced search-based techniques to automatically generate test scenarios for ADSs. MT is especially suitable in this context because it facilitates the verification of system behavior without requiring an exhaustive system specification, focusing instead on identifying violations of MR properties that must hold across different inputs.

Specifically, we re-express the problem as a search problem. Let $\mathcal{S}$ denote the space of all possible driving scenarios, and let $\mathcal{Q}$ denote the space of all possible perturbations derived from a set of MRs, all sharing the same output relation or. Let $E_{or}(s,q)$ be a function that quantifies the extent of violation of or between the source scenario $s \in \mathcal{S}$ and the follow-up scenario $q(s)$ derived from applying $q \in \mathcal{Q}$ to s. Imagine we are testing an ADS to check its behavior when encountering a pedestrian in various weather conditions. Here, $\mathcal{S}$ represents all possible driving scenarios, such as different road types, traffic conditions, and initial pedestrian positions. The set $\mathcal{Q}$ represents possible perturbations, like changes in weather conditions (e.g., fog or rain) based on an MR with an output relation or specifying that the ADS should reduce speed when visibility decreases. Now, suppose we have a source scenario $s \in \mathcal{S}$ where the ADS encounters a pedestrian on a clear day. By applying a perturbation $q \in \mathcal{Q}$, we create a follow-up scenario $q(s)$ in which fog or rain reduces visibility. The function $E_{or}(s,q)$ then quantifies the extent of violation in the speed reduction of the ADS in response to this visibility change, measuring how well the ADS meets the expected output relation or.

Definition 1 (Problem): The problem is to find a diverse set SP of scenario-perturbation pairs $(s,q) \in \mathcal{S} \times \mathcal{Q}$ such that the resulting pair $(s, q(s))$ violates or:

$$SP = \{(s,q) \in \mathcal{S} \times \mathcal{Q} \mid E_{or}(s,q) > 0\}. \tag{1}$$

The challenge lies in efficiently searching the vast space $\mathcal{S} \times \mathcal{Q}$ to discover pairs (s,q) that maximize the detection of MR violations, thereby exposing unsafe behavior in the ADS. This requires balancing thorough and efficient exploration of the search space, ensuring that a diverse set of test cases that violate the MRs is generated.

B. Challenges

However, the development of this testing methodology entails addressing several key challenges.

- **Efficiency.** Real-world driving scenarios are inherently complex, involving a multitude of dynamic and static objects, such as vehicles, pedestrians, and environmental conditions. Thus, the vast space of potential test scenarios is defined by various attributes such as vehicle speed, weather conditions, and object positions. Efficiently exploring this space to identify scenarios that violate MRs is computationally intensive. This challenge is compounded by the high cost of simulations, necessitating a strategy that minimizes the number of scenarios to be tested while maximizing their violation-detection capability. For instance, instead of testing every possible combination of vehicle speeds and weather conditions, testing can prioritize key scenarios like a car making a sudden stop during heavy rain or encountering a pedestrian in foggy conditions. This approach allows testers to focus on critical, high-risk scenarios, rather than wasting resources on every minor variation. However, finding what such high-risk scenarios are in practice is not an easy endeavor.

- **Metamorphic relation definition.** MT requires well-defined MRs that describe expected behaviors or invariants across different test scenarios. Defining comprehensive and meaningful MRs can be challenging, for example in ADSs where system behavior can be influenced by a wide range of factors. MRs must be specific enough to detect subtle unsafe behavior while being general enough to widely apply across various scenarios. However, MRs do not need to be perfect and complete to be useful and can be improved over time through the collective endeavor of domain experts.
- **Diversity of test scenarios.** Ensuring a diverse set of test scenarios is crucial for trustworthy testing.ADSs are likely to encounter many rare cases that were not considered during their training process [21]. A lack of diversity in test scenarios can lead to inadequate system evaluation and reduce the effectiveness of identifying violations. The challenge lies in developing mechanisms that promote diversity in the generated scenarios, thus covering a broader spectrum of potential real-world conditions, while still being guided towards problematic scenarios. For example, an ADS might need to be tested with a mix of pedestrian behaviors and varying weather conditions. One test could involve a pedestrian slowly crossing a busy road in clear daylight, while another might involve the same crossing at night in the rain. This variety ensures that the system is exposed to different lighting and environmental conditions, increasing the chances of identifying potential weaknesses in the system's response to real-world complexities.

Addressing these challenges requires a multifaceted approach that combines MT with advanced search techniques. Our proposed method employs CCEAs to generate and evaluate test scenarios, optimizing both the effectiveness and efficiency of the testing process, as justified in Section IV.

III. Related Work

In this section, we explore existing research pertaining to the issue of system-level testing for ADSs. Given the components of our approach, we begin by reviewing search-based testing methods, followed by an examination of MT approaches. Finally, we briefly highlight Adaptive Stress Testing (AST) techniques for system-level testing of ADSs.

A. Search-Based Testing for ADSs

One prominent approach to system-level testing of ADSs is search-based testing, which emphasizes the systematic exploration of the input parameter space of ADSs to identify subspaces that may lead to violations or unexpected behaviors.

Ben Abdessalem et al. [26] introduced a multi-objective search-based testing approach that uses neural network-based surrogate models to test advanced driver assistance systems in a simulated environment. In a later investigation [27], the researchers utilized a similar search-based method to detect feature interaction failures, such as conflicts between automated functionalities, demonstrating that their approach could identify a significantly higher number of interaction failures compared to baseline methods. Dreossi et al. [28] introduced a compositional framework for search-based testing, particularly designed for ADSs with machine learning components. Their method combines constraints from both the perception input domain and the overall system input domain, which reduces computational effort and improves the efficiency of finding counterexamples. In a practical setting, Li et al. [29] developed *AV-Fuzzer*, a testing framework that perturbs the maneuvers of traffic participants using a genetic algorithm to identify safety violations in the APOLLO platform [30]. Their results showed that *AV-Fuzzer* outperformed random fuzzing and AST by uncovering a broader range of unique safety-critical scenarios and detecting violations more efficiently. Kolb et al. [31] proposed a collection of fitness function templates for search-based testing of ADSs in intersection scenarios, extending previous work on highway scenarios [32]. Through comparison with random testing baselines, their study demonstrated that the adapted fitness functions effectively guided the search toward generating diverse and safety-critical intersection scenarios. Luo et al. [33] introduced *EMOOD*, an evolutionary search-based testing approach designed to generate test scenarios that reveal diverse combinations of requirements violations in ADSs. Birchler et al. [34], [35] explored test case prioritization methods to optimize the regression testing process in ADSs, showing that their approach improved early fault detection and testing efficiency over baseline methods.

The relationship between test inputs and system behavior has also been a research focus. Riccio and Tonella [36] introduced the concept of a *frontier of behaviors*, which marks the input boundary (threshold) at which the ADS begins to exhibit abnormal behavior. Zohdinasab et al. [37], [38] further refined this concept by using *Illumination Search* [39] to explore the feature space, leading to more effective scenario identification. Castellano et al. [40] investigated the impact of different road representations on search-based testing for lane-keeping systems, concluding that road curvature and orientation are key factors influencing system behavior.

Further advancements in search-based testing have been achieved through the development of enhanced algorithms. For instance, Gambi et al. [41] proposed a novel approach that integrates procedural content generation (PCG) with search-based testing to systematically generate virtual road scenarios for evaluating lane-keeping functionalities. Goss and Akbaş [42] applied an *Eagle Strategy*, which first broadly samples the state space to locate critical areas, then shifts to a focused local search around detected areas to define critical scenario boundaries more precisely. Zheng et al. [43] proposed a quantum genetic algorithm to reduce the necessary population size for scenario generation.

Considering the high computational cost of ADS simulations, surrogate models have been developed to accelerate the testing process. Ben Abdessalem et al. [26] created a surrogate model that links scenario parameters to fitness functions, facilitating the reduction of non-critical parameters. Likewise, Batsch et al. [44], Beglerovic et al. [45], and Sun et al. [46] explored different surrogate models to enhance the efficiency of search-based testing. More recently, Haq et al. [19] introduced *SAMOTA*, a surrogate-assisted many-objective optimization approach that accelerates the detection of safety violations by using surrogate models to approximate the simulator, thereby reducing computational costs while effectively guiding the search toward critical scenarios.

In conclusion, search-based testing has proven effective in identifying critical failures in ADSs through systematic exploration of the input parameter space. Its strengths lie in the ability to generate diverse but targeted test cases, often uncovering edge-case scenarios that other testing methods may overlook. However, current state-of-the-art search-based techniques for system-level testing of ADSs remain computationally intensive, particularly when applied to large-scale, complex scenarios with high-dimensional input spaces and high-fidelity ADS simulations. While advancements such as surrogate models have helped reduce some of these computational costs, existing methods continue to face difficulties in efficiently exploring large parameter spaces to meet testing objectives. These challenges emphasize the need for further advancements to improve the scalability of search-based ADS testing frameworks.

B. Metamorphic Testing for ADSs

Metamorphic Testing (MT) [47], [48] has become a vital technique for validating ADSs, particularly when one is struggling with the oracle problem.

At the module level, MT has been widely used to evaluate the robustness and reliability of perception modules in ADSs. Yang et al. [49] introduced *MetaLiDAR*, an automated MT framework for LiDAR-based ADSs, focusing on object-level transformations to generate realistic follow-up point clouds. The framework defines three MRs—object insertion, deletion, and affine transformations—targeting the evaluation of LiDAR-based object detection modules by ensuring consistency in detected object properties before and after transformations. Following up on this, they introduced *MetaSem* [50], an MT approach that uses semantic-based transformations, such as adding, removing, or replacing scene elements (e.g., traffic signs, road markings, and traffic signals) to evaluate the consistency of system behaviors, resulting in the detection of more errors compared to prior methods. Zhou and Sun [51] demonstrated the effectiveness of MT in detecting fatal errors in the LiDAR perception module of ADSs, revealing that even small, randomly added LiDAR data points outside the region of interest could cause the system to overlook nearby obstacles.

MT has also been adapted for system-level testing. Tian et al. [2] introduced *DeepTest*, a tool that leverages MT by defining transformation-specific MRs (e.g., variations in lighting or weather conditions) to detect erroneous behaviors in DNN-driven ADSs. Zhang et al. [52] extended this with *DeepRoad*, using Generative Adversarial Networks (GAN) to generate realistic weather conditions for MT,which enabled the detection of more inconsistent steering behaviors across scenarios compared to *DeepTest*. Pan et al. [53] introduced a MT approach for ADSs specifically in foggy conditions by defining MRs based on fog density and direction. In another study, Han and Zhou [54] applied metamorphic fuzz testing, to automatically generate and evaluate unexpected scenarios. It uses MRs as a filtering mechanism to differentiate true failures from false positives (alarms), when unexpected changes occur in the simulated environment. Cheng et al. [12] introduced *Decictor*, a scenario-based MT framework designed to evaluate the optimal decision-making of ADSs. *Decictor* uses a novel MR to detect non-optimal decision scenarios (NoDSs) by applying non-invasive mutations to the environment without altering the original optimal path of the ADS, thereby revealing instances where the ADS deviates from making optimal choices. While *Decictor*'s methodology is categorized as system-level testing, its focus is primarily on evaluating the routing decisions of ADSs with limited types of scenarios and MRs. This narrow focus restricts its ability to assess the system's behavior in a broader range of driving situations, such as those involving complex traffic interactions and varying weather conditions.

In summary, MT has proven to be an effective technique for identifying errors in both the module-level and system-level testing of ADSs, especially when facing the oracle problem. Its primary advantage lies in its ability to define MRs that capture expected system behaviors under transformed input conditions, allowing for the detection of subtle inconsistencies without the need for a full oracle. However, MT can be limited by the complexity of designing effective MRs that fully and precisely capture safety-critical behaviors, particularly in dynamic driving scenarios, which involve continuously changing traffic, environmental conditions, and agent interactions. This challenge is further compounded by the difficulty of automatically identifying effective MRs, which, as noted in a recent survey [55], remains an open research problem requiring deep domain expertise and a thorough understanding of system behavior. Existing practices indicate that human expertise is still essential, as manual effort is often necessary to establish the foundation for generating MRs [55]. However, unlike full system specifications, MRs do not need to be complete or perfect to effectively drive testing. Incomplete or imperfect MRs can still successfully detect undesirable system behaviors, making them a valuable practical tool despite their limitations.

C. Adaptive Stress Testing

Adaptive Stress Testing (AST) is a test case generation approach that dynamically adjusts its focus by prioritizing certain test cases and allocating resources accordingly [11]. The core of this approach lies in the policy used to assign priorities to different test scenarios.

Koren et al. [56] implemented AST for ADSs by designing a priority assignment policy that considers the disparity between the expected and the observed behavior of the system. Their results demonstrated that using AST with Reinforcement Learning (RL) effectively uncovered high-probability failure scenarios with fewer simulator calls, showing promising results in both efficiency and effectiveness. In subsequent work, Corso et al. [57] introduced an enhanced priority assignment policy based on the concept of Responsibility-Sensitive Safety (RSS) [58], a formal framework that outlines idealized behaviors to prevent collisions in scenarios. The researchers used a reward augmentation technique to incorporate RSS into the AST's reward function, prioritizing scenarios where the ADS behavior significantly diverges from the idealized, collision-free behaviors defined by RSS. Their results showed that reward augmentation with RSS uncovered a broader range of diverse, high-relevance failure scenarios, improving its ability to identify critical safety violations. Baumann et al. [59] explored the use of RL, to identify and generate more critical scenarios within the context of overtaking maneuvers. Moreover, RL has been combined with RSS rules to further refine the generation of edge cases, as demonstrated by Karunakaran et al. [60].

In conclusion, AST can be used to effectively generate critical scenarios by focusing on high-risk cases, especially through priority policies and RL. However, AST is limited by the complexity of defining priority policies, significant computational costs from repeated scenario similarity evaluations, and dependence on predefined safety models like RSS, which may restrict the exploration of unanticipated failures.

D. Discussion

The state-of-the-art suggests that MT and search-based testing are two prominent approaches for system-level testing of ADSs, each with distinct strengths and limitations. Specifically, search-based testing, when well-designed, is effective at uncovering corner cases by systematically exploring the parameter space, while MT is particularly useful for identifying subtle problems in system behavior through well-defined MRs.

Therefore, a compelling research direction is the integration of these two approaches into a unified framework, designed to maximize their strengths while addressing their weaknesses. In such a combined approach, MT could address the oracle problem by providing meaningful behavioral expectations to guide the search process. Conversely, the systematic exploration capabilities of search-based testing could enhance the effectiveness of MT, turning mild violations identified through MRs into severe ones by extending them along high-risk parameter dimensions. Therefore, this combination could potentially lead to a more effective and efficient testing framework.

The central question, which this paper is the first to address, is then how to optimally combine MT and search-based testing into a cohesive framework that is both effective in detecting unsafe behavior and scalable for complex ADSs. The following sections describe our proposed novel approach in detail.

IV. Our Approach

This section details our proposed solution to the problem described in Section II.

Given the high-dimensional nature of the search space derived from real-world scenarios, a conventional search-based algorithm (e.g., evolutionary algorithm) might be insufficient to explore it effectively within practical time and computational limits [61]. To address this, we employ a Cooperative Co-Evolutionary Algorithm (CCEA), which decomposes the problem into lower-dimensional subproblems, each solved by independently evolving populations. This decomposition not only enables parallelism but also increases the efficiency of the search [61].

We must therefore reframe our objective as a CCEA, where, based on the problem, source scenarios and perturbations are made to be two independent populations, but yet collaborate to discover complete solutions, i.e., test cases that violate predefined MRs.

In the following subsections, we first describe the representation of MRs, alongside the representation of the two separate populations within the context of the search problem (Section IV-A). We then introduce a fitness evaluation framework, which incorporates both a joint fitness function [62], [63], evaluating the extent of violation in a complete solution formed by collaboration between individuals from the two populations, and an individual fitness function [62], [63], assessing the contribution of each individual within the two populations (Section IV-B). Ultimately, we unveil our innovative method founded on CCEAs that leverages the aforementioned representation and fitness functions (Section IV-C), and present the involved genetic operators (Section IV-D).

A. Representations

Before detailing the representation of the two separate populations, we first introduce the representation of the MRs used in the search, followed by a detailed description of the two populations. We consider two populations: one representing source scenarios and the other representing perturbations derived from a predefined set of MRs. These perturbations are subsequently applied to the source scenarios to generate follow-up scenarios.

1) Metamorphic relations: Generally, an MR can be expressed as a relation $\mathcal{R}$ that defines a relationship between a sequence of inputs $\langle x_1, x_2, \ldots, x_n\rangle$ and their respective outputs $\langle f(x_1), f(x_2), \ldots, f(x_n)\rangle$, where f represents the target function or algorithm [47]. A common special case occurs when $n=2$, meaning that $\mathcal{R}$ imposes a relationship between two inputs $\langle x_1, x_2\rangle$ and their outputs $\langle f(x_1), f(x_2)\rangle$ formulated as $\langle x_1, x_2, f(x_1), f(x_2)\rangle \in \mathcal{R}$ or simply $\mathcal{R}(x_1, x_2, f(x_1), f(x_2))$. This general formulation does not impose constraints on how the two inputs and outputs should relate to each other. A special case defines $\mathcal{R}$ in terms of an input relation ir and an output relation or, where ir specifies a relationship between inputs x_1 and x_2, and or dictates the corresponding relationship that must hold between $f(x_1)$ and $f(x_2)$ when x_1 and x_2 satisfy ir:

$$\mathcal{R}(x_1, x_2, f(x_1), f(x_2)) := ir(x_1, x_2) \wedge or(f(x_1), f(x_2))$$

In this formulation, an MR can be perceived as a statement that determines how outputs should behave in response to specific changes in inputs.

In our work, the input relation ir describes the relationship between the source scenario and the follow-up scenario, e.g., adding one pedestrian in the field of view of the ego vehicle within the source scenario. Based on the input relation, we are able to generate corresponding metamorphic transformations to apply to the source scenario. Note that the input relation is abstract, described in natural language, while the derived metamorphic transformations are precise, specifying exact perturbations to objects (e.g., position or speed). Therefore, distinct metamorphic transformations can be derived from the same input relation.

The output relation or specifies how the outputs of the source inputs relate to those of the follow-up inputs [64]. When the output relation is not fulfilled, the MR is violated, indicating unsafe behavior from the ADS. We employ three output relations in our search process, i.e., *invariance*, *increasing*, and *decreasing* relations. *Invariance* output relations stipulate that the difference between the results of source and follow-up scenarios should remain within a predefined, typically small, threshold. *Increasing* output relations stipulate that the results of follow-up scenarios should exceed those of source scenarios, while *decreasing* output relations imply the opposite.

Formally, the representation of an MR can be defined as the tuple $mr := (ir, or)$. In our search, we use a set of MRs that share the same output relation or to guide the search, which can be defined as $MR := \{mr_i = (ir_i, or) \mid i \in \mathbb{N}^*\}$. The set of metamorphic transformations derived from a set of MRs constitutes a perturbation as elaborated later in this subsection.

2) Scenarios: A scenario can be modeled as a vector consisting of real and integer values. In the context of ADSs, a scenario encompasses the ego vehicle, the trajectory, the environment (e.g., the weather), and other dynamic (e.g., vehicles and pedestrians) and static (e.g., obstacles) objects [65], [66]. Each of them has multiple attributes of various types, including real numbers and categorical values that can be encoded as integers. The scenario domain model for our case study is illustrated in Fig. 1. In practice, this is (partly) determined by what can be observed and controlled in the underlying simulator, i.e., CARLA simulator [67]. From the model, we can see that a scenario comprises one ego vehicle, zero or more other objects (including vehicles, pedestrians, and static objects), as well as a defined mission and weather conditions. In contrast to other studies [66], [68], we use a finer-grained level of simulation control, implying a larger scenario size due to the increased number of parameters that must be adjusted by the search algorithm. In practice, this is necessary if one wants to have precise MRs. For instance, we exert precise control over the specific positions of vehicles within the scenario by defining the precise three-dimensional coordinates, while other studies often limit their control to the number or approximate direction of vehicles, in the field of view of the ego vehicle or in the opposite lane.

Formally, let e be the ego vehicle and its attributes, W be a set of waypoints that characterizes the trajectory, A be a set of global attributes, e.g., weather and brightness, B be a set of static objects, e.g., traffic signs, and D be a set of dynamic objects, e.g., pedestrians and vehicles. The representation of a scenario can be defined as the tuple $s := (e, W, A, B, D)$, where:

$$\begin{aligned}
e &:= (mod_e, pos_v, rot_e, v_e) \\
W &:= \{w_i \mid i \in \mathbb{N}^*\} \\
A &:= \{a_i \mid i \in \mathbb{N}^*\} \\
B &:= \{b_i := (mod_i, pos_i, rot_i) \mid i \in \mathbb{N}^*\} \\
D &:= \{d_i := (mod_i, pos_i, rot_i, v_i) \mid i \in \mathbb{N}^*\}.
\end{aligned}$$

Here, mod represents the type of the object, e.g., the model of the vehicle such as a motorcycle or a car, pos represents the position of the object in three-dimensional space, rot represents the direction the object is facing in three-dimensional space, and v represents the speed of the respective objects.

Each attribute (weather, brightness, object speed, etc.) is limited to a predefined range that reflects realistic conditions and is aligned with CARLA's valid parameters. These ranges are also configurable, allowing testers to constrain or expand the space of possible values to suit different testing requirements or fidelity levels.

To facilitate scenario evolution, we explicitly separate global scenario attributes A from object attributes B and D. This independence helps ensure that modifying global attributes does not affect individual objects' parameters and vice versa, thus simplifying crossover and mutation. Furthermore, the attributes within B and D are designed to be independent from each other, so modifying one object does not unintentionally influence others. Additionally, we use relative object positions w.r.t. the ego vehicle to ensure scenario consistency when the ego vehicle's starting position changes.

3) Perturbations: A perturbation represents the changes applied to a source scenario, which can be used to generate a follow-up scenario by applying these changes sequentially. In particular, given a set of MRs, we encode a perturbation as a sequence of metamorphic transformations to a source scenario: $q := \langle c_1, c_2, \ldots, c_k \rangle$. Each entry c_i in q is derived from a unique input relation ir_i from the set of MRs, indicating a metamorphic transformation applied to that source scenario. Each entry c_i represents no change or a change to either a global attribute in A, a static object in B, or a dynamic object in D. Such changes can involve the *addition* of an object, or the *deletion* or *replacement* of an existing object in the scenario. Note that an entry c_i can be a no-op, meaning that no change is applied to the source scenario for that specific metamorphic transformation. This allows perturbations to be selectively constructed, as not every available transformation needs to be included in a given perturbation. However, we ensure that at least one metamorphic transformation is applied to the source scenario to generate a follow-up scenario, ensuring meaningful variation.

B. Fitness Function

Our objective is to define a fitness function that can both effectively direct the search towards solutions violating the predefined MRs and maximize the diversity of generated solutions.

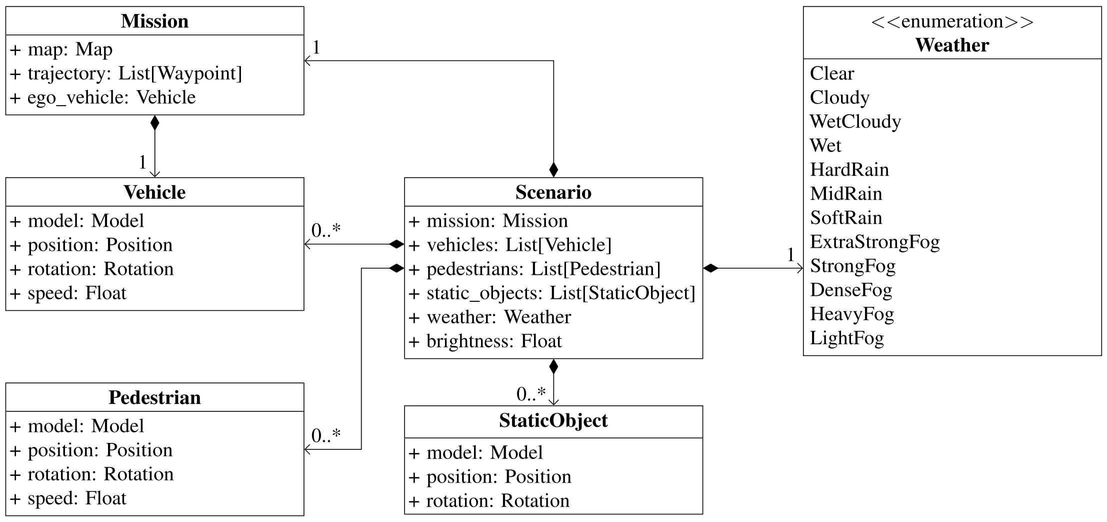

Fig. 1. Scenario domain model for the case study.

To achieve this, we first define how to measure the extent of MR violations between source and follow-up scenarios, followed by an optimization strategy to maximize diversity.

In the context of scenario-based simulation, a scenario's outcome comprises time series data capturing ego vehicle properties at regular intervals, e.g., speed or position at each timestamp. Consequently, the extent of violation is determined by comparing these time series across scenarios. Nevertheless, the time series data of the ego vehicle in source and follow-up scenarios may not always be synchronized. For instance, differences in pedestrian distances from the ego vehicle can lead to variations in the timing of stops, resulting in desynchronized speed data between the scenarios. To overcome such an effect and enable meaningful comparisons, we use *Dynamic Time Warping (DTW)*, a method that finds the optimal alignment between two time series, even when they are not perfectly synchronized [69]. Unlike direct pointwise comparisons, *DTW* finds an optimal alignment between two time series by minimizing the cumulative distance between their matched points.

Definition 2 (DTW alignment): Given two time series, a source scenario time series $S^t = \{s^t_1, s^t_2, \ldots, s^t_n\}$ and a follow-up scenario time series $Q^t = \{q^t_1, q^t_2, \ldots, q^t_m\}$, *DTW* constructs a set of matched points W^t that form the optimal alignment between two time series, constrained by the following conditions:

- *Boundary Constraints*: The start and end points must align.
- *Monotonicity*: The alignment maintains the temporal order of points.
- *Warping Window*: A predefined constraint to limit excessive stretching or compressing.

The optimal alignment under these constraints is achieved by minimizing the cumulative distance between matched points:

$$DTW(S^t, Q^t) = \min_{W^t} \sum_{(i,j)\in W^t} (s^t_i - q^t_j)^2$$

Our approach applies *DTW* to align corresponding points in the time series of the source scenario s and the follow-up scenario $q(s)$. This alignment results in a set of matched points $(t_s, t_q) \in W^t$ that provides an optimal alignment between the time series data of the two scenarios. To ensure that the alignment remains realistic and does not lead to extreme distortions, we impose the Sakoe-Chiba global constraint [70] on *DTW*. This constraint restricts how much a point in one time series can be shifted relative to the other, allowing only a limited range of alignment deviations. Consequently, the alignment maintains the original time order while allowing for small timing differences between the time series.

Given the set of matched points based on *DTW*, we can calculate the extent of violation between the time series of the source and follow-up scenarios. However, not all matched points are relevant for determining the extent of the violation. For example, if a pedestrian enters the ego vehicle's field of view but is initially far away, the ego vehicle's speed may remain unaffected at these early points. Consequently, evaluating matched points at this stage would not accurately reflect a violation, as a slowdown is not yet expected. To address this, we introduce the concept of *critical interval*, which defines the interval during which we expect the MR condition to be met. The *critical interval* is therefore specific to the input relation of the MR in use. For instance, if the input relation requires the ego vehicle to reduce speed when a pedestrian enters its field of view, the *critical interval* would be the period when the pedestrian is close enough to necessitate a slowdown. Conversely, if the input relation specifies that the ego vehicle should slow down under certain weather conditions, the *critical interval* would encompass the entire scenario, as the slowdown is expected at every timestamp. To determine such intervals in practice, simulation-based testing approaches can leverage detailed contextual information (e.g., exact positions of actors, weather conditions) provided by the simulator. In our case, we utilize contextual information provided by the simulator to measure the distance between the ego vehicle and the objects within its field of view at each timestamp to determine the *critical interval*. In our approach, we only consider matched pairs of points identified by *DTW*, where at least one point falls within the *critical interval*, discarding pairs where neither point is within this interval. This choice ensures that we focus on the most relevant portions of the scenario where the MR condition clearly applies, reducing the risk of false violations caused by ambiguous or borderline cases. Finally, we average the extent of violation across matched pairs within the *critical interval*. The extent of violation for each matched pair is quantified by the function $diff_{or}$, which depends on the specific output relation or of the MR in use. This function captures the difference between the actual outcome in the follow-up scenario and the expected outcome derived from the source scenario. The definition of $diff_{or}$ varies according to the type of output relation, as described in Section IV-A1, and thus the extent of violation is computed accordingly. For an *invariance* output relation, the difference between the source point t_s and the follow-up point t_q should remain within a specified threshold. This threshold can be defined either as a percentage θ of t_s or as an absolute value ϕ, with the extent of violation calculated as $diff_{or} := |t_q - t_s| - t_s * \theta$ or $diff_{or} := |t_q - t_s| - \phi$, respectively. For an *increasing* output relation, the follow-up point t_q is expected to exceed the source point t_s by a specified percentage θ or by an absolute value ϕ. In this case, the extent of violation is calculated as $diff_{or} := t_s * (1 + \theta) - t_q$ or $diff_{or} := t_s + \phi - t_q$, respectively. For a *decreasing* output relation, the follow-up point t_q should be lower than the source point t_s by a specified percentage θ or by an absolute value ϕ, with the extent of violation defined as $diff_{or} := t_q - t_s * (1 - \theta)$ or $diff_{or} := t_q - t_s + \phi$, respectively.

Definition 3 (Extent of MR violation): Given a source scenario s and a perturbation q derived from a set of MRs with the same output relation or, the extent of violation of the output relation or is defined as follows:

$$E(s, q) := \frac{1}{|T_{ci}|} \sum_{(t_s, t_q) \in T_{ci}} diff_{or}(t_s, t_q) \tag{2}$$

where T_{ci} denotes the set of pairs of matched points (t_s, t_q) such that at least one of t_s or t_q is within the *critical interval (ci)*. In Eq. (2), the pair (s, q) forms a complete solution, and E can be regarded as the joint fitness function, which should be maximized in our problem. Eq. (2) quantifies how much the follow-up scenario deviates from the expected behavior defined by the MR. It does so by comparing the outputs of the source and follow-up scenarios at critical moments, i.e., time intervals where the MR should hold. The function $diff_{or}(t_s, t_q)$ measures how much the actual response in the follow-up scenario differs from the expectation. The formula then averages these differences over the entire *critical interval* to produce a single number that summarizes the extent of MR violation. A higher value means a greater deviation from the expected behavior, signaling a more critical issue.

Following the definition of the violation extent, or the joint fitness function for complete solutions, we introduce the fitness function that evaluates each individual's contribution within the two populations. This fitness function is particularly valuable for mechanisms that operate on each population separately, allowing optimization of each component based on its unique role in the complete solution.

Definition 4 (Violation detection fitness function): Given an individual i, the violation detection fitness function $fitness$ is defined as follows:

$$fitness(i) := \begin{cases} \max_j(E(i, q_j)) & \text{if } i \text{ is a scenario} \\ \max_j(E(s_j, i)) & \text{if } i \text{ is a perturbation} \end{cases} \tag{3}$$

where j indexes individuals in the other population. In other words, the violation detection fitness is the maximum joint fitness value across all complete solutions involving individual i, and $fitness$ can be regarded as the individual fitness function. Previous studies [63], [71], [72], [73] have demonstrated the effectiveness of using the maximum joint fitness value with collaborators from the other population, rather than the average or minimum.

To make the search more exploratory and prevent the algorithm from converging too rapidly to suboptimal solutions, there are several methods to optimize the diversity of solutions. *Fitness Sharing* [74], [75] and *Fitness Clearing* [76] are amongst the best known and prevalent diversity mechanisms. The basic idea of fitness sharing is that individuals in the same niche (within a certain niche radius) of the search space should be penalized for being too similar. Nonetheless, fitness sharing faces challenges in problems with multiple optima because it struggles to distinguish global optima (the best possible solutions) from local optima (suboptimal solutions that are better than their immediate neighbors but not globally optimal) [77]. This difficulty arises due to the way fitness sharing distributes resources, i.e., allocates fitness values, among similar individuals, which can cause local optima to receive undue attention, making it harder for the search process to prioritize and converge on the true global optima. Fitness clearing resembles fitness sharing; however, it is grounded in the concept of limited environmental resources [76], aligning with our requirements mentioned in Section II. Instead of distributing resources among all individuals within the same niche, fitness clearing allocates them solely to the best members in the niche. Practically, the capacity $\kappa \in \mathbb{N}^*$ of a niche is used to indicate the maximum number of individuals it can accept. Consequently, fitness clearing maintains the fitness of the top κ individuals in the niche while removing the remaining individuals from the population. Individuals whose fitness is cleared are largely excluded from selection in the next generation, preventing them from effectively contributing offspring. This process eliminates competition within the niche, allowing only the dominant individuals to propagate their genetic material. While it is theoretically possible for cleared individuals to be selected through mechanisms like tournament selection, the likelihood of this happening is very low due to their cleared fitness, which significantly reduces their chances of being favored during selection. Fitness clearing significantly reduces the impact of genetic drift [77], which causes the population to trend towards individuals resembling those with high fitness [75], [78], [79], [80]. Furthermore, fitness clearing can be combined with elitism strategies to retain the top-performing individuals within the niches that are maintained throughout generations, and it is less complex than fitness sharing. As demonstrated by previous studies [77], [81], fitness clearing proves to be one of the most effective approaches.

To determine whether the distance between two individuals (either scenarios or perturbations) falls within a niche radius, we define a heterogeneous distance [82] function to quantify the distance between them. In the representation we have defined, a scenario is an amalgamation of numerical and categorical values, as elaborated in Section IV-A. A perturbation is characterized by a series of metamorphic transformations, which themselves are also composed of numerical and categorical values.

Definition 5 (Heterogeneous distance): Given two individuals i and j, the heterogeneous distance between i and j is defined as follows:

$$HD(i,j) := \sqrt{\sum_{L\in\{B,D\}} HD^2_{set}(i_L, j_L) + HD^2_{attr}(i_A, j_A)} \tag{4}$$

where HD_{set} is the heterogeneous distance between two object sets, i.e., the set of static objects B or dynamic objects D of a scenario, which is defined in Eq. (5). HD_{attr} is the heterogeneous distance between two attributes sets, e.g., global attributes A of a scenario, which is defined in Eq. (6).

$$HD_{set}(I,J) := \sqrt{\sum_{i\in I} \min_{j\in J}(HD^2_{attr}(i,j))} \tag{5}$$

where i and j denote the objects in set I and J, respectively. Note that the size of set I needs to be no smaller than that of set J for this equation to be applicable. In short, this equation calculates the sum of the shortest heterogeneous distances between each object of the longer set and each object of another set, where the heterogeneous distance of objects is computed by HD_{attr}.

$$HD_{attr}(i,j) := \sqrt{\sum_{a=1}^{m} dist^2_a(i_a, j_a)} \tag{6}$$

where m is the number of attributes. The function $dist_a(i,j)$ returns a distance between the two values i and j for attribute a, which is defined as follows:

$$dist_a(i,j) := \begin{cases} \frac{|i-j|}{a_{\max}-a_{\min}} & \text{if } a \text{ is numerical} \\ \phi & \text{if } a \text{ is categorical and } i \neq j \\ 0 & \text{if } a \text{ is categorical and } i = j \end{cases} \tag{7}$$

where $a_{\max}$ and $a_{\min}$ represent the maximum and minimum values for attribute a, respectively. When considering categorical attributes, a distance $\phi \in (0,1]$ is assigned when the values are distinct, while a zero distance is assigned when the values are the same.

To penalize an individual's fitness when other individuals are in close proximity, we need to first define a clearing radius and a niche capacity, as proposed by Petrowski [76]. Specifically, the fitness of the dominant individuals (i.e., winners) is preserved while the fitness of all the other individuals is cleared so that only the winners can survive to the next generation. To leverage this concept, we introduce the notion of fitness clearing.

Algorithm 1: Fitness clearing procedure.

```
Input: Population $P$
       Clearing radius $\sigma$
       Niche capacity $\kappa$
1  $sortFitness(P)$;
2  for $i \leftarrow 1$ to $|P|$ do
3      if $isValid(fitness(P[i]))$ then
4          Number of winners $nbWinners \leftarrow 1$;
5          for $j \leftarrow i+1$ to $|P|$ do
6              if $isValid(fitness(P[j]))$ and
                 $HD(P[i], P[j]) < \sigma$ then
7                  if $nbWinners < \kappa$ then
8                      $nbWinners \leftarrow nbWinners + 1$;
9                  else
10                     Clear $fitness(P[j])$;
11                 end
12             end
13         end
14     end
15 end
```

Definition 6 (Fitness clearing): Given a population P, a clearing radius $\sigma \in \mathbb{R}^+$, and a niche capacity $\kappa \in \mathbb{N}^*$, the fitness clearing procedure for the population P is defined in Algorithm 1

The algorithm begins by sorting the population P in descending order of fitness (line 1), so the highest-fitness individuals are processed first. For each individual $P[i]$ in P (line 2), the algorithm checks if its fitness is valid (line 3), ensuring that previously cleared individuals are not reconsidered. If the fitness of $P[i]$ is valid, it is counted as the first winner in its niche by initializing a counter $nbWinners$ to 1 (line 4). The algorithm then iterates over subsequent individuals $P[j]$ where $j > i$ (line 5) to check if they have valid fitness values and their heterogeneous distance from $P[i]$ is within the clearing radius σ (line 6). For each such individual $P[j]$, the $nbWinners$ counter is incremented (line 8) until it reaches the niche capacity κ, limiting the number of individuals that can occupy the same niche. Once $nbWinners$ reaches κ, any additional individuals within σ have their fitness values cleared (line 10), meaning that their values are set to undefined, thus preventing them from being selected or archived. This process continues until all individuals in P have been processed, ensuring that no niche has more than κ high-fitness individuals, thus maintaining diversity while preserving top solutions.

However, the clearing radius σ is usually unknown in many optimization problems [83], making it difficult to determine an appropriate value without prior knowledge or extensive tuning. To address this challenge, we borrow the concept of a dynamic fitness sharing method proposed by Tan et al. [84], which adaptively adjusts the clearing radius based on the characteristics of the population at each generation.

Definition 7 (Dynamic clearing radius): Given a generation n, we define the dynamic clearing radius $\sigma^{(n)}$ at generation n as follows:

$$\sigma^{(n)} := \frac{HD^{(n)}_{\max}}{2 \times N^{(n)}} \tag{8}$$

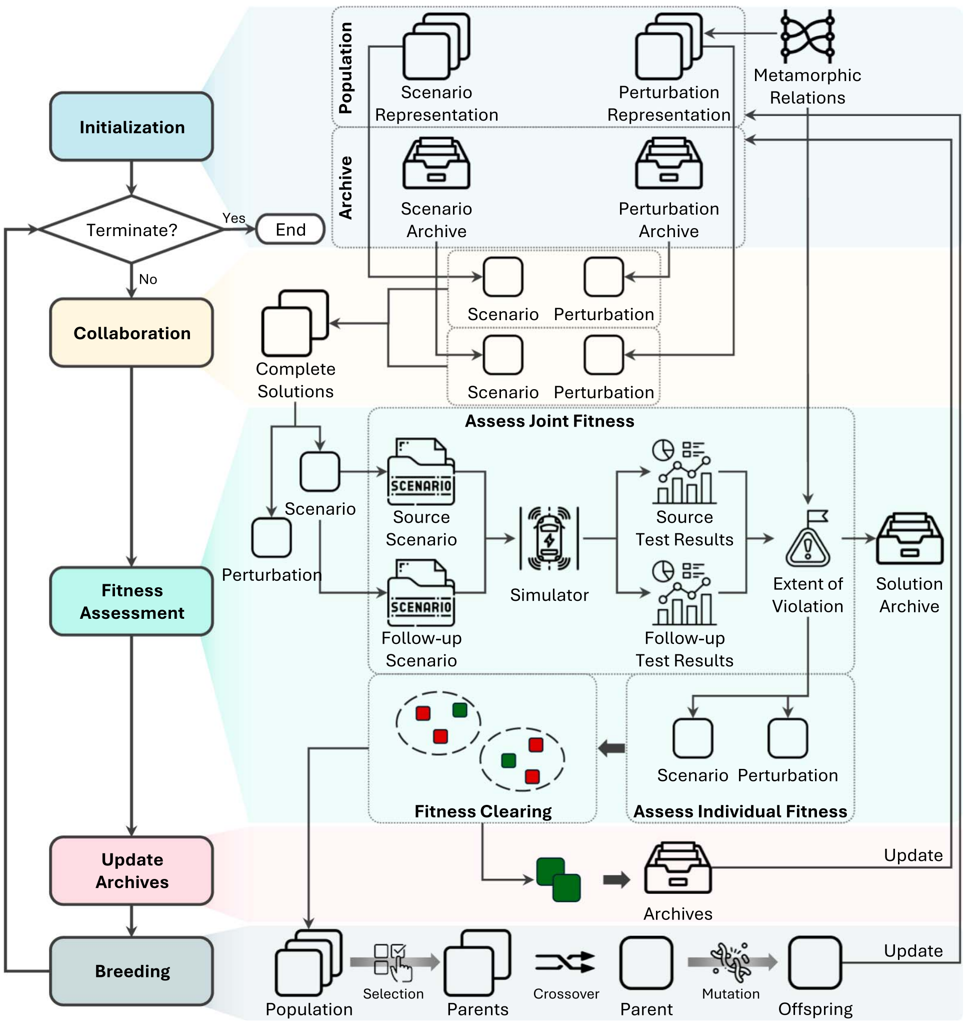

Fig. 2. Overview of *CoCoMEGA*.

where $N^{(n)}$ represents the population size at generation n, while $HD_{\max}^{(n)}$ represents the maximal HD value among individuals in the population at generation n, reflecting the spread of the population. The dynamic clearing radius adapts by scaling with the population's spread and inversely adjusting with the population size. By scaling with $HD_{\max}^{(n)}$, the clearing radius adapts to the overall distribution of individuals. This ensures that larger spreads result in proportionally larger radius, promoting wider dispersion of individuals across the search space, and encouraging exploration. As the population converges and spreads shrink, the radius becomes smaller, allowing finer distinctions between individuals in high-potential regions of the search space. This adaptive behavior helps the algorithm remain responsive to different search phases without requiring manual tuning.

Dividing by $2 \times N^{(n)}$ inversely scales the clearing radius with the population size. This serves as a scaling factor that adapts competition pressure to the number of individuals, ensuring the radius is small enough to retain a reasonably large proportion as winners.

To develop an intuition for the dynamic clearing radius $\sigma^{(n)}$, consider a scenario where a population of N individuals is confined within a sphere of radius R. In this setup, the maximum heterogeneous distance in the population $HD_{\max}^{(n)}$ can be approximated by the diameter of the sphere $2R$. Substituting this approximation into the formula for $\sigma^{(n)}$, we obtain:

$$\sigma^{(n)} \approx \frac{2R}{2N} = \frac{R}{N}. \tag{9}$$

This indicates that the clearing radius for each individual is approximately $\frac{1}{N}$ of the sphere's radius. As the population size N increases, the clearing radius decreases proportionally, dividing the sphere into finer niches and ensuring that the individuals are distributed across the sphere in a balanced way.

By eliminating the need for a fixed parameter, the dynamic clearing radius $\sigma^{(n)}$ simplifies the optimization process and reduces dependency on prior knowledge about the often unpredictable distances between individuals in the search space [84].

C. Algorithm

Building upon the scenario and perturbation representations detailed in Section IV-A and the violation detection fitness function introduced in Section IV-B, this section presents our proposed algorithm, *Cooperative Co-evolutionary MEtamorphic test Generator for Autonomous systems (CoCoMEGA)*. The overall workflow of *CoCoMEGA* is shown in Fig. 2. This approach combines an archive-based CCEA with MT to effectively and efficiently explore high-dimensional and complex search spaces. *CoCoMEGA* operates by decomposing the complex search space into separate populations for scenarios and perturbations, enabling these populations to evolve independently but collaborate to form complete solutions. The evolution of each population is guided by the MRs, which not only address the oracle problem by providing a structured way to evaluate solutions but also accelerate search convergence by focusing on meaningful variations. Furthermore, diversity optimization strategies are applied during the archiving process to ensure that the archived individuals promote a broad exploration of the search space, effectively capturing a diverse set of complete solutions.

Algorithm 2 shows the pseudocode of *CoCoMEGA*. As input, it receives a population size n and a maximum archive population size l and returns an archive X of distinct complete solutions.

The algorithm commences by randomly initializing populations of scenarios P_s and perturbations P_q at lines 1 and 2, respectively. Corresponding population archives, X_s and X_q, are also initialized at lines 3 and 4. Additionally, an empty archive of complete solutions X is created at line 5. The algorithm then co-evolves P_s and P_q, guided by X_s and X_q, until a $stopping\ condition$ is reached (line 6). This co-evolutionary process involves three iterative steps: enumerate∗evaluating the fitness of individuals in both P_s and P_q, and updating the archive X with complete solutions and their joint fitness values as determined by the simulator (line 7);updating X_s and X_q based on individual fitness values and l (lines 8–9); andbreeding P_s and P_q, and merging the resulting populations with X_s and X_q to form the next generation of P_s and P_q (lines 10–11). The algorithm concludes by returning the final archive X (line 13).

1) Fitness assessment: The function $assessFitness$ initially computes joint fitness values of complete solutions constructed by combining individuals from P_s, P_q, X_s, and X_q. To optimize computational efficiency, complete solutions identical to those in X (i.e., previously generated complete solutions) are excluded from re-simulation. Subsequently, the fitness value of each individual is determined based on the joint fitness of the complete solutions in which it is included. Finally, fitness clearing is performed for each population as detailed in Section IV-B.

The pseudocode for $assessFitness$ is provided in Algorithm 3, with inputs the population of scenarios P_s, the population of perturbations P_q, the population archive of scenarios X_s, the population of perturbations X_q, and the niche capacity κ. The function returns updated populations P_s and P_q with individual fitness values, along with an updated archive X incorporating newly created complete solutions and their corresponding fitness values.

The algorithm initiates by generating an initial set of complete solutions through combinations of individuals from both populations and population archives (line 1). To be precise, each individual in P_s collaborates with every individual in X_q, and vice versa for P_q and X_s. For each generated complete solution cs (line 2), if $cs \notin X$, i.e., cs has not been previously evaluated (line 3), its joint fitness is computed using the simulator (line 4–5) and cs with its evaluated result is added to X (line 6). In practice, it is possible that the simulator fails to evaluate a complete solution due to conflicts among its parameters, such as overlapping object locations. In such cases, the complete solution is considered as *invalid* and its fitness value is deemed undefined. Such invalid solutions will then be ignored in subsequent steps. To avoid redundant simulations, the algorithm stores the results of both source and follow-up scenarios derived from a complete solution. Upon updating X using CS, for each population $P \in \{P_s, P_q\}$ (line 9) and for each individual $i \in P$ (line 10), the algorithm first assigns the maximum joint fitness

Algorithm 2: Cooperative co-evolutionary algorithm.

```
Input:  Population size $n$
        Maximum size of archive population $l$
Output: Archive of complete solutions $X$
1  Population of scenarios $P_s \leftarrow initPopulation(n)$;
2  Population of perturbations $P_q \leftarrow initPopulation(n)$;
3  Archive of scenarios $X_s \leftarrow P_s$;
4  Archive of perturbations $X_q \leftarrow P_q$;
5  Archive of complete solutions $X \leftarrow \emptyset$;
6  while not(stopping condition) do
7      $P_s, P_q, X \leftarrow assessFitness(P_s, P_q, X_s, X_q, X)$;
8      $X_s \leftarrow updatePopulationArchive(P_s, l)$;
9      $X_q \leftarrow updatePopulationArchive(P_q, l)$;
10     $P_s \leftarrow breed(P_s) \cup X_s$;
11     $P_q \leftarrow breed(P_q) \cup X_q$;
12 end
13 return $X$;
```

Algorithm 3: assessFitness.

```
Input:  Population of scenarios $P_s$
        Population of perturbations $P_q$
        Archive of scenarios $X_s$
        Archive of perturbations $X_q$
        Archive of complete solutions $X$
        Niche capacity $\kappa$
Output: Updated population of scenarios $P_s$
        Updated population of perturbations $P_q$
        Updated archive of complete solutions $X$
1  Set of complete solutions
     $CS \leftarrow collaborate(P_s, P_q, X_s, X_q)$;
2  foreach Complete solution $cs \in CS$ do
3      if $cs \notin X$ then
4          $simulate(cs)$;
5          $cs.fitness \leftarrow assessJointFitness(cs)$;
6          $X \leftarrow X \cup \{cs\}$;
7      end
8  end
9  foreach Population $P \in \{P_s, P_q\}$ do
10     foreach Individual $i \in P$ do
11         $i.fitness \leftarrow assessIndividualFitness(i, X)$;
12     end
13     $fitnessClearing(P, \kappa)$;
14 end
15 return $P_s, P_q, X$;
```

Algorithm 4: updatePopulationArchive.

```
Input: Population $P$
       Maximum size of population archive $l$
Output: Population archive $X_p$
1  $X_p \leftarrow \{popBestFitnessIndividual(P)\}$;
2  while $|X_p| < l$ and $|P| > 0$ do
3      Best individual $best \leftarrow \Box$;
4      foreach Individual $i \in P$ do
5          if $div(X_p \cup \{i\}) > div(X_p \cup \{best\})$ then
6              $best \leftarrow i$;
7          end
8      end
9      $X_p \leftarrow X_p \cup \{popIndividual(best)\}$;
10 end
11 return $X_p$;
```

Algorithm 5: breed.

```
Input: Population $P$
Output: Population of Offspring $O$
1  $O \leftarrow \emptyset$;
2  while $|O| < |P|$ do
3      Parents $p_1, p_2 \leftarrow selection(P)$;
4      Offspring $o_1, o_2 \leftarrow crossover(p_1, p_2)$;
5      $o_1 \leftarrow mutate(o_1)$;
6      $o_2 \leftarrow mutate(o_2)$;
7      if $div(O \cup \{o_1\}) > div(O \cup \{o_2\})$ then
8          $O \leftarrow O \cup \{o_1\}$;
9      else
10         $O \leftarrow O \cup \{o_2\}$;
11     end
12 end
13 return $O$;
```

value of the complete solutions involving i as the individual fitness of i (line 11). This strategy of assessing individual fitness using an elitist approach (i.e., choosing the highest fitness value) aligns with findings from previous experimental research [61], [85]. Subsequently, the algorithm performs fitness clearing for P with a given niche capacity κ. The algorithm finishes by returning the revised P_s, P_q, and X (line 15).

2) Update population archive: The function $updatePopulationArchive$ updates the population archives X_s and X_q for the next generation. These archives are instrumental in guiding the search process, as each individual must collaborate with archive members to form complete solutions whose fitness is subsequently evaluated. To balance exploitation and exploration, the function selects one best individual based on fitness values to ensure that high-fitness solutions are preserved and refined. Additionally, it selects other individuals to maximize the diversity of the population archive, encouraging broader exploration of the search space and reducing the risk of premature convergence to suboptimal solutions.

The pseudocode for updating the population archive is outlined in Algorithm 4. The algorithm accepts a target population P and a maximum size of population archive l as input, returning a population archive X_p of P with a size less than or equal to l.

The algorithm begins by creating a population archive X_p for population P utilizing the individual that possesses the highest fitness value from all the individuals in P (line 1). As long as the size of archive X_p is below the limit l and P is not empty (line 2), the algorithm repeatedly selects the most dissimilar individual leading to maximal population diversity (div) in the archive once added (line 3–8). Several metrics have been proposed to measure population diversity [86], e.g., *Spacing* [87], [88], *Maximum Spread (MS)* [89] and *Pure Diversity (PD)* [90]. These metrics are designed to require no additional problem-specific knowledge and maintain a computational complexity that is manageable, at most quadratic time complexity w.r.t. the population size. Among these metrics, *Spacing* evaluates diversity by calculating the minimum Euclidean distance of each individual to all others within the population. Specifically, it measures the standard deviation of these minimum distances, thereby assessing the uniformity of the population without considering the overall spread across the search space [86], [91]. Although this metric provides insight into how evenly individuals are distributed [92], it does not address the extent of spread across the search space. Thus, a population may exhibit high uniformity with a small spread if individuals are closely clustered, indicating limited exploration of the search space. In contrast, *MS* captures diversity by focusing solely on the extreme individuals within the population [86], [90], representing the widest range that the population spans in the search space. However, this metric lacks information about the distribution of individuals between these extremes and cannot fully evaluate how well the population covers the search space. This limitation makes *MS* insufficient for assessing overall population diversity [90]. To address these shortcomings, we select *PD* as the preferred metric for gauging population diversity. *PD* measures the total spread by summing the dissimilarities of individuals relative to the rest of the population in a greedy order, with priority given to the individual exhibiting maximal dissimilarity. This approach provides a holistic view of diversity, effectively reflecting the primary dissimilarities within the population and the extent to which it covers the search space.

3) Breed: The breeding algorithm is designed to generate new individuals by combining the genetic material of selected parents from the current populations. This process is crucial for exploring the search space and finding better solutions over successive generations. The algorithm leverages crossover and mutation operators to create offspring that inherit traits from their parents while introducing variability to enhance diversity.

The algorithm starts by initializing an empty set O to store the offspring (line 1). The goal is to fill this set with new individuals until its size matches the original population P (line 2). First, two parent individuals, p_1 and p_2, are chosen from the existing population P (line 3). The selection process typically uses tournament selection, which provides individuals with higher fitness a greater likelihood of being chosen. Next, the selected parents undergo a crossover operation to produce two offspring, o_1 and o_2 (line 4). The crossover operator combines parts of the parents' genetic material to create new individuals that inherit traits from both parents. Each offspring o_1 and o_2 is then subjected to mutation (line 5–6). The mutation operator applies minor, random modifications to the genetic material of the offspring. This step is essential for preserving genetic diversity in the population and enabling the algorithm to investigate new regions of the search space. The details of the genetic operators are introduced later in Section IV-D.

After the offspring are mutated, the algorithm evaluates the diversity contribution of each offspring to the current set O (line 7), where the function div measures the diversity of a given population. The objective is to maximize diversity within the offspring population to avoid premature convergence to suboptimal solutions. The offspring that contributes more to the diversity of the population is added to O (line 7–10), ensuring that the new population maintains a good balance between exploring new solutions and the exploitation of established effective solutions. These steps are repeated until the set O contains as many individuals as the original population P. This ensures that the population size remains constant across generations. Finally, the algorithm returns the new population O (line 13), which will replace the old population P in the next generation.

D. Genetic Operators

Given the complexity of the representation, conventional crossover and mutation operators designed for simple vector representations are inadequate. Therefore, we devised novel operators for scenario and perturbation representations.

1) Crossover: In this section, we define the crossover operator for scenarios as well as for perturbations.

Definition 8 (Scenario crossover): Given two parent scenario representations $s_1 = (e_1, W_1, A_1, B_1, D_1)$ and $s_2 = (e_2, W_2, A_2, B_2, D_2)$, the offspring scenarios s_1' and s_2' are defined as follows:

$$s_1 = (e_1, W_1, A_1', B_1', D_1')$$
$$s_2 = (e_2, W_2, A_2', B_2', D_2')$$

where A_1', A_2' are two offspring resulting from a uniform crossover [85], [93] between A_1 and A_2 with randomly selected crossover points, since there are no preferred specific crossover points. This method enables a fine-grained exchange of genetic material, making it well-suited to our heterogeneous and complex representation. By independently mixing genes, uniform crossover enhances diversity in offspring and enables a broader exploration of the search space. Similarly, B_1' and B_2' are derived from B_1 and B_2, D_1' and D_2' are derived from D_1 and D_2.

The crossover operator for scenarios is designed to combine characteristics from two parent scenarios while maintaining the integrity of essential components. Specifically, the parameters of the ego vehicle (e_1, e_2) and the trajectory (W_1, W_2) remain unchanged, while a uniform crossover is performed between the global attributes (A_1, A_2), the static objects (B_1, B_2) and dynamic objects (D_1, D_2). Intuitively, the scenario crossover function generates new test scenarios by swapping global attributes (e.g., weather, brightness) and exchanging objects (e.g., vehicles, pedestrians, signs) between parent scenarios. This approach ensures that the offspring inherit traits from both parents while preserving the overall structure of the scenarios.

Definition 9 (Perturbation crossover): Given two parent perturbation representations $q_1 = \langle c_1^1, c_2^1, \ldots, c_k^1 \rangle$ and $q_2 = \langle c_1^2, c_2^2, \ldots, c_k^2 \rangle$, the offspring perturbations q_1' and q_2' are generated through pairwise crossover between q_1 and q_2, which is defined as follows:

$$q_1' = \langle c_1'^1, c_2'^1, \ldots, c_k'^1 \rangle$$
$$q_2' = \langle c_1'^2, c_2'^2, \ldots, c_k'^2 \rangle$$

where each pair of metamorphic transformations c_i^1 and c_i^2 undergoes uniform crossover with randomly selected crossover points. For example, if c_i^1 represents adding a vehicle d_i^1 to the scenario and c_i^2 represents adding a vehicle d_i^2, the uniform crossover randomly swaps some of the parameters of d_i^1 and d_i^2 to generate two offspring, $c_i'^1$ and $c_i'^2$. Similarly, if both c_i^1 and c_i^2 modify a global attribute, the crossover operation may simply swap them. However, if c_i^1 and c_i^2 modify different types of parameters (e.g., static objects, dynamic objects, or global attributes), the crossover operator preserves them as they are.

2) Mutation: Similar to the crossover, the mutation operator is defined for scenario and perturbation representations separately.

Definition 10 (Scenario mutation): Given a scenario representation $s = (e, W, A, B, D)$ with both numerical and categorical attributes, the polynomial mutation [94] and integer randomization mutation [85] are applied to scenario attributes, respectively. Similar to the scenario crossover, the parameters of the ego vehicle e and the trajectory W remain unchanged. Consequently, the scenario after mutation is represented as $s' = (e, W, A', B', D')$, where A', B', and D' denote the mutated global attributes, static objects, and dynamic objects, respectively. Furthermore, we also designed two additional operators that in turns *add* or *remove* dynamic or static objects in a scenario. The probability of applying an operator and triggering a mutation are both configurable.

The *add* operator incrementally incorporates multiple objects based on a hyperbolic distribution. Specifically, it adds a randomly generated object into the scenario s with a probability of η. Upon its addition, a second randomly generated object is added with a probability of η^2. This process continues until no further objects are appended. Generally, during each mutation iteration n, a new object is added with a probability of η^n. This non-linear operator draws inspiration from the add operator used in EvoSuite [95], [96] for unit-test suite or case generation.

The *remove* operator randomly removes multiple dynamic or static objects from the scenario s. Similar to the *add* operator, the *remove* operator also removes objects following a hyperbolic distribution. This operator is designed to counteract the potential bloating effect [97], [98], i.e., the length of scenarios incrementally expanding over successive generations.

Intuitively, scenario mutation may introduce random changes to a scenario by modifying global attributes (e.g., adjusting weather conditions or brightness) and altering actors (e.g., shifting vehicle positions, changing pedestrian speed, or adding/removing objects). These subtle variations help explore a wider range of driving conditions while preserving the core structure of the scenario.

Definition 11 (Perturbation mutation): Given a perturbation representation $q = \langle c_1, c_2, \ldots, c_k \rangle$, the polynomial mutation and integer randomization mutation are applied to each metamorphic transformation c_i with a certain mutation probability, as the algorithm does to mutate scenarios. Since there are three distinct operations, namely *addition*, *deletion*, and *replacement*, as mentioned in Section IV-A3, the mutation process varies. If the transformation c_i represents a change to a global attribute, the algorithm applies a polynomial or an integer randomization mutation to generate c_i', which assigns a new random value to the same global attribute. In case c_i represents an operation on static or dynamic objects, perturbation mutation introduces random changes to c_i. This can involve altering the parameters of a perturbation (e.g., changing a vehicle's initial speed) or disabling/enabling a perturbation entirely. This process ensures a diverse set of scenario variations while maintaining meaningful modifications.

3) Selection: The selection operator for both populations is standard tournament selection, the most prevalent selection technique in evolutionary algorithms [93]. It selects candidate individuals for breeding based on a simple ranking of their fitness values. Note that hyperparameter values for crossover, mutation, and selection (e.g., tournament size, crossover rate, and mutation rate) can significantly impact breeding performance. A detailed discussion of hyperparameter tuning in our evaluation is provided in Section V-B3.

V. Empirical Evaluation

In this section, we empirically assess the effectiveness and efficiency of *CoCoMEGA* against two baseline methods in generating test cases for MR violation detection. Specifically, we answer the following Research Questions (RQs).

RQ1: How *effectively* can *CoCoMEGA* find test cases violating MRs compared to baseline methods?

To answer RQ1, we evaluate the performance of different search methods in terms of the number and diversity of complete solutions identified that violate the predefined MRs within a given search budget.

RQ2: How *efficiently* can *CoCoMEGA* find test cases violating MRs compared to baseline methods?

To answer RQ2, we examine the speed at which different search methods identify complete solutions that violate the predefined MRs.

RQ3: How *effectively* do the diversity mechanisms of *CoCoMEGA* enhance the identification of diverse test cases violating MRs?

To answer RQ3, we assess the performance of *CoCoMEGA* with and without diversity mechanisms, in terms of the number and diversity of complete solutions which violate the predefined MRs within a given search budget.

A. Baseline Methods

To the best of our knowledge, this is the first study to generate test cases using multiple MRs to identify MR violations in ADSs, and there are no direct baselines. Consequently, we compare *CoCoMEGA* against two baseline methods, i.e., *Random Search (RS)* and *Standard Genetic Algorithm (SGA)* [99].

- ***Random Search (RS):*** It randomly generates complete solutions. *RS* is commonly used as a baseline for evaluating search-based methods, given its simplicity and effectiveness for simple problems. The results of *RS* will thus reveal the difficulty of the search problem.
- ***Standard Genetic Algorithm (SGA):*** It generates complete solutions without utilizing distinct populations for scenarios and perturbations. The results of *SGA* will show how effective *CoCoMEGA* is compared to a standard and simpler search method.

We include *RS* as a simple, non-optimized baseline to provide a lower-bound comparison. While it is not an advanced optimization approach, *RS* serves as a "sanity check" that highlights whether *CoCoMEGA* yields significant improvements beyond mere chance. This ensures that any gains in performance can be attributed to our method's evolutionary design rather than random variation alone. We also compare *CoCoMEGA* with *SGA*, which does not separate scenarios and perturbations into distinct populations. This helps isolate the effect of cooperative co-evolution in *CoCoMEGA*, demonstrating whether the additional complexity of evolving two populations independently truly enhances exploration and solution quality.

B. Experimental Setup

We now describe the experimental setup used to address the defined RQs, including the simulation platform utilized, the involved MRs, and the settings of *CoCoMEGA* and baseline methods.

1) Simulation platform: For the experiments, we utilized CARLA [100], a widely adopted open-source simulator for research in autonomous driving. CARLA offers realistic urban environments, configurable weather conditions, and dynamic objects such as vehicles and pedestrians, allowing for the simulation of diverse driving scenarios. In addition, we integrated INTERFUSER [101], a safety-enhanced autonomous driving framework designed to improve scene understanding through multi-modal sensor fusion, which is one of the top-performing ADSs on the CARLA leaderboard [102] during our evaluation period. INTERFUSER combines data from LiDAR and multi-view cameras to generate a comprehensive understanding of the driving environment.

Our resource-intensive experiments were carried out on a remote server equipped with 8 CPU cores and 16 threads, alongside 2 high-performance GPUs, each with 24GB of memory. To maximize computational efficiency, we parallelized scenario execution by running 6 Docker containers, each hosting a separate CARLA instance (version 0.9.10.1) and leveraging CUDA 11.8 for GPU-accelerated computations, allowing multiple simulations to run concurrently.

2) MRs involved in the experiments: We employed the MRs from Table I, which were extracted from the relevant literature [52], [53], [64], [103], [104] and are both applicable to ADS testing and compatible with the simulation platform we use. For

TABLE I
MRS INVOLVED IN THE EXPERIMENTS

Category	MR	Description
GP_1 Speed Reduction	MR_1[*]	If a pedestrian appears on the roadside, then the ego vehicle should slow down.
	MR_2[*,§]	If the driving time changes into night, then the ego vehicle should slow down.
	MR_3[§]	Adding a vehicle in the front of the ego vehicle, the speed of the ego vehicle should decrease in t1% to t2%.
	MR_4[§]	Adding a pedestrian in the front of the ego vehicle, the speed of the ego vehicle should decrease in t1% to t2%.
	MR_5[§]	Changing from sunny to rainy, the speed of the ego vehicle should decrease in t1% to t2%.
GP_2 Environmental Invariance	MR_6[†]	The density of fog (light fog, heavy fog, dense fog, strong fog, and extra strong fog) in a foggy driving scene will not affect the steering angle of the autonomous driving systems.
	MR_7[‡]	No matter how the driving scenes are synthesized to cope with different weather conditions (sunny and rainy), the driving steering angle is expected to be consistent.
GP_3 Actor Invariance	MR_8[‖]	In a given driving scenario where the ego vehicle detects a target obstacle and attempts to avoid a collision, the ego vehicle should keep the steering unchanged when the speed of the ego vehicle changes (increased or decreased by a certain factor).
	MR_9[‖]	In a given driving scenario where the ego vehicle detects a target obstacle and attempts to avoid a collision, the ego vehicle should keep the steering unchanged the same when adjusting (i.e., scaled down or up) the size of the ego vehicle.
	MR_{10}[‖]	In a given driving scenario where the ego vehicle detects a target obstacle and attempts to avoid a collision, the ego vehicle should keep the steering unchanged when adjusting (i.e., scaled down or up) the size of the target obstacle.
	MR_{11}[‖]	In a given driving scenario where the ego vehicle detects a target obstacle and attempts to avoid a collision, the ego vehicle should keep the steering unchanged when changing the position of the ego vehicle.
	MR_{12}[‖]	In a given driving scenario where the ego vehicle detects a target obstacle and attempts to avoid a collision, the ego vehicle should keep the steering unchanged when changing the speed of the target obstacle.
	MR_{13}[‖]	In a given driving scenario where the ego vehicle detects a target obstacle and attempts to avoid a collision, the ego vehicle should keep the steering unchanged when adding additional actors.

* These MRs are derived from study [64].
† This MR is derived from study [53].
‡ This MR is derived from study [52].
§ These MRs are derived from study [103].
‖ These MRs are derived from study [104].

instance, an MR that specifies replacing buildings with trees is incompatible with our platform due to its limitations.

In our evaluation, we categorized MR_1 to MR_5, MR_6 to MR_7, and MR_8 to MR_{13} into three distinct groups, respectively. Each group is characterized by a common output relation. This categorization allows our approach to effectively identify MR violations that impact multiple MRs within these groups. For GP_1, the output relation was defined as a decrease in the speed of the ego vehicle by at least 20%. For GP_2 and GP_3, the output relation was defined as maintaining a steering angle within one degree.

The thresholds of output relations used in our study are derived from the established literature and refined through a pilot experiment conducted prior to the main study. This preliminary step helped identify and resolve potential discrepancies and ensure proper calibration of the MRs for our experiments. While our focus is not on optimizing these thresholds, they were chosen to support experimentation based on practical considerations and empirical observations from the pilot experiment. In practice, domain experts determine these thresholds based on industry standards and domain knowledge to ensure that the MRs accurately reflect expected system behavior.

3) Settings of CoCoMEGA and baseline methods: For all methods, we established a small population size of seven individuals. This decision takes into account the high cost of fitness evaluation, which averages two minutes per individual (scenario). The literature commonly recommends a small population size for costly fitness functions [105], [106]. Regarding the selection operator utilized in *CoCoMEGA* and *SGA*, we employed tournament selection, which is the most commonly used selection method in modern genetic algorithms [93]. This method is particularly useful as it allows direct control over selection pressure by adjusting the tournament size. In our case, we selected a tournament size of 3.This choice is supported by prior studies, as small tournament sizes (2 or 3) are widely used in the literature [107] to moderate selection pressure, balance exploration and exploitation, and thereby prevent early convergence to suboptimal solutions. Furthermore, when using genetic algorithms in practice, a tournament size of 3 has been observed to potentially enhance efficiency compared to a size of 2 by reducing function evaluations and improving CPU usage [108]. The mutation rate used in *SGA* is set to 0.2, which is considered relatively high; however, it helps to prevent genetic drift in the context of a small population size [109]. Lastly, we established the crossover rate to 0.8, a value commonly recommended [110]. For the hyperparameters used in *CoCoMEGA*, since there are no suggested values for CCEAs, we decided to tune them by performing a pilot experiment with smaller budgets. As a result, we used the following hyperparameters for *CoCoMEGA* : maximum population archive size $= 3$, crossover rate for individuals $= 0.8$, mutation rate for individuals $= 0.2$.

For all methods, we established the total number of simulations to 200 as the search budget, which is sufficient to observe the convergence of effectiveness metrics based on our pilot experiment. Specifically, we monitored convergence by tracking the average fitness value of the population and archives across generations, identifying stagnation when improvements became negligible over successive generations, thus signaling convergence. As the majority of the execution cost is associated with running simulations, the computational budget for the experiments primarily relies on the number of simulations conducted. Therefore, we consider the total number of simulations to be the search budget. It is worth noting that for *CoCoMEGA* and *SGA*, the actual number of simulations may slightly exceed the predefined limit, due to population-based methods checking whether the search budget has been exhausted only after completing one generation. Furthermore, since the number of simulations per generation can vary across methods, the results cannot be perfectly aligned with the search budget. To address this misalignment, we use linear interpolation based on the search budget to align the results across methods. Specifically, for a given metric, we interpolate the values at points where the actual number of simulations exceeds or falls short of the predefined search budget. Linear interpolation estimates the metric value at the exact search budget by assuming a linear relationship between the metric and the number of simulations conducted. This approach provides a fair basis for evaluating all methods under a consistent computational budget.

C. RQ1: Effectiveness of CoCoMEGA

1) Methodology: To answer RQ1, we generate sets of complete solutions violating the predefined MRs using *CoCoMEGA* and other baseline methods within the same search budget. We compare the methods in terms of *DS* and *Solution Diversity (SD)*.

- ***Distinct Solutions (DS)*** denotes the number of distinct complete solutions the method finds that violate the given MRs within a certain search budget. Specifically, given a fitness threshold θ_f and a distance threshold θ_d, let CS_V denote the set of complete solutions produced by a search method V, satisfying the following conditions: 1) the fitness value of each complete solution in CS_V is greater than θ_f and 2) the pairwise distance between any two complete solutions in CS_V exceeds θ_d. Then, *DS* of V is defined as $DS(V) := |CS_V|$.
- ***Solution Diversity (SD)*** measures the diversity of the distinct complete solutions identified by the method, based on applied fitness thresholds θ_f and distance thresholds θ_d. Specifically, we quantify *SD* using the following three metrics:
  - ***Average Pairwise Distance (APD)*** represents the average heterogeneous distance (cf. Eq. (4)) between all pairs of distinct complete solutions identified under a given search budget. The heterogeneous distance between two complete solutions is the heterogeneous distance between their follow-up scenarios, generated by applying each solution's perturbation to its source scenario. This metric provides insight into the degree of variation among solutions in terms of scenario representation.
  - ***Metamorphic Relation Coverage (MRC)*** measures the percentage of predefined MRs covered by the complete solutions found within a specified search budget. This metric reflects how thoroughly the solutions explore potential violations relative to the predefined MRs.
  - ***Combinations of Metamorphic Relations (CMR)***: indicates the number of unique MR combinations that the complete solutions can violate within a given search budget. This metric offers insight into the capability of the complete solutions to expose a variety of undesirable behaviors characterized by combinations of MR violations, highlighting the breadth of testing.

Each of these metrics captures a distinct aspect of solution diversity. *APD* measures spatial diversity by quantifying how spread out solutions are in the search space but does not indicate whether different types of MR violations are covered. *MRC* addresses this by measuring the number of violated MRs, ensuring broad behavioral exploration. However, *MRC* does not account for cases where multiple MRs are violated simultaneously, which is captured by *CMR*. By combining these three metrics, we provide a more comprehensive assessment of solution diversity.

To gain a better understanding of how *DS* and *SD* varies based on different θ_f and θ_d thresholds, we vary θ_f and θ_d to analyze their impact. We set θ_f to 0.3, 0.5, 0.8, 1.0, 1.3, and 1.5. This range covers the median fitness value (approximately 0.3) to the 90th percentile (approximately 1.5) of the identified complete solutions. We excluded solutions with fitness values below 0.3, as they correspond to trivial violations, and those above 1.5, as they are few in number and exhibit a wide range of fitness values. For θ_d, we selected values from 0.0 to 1.7, with increments of 0.1. This range encompasses the median pairwise distance (approximately 1.7) of the identified complete solutions. Additionally, to address the randomness in search-based methods, we conduct each experiment 10 times per method.

2) Results: Based on GP_1, Fig. 3 visually compares the *DS* achieved by *CoCoMEGA*, *SGA*, and *RS* across 10 experimental runs, varying by fitness threshold θ_f and distance threshold θ_d. Each subplot displays θ_d on the x-axis and the average *DS* on the y-axis, with 95% confidence intervals represented as error bars.

Overall, for all three methods, *DS* values decrease as θ_f increases, reflecting that higher θ_f values led to only retaining more stringent solution quality, thus limiting the number of complete solutions meeting the criteria. When θ_f is set high (e.g., $\theta_f = 1.5$), *DS* values are low across all methods, indicating a lack of high-fitness solutions with severe violations. This observation implies that within the allocated search budget, the methods were unable to explore solutions with fitness values beyond 1.5.

For *CoCoMEGA*, *DS* values naturally decrease as θ_d increases, since stricter distinctiveness constraints inherently filter out more solutions. In contrast, *SGA* and *RS* show more consistent trends, especially at $\theta_f \geqslant 1.0$, where the algorithms can occasionally identify sufficiently different complete solutions, though the total number of the solutions remains limited. This implies that identifying MR violations is a challenging problem, which *RS* and *SGA* struggle to address effectively. *CoCoMEGA*, by comparison, identifies complete solutions across a broader range of distances.

Additionally, for a given θ_d, the gap between *CoCoMEGA* and the baseline methods (i.e., *SGA* and *RS*) generally widens as θ_f decreases. However, an exception occurs when θ_d is relatively high ($\theta_d \geqslant 1.2$), especially for low fitness thresholds ($\theta_f \leqslant 0.8$), where the average *DS* of *CoCoMEGA* is lower than that of the baseline methods. This outcome suggests that *SGA* and *RS* are more likely to find more trivial violations at greater distances from each other in these cases. In contrast, *CoCoMEGA* focuses on severe violations, resulting into more guided testing rather than maximizing diversity. This targeted approach is advantageous when maximizing diversity must be balanced with thoroughly exploring severe violations.

Our analysis shows that *CoCoMEGA* consistently outperforms *SGA* and *RS* in identifying severe violations when $\theta_f \geqslant 1.0$. For instance, under moderate thresholds ($\theta_f = 1.0, \theta_d = 1.0$), *CoCoMEGA* discovers on average 3.8 distinct MR-violating solutions within the specified budget, whereas *SGA* and *RS* detect about 1.8 and 2.1, respectively, corresponding to over a twofold improvement over *SGA* and more than 80% over

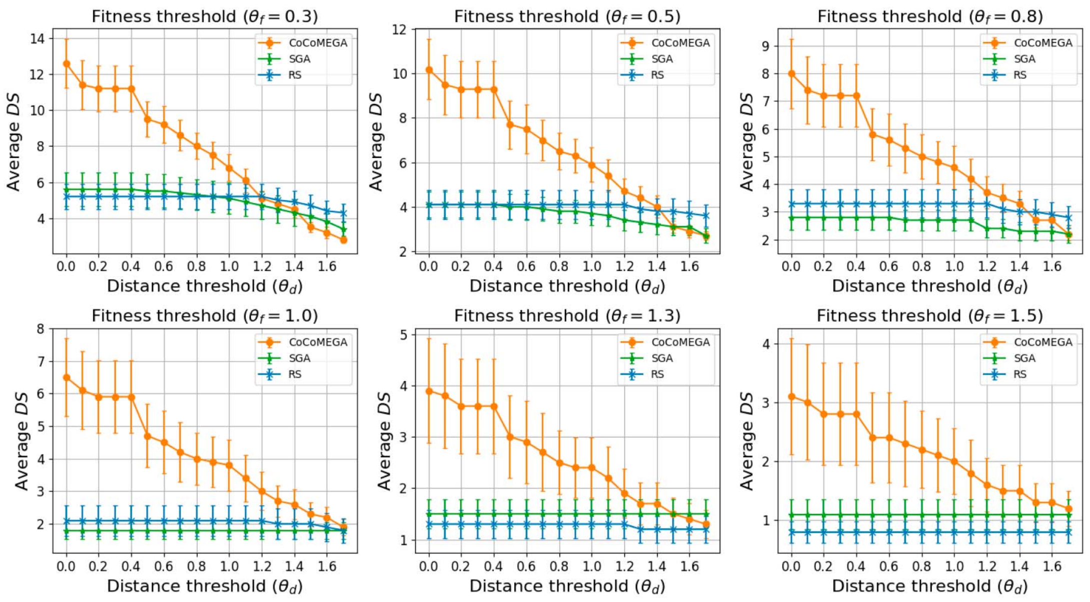

Fig. 3. *Distinct Solutions (DS)* vs. distance threshold (θ_d) across different fitness thresholds (θ_f). A higher *DS* indicates more distinct violating solutions discovered by the method. Each curve plots the average *DS* at different θ_d settings, under a specific fitness threshold θ_f. This figure reveals how each method balances the quantity and diversity of solutions as θ_f varies.

RS. Across all fitness and distance thresholds, the advantage remains consistent, with *CoCoMEGA* discovering, on average, 87% (p-value $< 10^{-31}$) more distinct solutions than *RS* and 83% (p-value $< 10^{-35}$) more than *SGA*. To assess the statistical significance of the observed differences, we conducted non-parametric Mann-Whitney U tests with Fisher's method for p-value correction. The results confirm *CoCoMEGA* 's clear advantage in discovering more distinct MR-violating solutions across most threshold configurations. Although in certain conditions with low fitness thresholds ($\theta_f \leqslant 0.8$) and high distance thresholds ($\theta_d \geqslant 1.2$), *RS* and *SGA* may excel at exploring the search space and yield higher *DS* values, these solutions tend to be of lower quality, as reflected by their low fitness, and thus of less practical interest.

While *DS* provides a measure of the quantity and quality of identified complete solutions based on fitness and distance thresholds, *APD* offers insight into the spatial diversity of these solutions by assessing how dispersed they are within the search space. To analyze the *APD* results, we examine two aspects: 1) the relationship between *APD*, fitness threshold θ_f, and distance threshold θ_d for *CoCoMEGA*, *SGA*, and *RS* across generations, and 2) the evolution of *APD* across generations under different θ_f and θ_d values. The first analysis helps us evaluate each method's ability to generate a diverse set of final solutions, while the second allows us to understand how effectively each method explores diverse regions of the search space over time.

Fig. 4 shows the relationship between *APD*, θ_f, and θ_d for *CoCoMEGA*, *SGA*, and *RS* across 10 experimental runs. Each subplot corresponds to a specific θ_d value, with θ_f values plotted along the x-axis. The y-axis represents the *APD*, with 95% confidence intervals represented by error bars. It is worth mentioning that the maximum fitness threshold has been adjusted to 1.0, as the number of solutions significantly diminishes beyond this fitness threshold, making it infeasible to measure *APD* accurately at higher θ_f values. Across all thresholds, *RS* consistently achieves higher *APD* values compared to both *CoCoMEGA* and *SGA*, suggesting that *RS* produces more widely spaced solutions. However, *RS* 's higher *APD* may not translate to solution quality or relevance, as *RS* lacks a guided search mechanism and generates solutions randomly.

When comparing *CoCoMEGA* and *SGA*, we observe that *CoCoMEGA* achieves *APD* values that are generally on par with *SGA* across all tested distance thresholds θ_d and fitness thresholds θ_f. This suggests that both methods produce complete solutions with similar levels of diversity within the search space. The comparable diversity between *CoCoMEGA* and *SGA* should be interpreted by considering the different mechanisms through which they aim to preserve diversity. Since *CoCoMEGA* uses separate populations for scenarios and perturbations, its candidate solutions at each generation are formed by combining individuals from these two populations. This setup results in significant commonalities among candidate solutions, as each candidate solution consists of a specific pairing of a scenario and a perturbation from their corresponding populations. In contrast, *SGA* operates within a single population, allowing each candidate solution to evolve independently. Consequently, *CoCoMEGA* is inherently more inclined to produce a set of complete solutions with lower diversity compared to *SGA*. However, Fig. 4 suggests that *CoCoMEGA* effectively compensates for this potential disadvantage, likely through its diversity optimization mechanisms, such as the population archive update strategy (cf. Section IV-C2) and fitness clearing (cf. Section IV-B). This indicates that *CoCoMEGA* 's design is robust enough to maintain high diversity even within the interdependent structure of its co-evolutionary framework.

Fig. 5 displays the *APD* across generations for *CoCoMEGA*, *SGA*, and *RS* under varying fitness thresholds θ_f and distance thresholds θ_d. Each subplot corresponds to a combination of

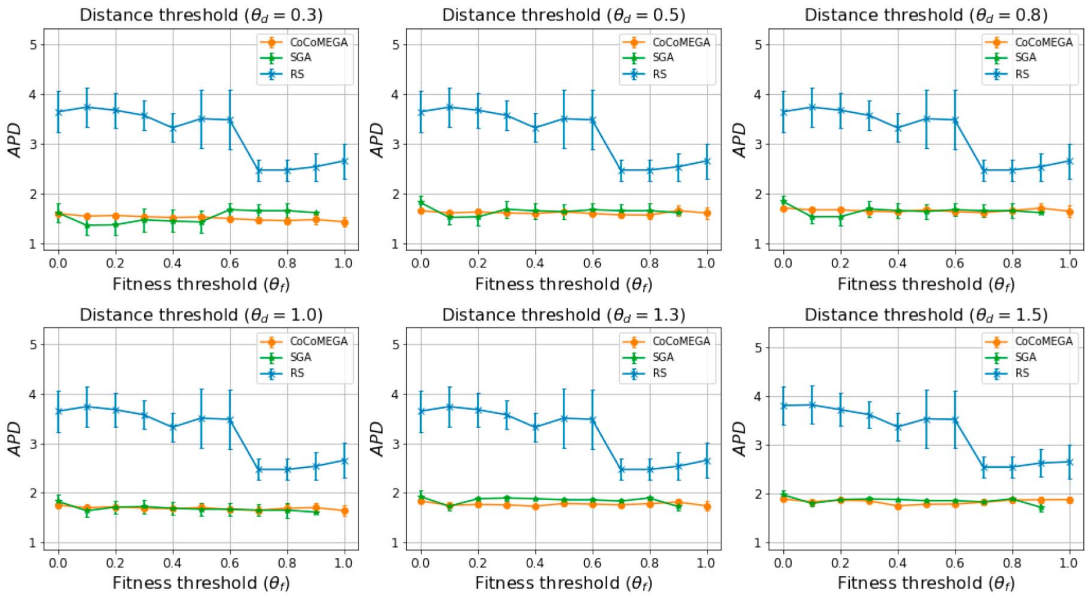

Fig. 4. *Average Pairwise Distance (APD)* vs. fitness threshold (θ_f) across different distance thresholds (θ_d). *APD* quantifies how spread out the solutions are. A higher *APD* indicates greater diversity among the found solutions. Each subplot corresponds to a certain distance threshold θ_d, and the y-axis shows the *APD* of discovered solutions under various θ_f values. This figure demonstrates how each method maintains diversity while striving for higher fitness (i.e., more severe violations).

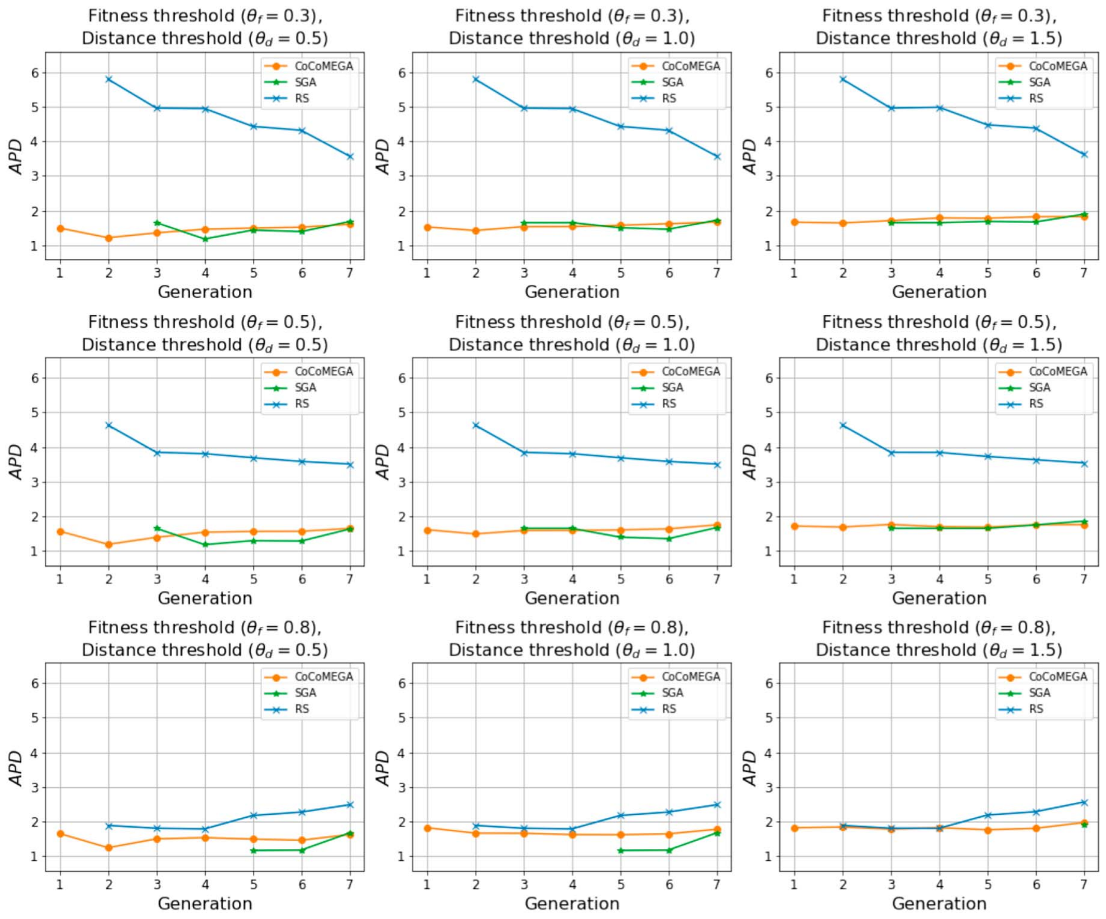

Fig. 5. Evolution of *Average Pairwise Distance (APD)* across generations. Higher *APD* values imply that solutions are more spread out in the search space, indicating broad exploration. Subplots differ by the chosen fitness threshold (θ_f) and distance threshold (θ_d). Each curve tracks the *APD* for a specific method over several generations of search. This figure illustrates how each method maintains or improves solution diversity over generations.

TABLE II
MRC AND CMR VALUES FOR DIFFERENT METHODS AT DIFFERENT VALUES OF θ_f AND θ_d

$\boldsymbol{\theta_d}$	**Method**	**Average** ***MRC*** $\boldsymbol{\pm CI_{0.95}}$ **(Average** ***CMR*** $\boldsymbol{\pm CI_{0.95}}$**)**					
		$\theta_f = 0.3$	$\theta_f = 0.5$	$\theta_f = 0.8$	$\theta_f = 1.0$	$\theta_f = 1.3$	$\theta_f = 1.5$
$\theta_d =$ 0.0	*CoCoMEGA*	**100.0 ± 0.0** (3.0 ± 0.3)	**98.0 ± 1.8** (2.6 ± 0.3)	**90.0 ± 7.2** (2.1 ± 0.3)	**78.0 ± 10.9** (1.8 ± 0.4)	**66.0 ± 12.3** (1.3 ± 0.3)	**56.0 ± 13.1** (1.0 ± 0.3)
	SGA	88.0 ± 7.2 (4.2 ± 0.5)	84.0 ± 8.8 (3.4 ± 0.5)	82.0 ± 8.7 (2.6 ± 0.4)	66.0 ± 10.8 (1.8 ± 0.3)	56.0 ± 11.7 (1.5 ± 0.3)	40.0 ± 11.4 (1.1 ± 0.2)
	RS	92.0 ± 7.2 (1.0 ± 0.0)	90.0 ± 9.0 (0.9 ± 0.1)	90.0 ± 9.0 (0.9 ± 0.1)	64.0 ± 13.4 (0.8 ± 0.1)	48.0 ± 12.9 (0.8 ± 0.1)	22.0 ± 8.3 (0.7 ± 0.1)
$\theta_d =$ 1.0	*CoCoMEGA*	**100.0 ± 0.0** (2.9 ± 0.3)	**98.0 ± 1.8** (2.6 ± 0.3)	**90.0 ± 7.2** (2.1 ± 0.3)	**78.0 ± 10.9** (1.8 ± 0.4)	**66.0 ± 12.3** (1.3 ± 0.3)	**56.0 ± 13.1** (1.0 ± 0.3)
	SGA	88.0 ± 7.2 (4.1 ± 0.5)	84.0 ± 8.8 (3.3 ± 0.4)	82.0 ± 8.7 (2.5 ± 0.4)	66.0 ± 10.8 (1.8 ± 0.3)	56.0 ± 11.7 (1.5 ± 0.3)	40.0 ± 11.4 (1.1 ± 0.2)
	RS	92.0 ± 7.2 (1.0 ± 0.0)	90.0 ± 9.0 (0.9 ± 0.1)	90.0 ± 9.0 (0.9 ± 0.1)	64.0 ± 13.4 (0.8 ± 0.1)	48.0 ± 12.9 (0.8 ± 0.1)	22.0 ± 8.3 (0.7 ± 0.1)
$\theta_d =$ 1.7	*CoCoMEGA*	**98.0 ± 1.8** (2.0 ± 0.3)	**94.0 ± 3.8** (2.1 ± 0.2)	86.0 ± 7.6 (1.6 ± 0.1)	**78.0 ± 10.9** (1.3 ± 0.2)	**64.0 ± 12.0** (1.1 ± 0.2)	**56.0 ± 13.1** (1.0 ± 0.3)
	SGA	88.0 ± 7.2 (3.1 ± 0.3)	84.0 ± 8.8 (2.5 ± 0.4)	80.0 ± 8.5 (2.1 ± 0.3)	66.0 ± 10.8 (1.8 ± 0.3)	56.0 ± 11.7 (1.5 ± 0.3)	40.0 ± 11.4 (1.1 ± 0.2)
	RS	92.0 ± 7.2 (1.0 ± 0.0)	90.0 ± 9.0 (0.9 ± 0.1)	**90.0 ± 9.0** (0.9 ± 0.1)	64.0 ± 13.4 (0.8 ± 0.1)	48.0 ± 12.9 (0.8 ± 0.1)	22.0 ± 8.3 (0.7 ± 0.1)

θ_f and θ_d, showing the progression of *APD* values over seven generations. It is notable that for some methods, certain *APD* values are absent in the initial generations, particularly at higher θ_d and θ_f levels, due to an insufficient number of complete solutions meeting the specified thresholds in the early generations. To address this limitation, the maximum fitness threshold has been capped at 0.8, as the number of solutions diminishes significantly beyond this point in the early generations, making further measurement impractical.

As expected, given that it does not include any form of guidance, across all settings, *RS* maintains consistently higher *APD* values compared to *CoCoMEGA* and *SGA*, aligning with the trends observed in Fig. 4. However, as θ_f increases, the gap between *RS* and the other methods gradually closes, suggesting that *RS* 's broad exploratory advantage diminishes under stricter fitness constraints, suggesting that many of the solutions it produces are less interesting from a practical standpoint. For *CoCoMEGA* and *SGA*, the *APD* remains relatively stable across generations, suggesting that both methods effectively maintain diversity levels while progressively enhancing solution quality. Under all tested configurations, *CoCoMEGA* and *SGA* demonstrate comparable *APD* values, reflecting similar levels of diversity within the structured, guided search frameworks of each method. This consistency in diversity indicates that both methods effectively balance exploration and refinement in their search processes, allowing them to preserve solution diversity over generations.

While *APD* captures the spatial diversity of solutions within the search space, *MRC* and *CMR* focus on behavioral diversity by measuring how thoroughly the complete solutions test predefined MRs and explore unique combinations of MR violations. Table II presents the relationship between *MRC* and *CMR* values across varying θ_f and θ_d values for each method under the same search budget. To capture the range of distance constraints, we selected three representative values for $\theta_d \in \{0.0, 1.0, 1.7\}$.

Across all methods and settings, except when $\theta_f = 0.8$, *CoCoMEGA* consistently achieves the highest *MRC* values, indicating it covers a greater percentage of predefined MRs than *SGA* and *RS*. This performance remains strong even as θ_f increases, though *MRC* declines, reflecting the increasing difficulty in covering MRs under more stringent fitness requirements.

While *SGA* has a slight edge in *CMR* values across certain settings (notably, with low fitness thresholds like $\theta_f = 0.3$), *CoCoMEGA* generally achieves competitive *CMR* values, particularly as θ_f increases. The *CMR* values for *RS* remain low across all settings, indicating its relative inefficiency in discovering diverse MR combinations.

Changing θ_d has a relatively small effect on the *MRC* values for *CoCoMEGA*, as it maintains high coverage across nearly all distance thresholds, although there is a slight decline at $\theta_d = 1.7$ and higher θ_f values. In contrast, *RS* shows significant decreases in both *MRC* and *CMR* as θ_f and θ_d increase, underscoring its limitations in maintaining *MRC* under more restrictive conditions.

CoCoMEGA stands out for its robustness across a range of θ_f and θ_d values, consistently achieving high *MRC*. This demonstrates its effectiveness in identifying complete solutions that thoroughly explore and violate the predefined MRs, especially under varying fitness and distance thresholds.

For GP_2, none of the three methods were able to identify complete solutions. A possible reason is that INTERFUSER may be particularly adept at handling weather variations and such changes have thus minimal impact on the steering angle of the ego vehicle.

For GP_3, the results align closely with those of GP_1, reinforcing our conclusions. Notably, some performance metrics exhibit further improvements, underscoring the robustness of *CoCoMEGA*. Given the similar trends and to avoid redundancy, a detailed breakdown, including all relevant figures and tables, is provided in the appendix.

Answer to RQ1

CoCoMEGA outperforms *SGA* by 83% and *RS* by 87% in identifying severe MR violations, as measured by the *DS* metric. Furthermore, it also achieves both good spatial and behavioral diversity, effectively covering diverse regions within the search space.

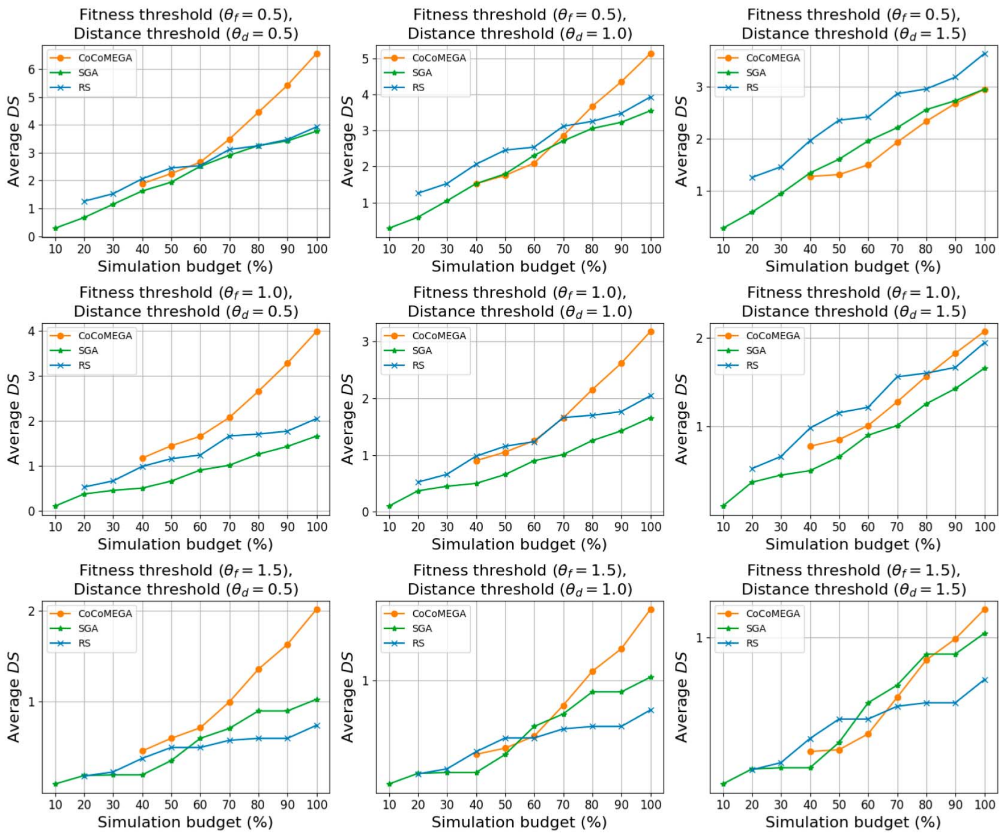

Fig. 6. *Distinct Solutions (DS)* vs. search budget. A higher *DS* indicates more distinct violating solutions discovered by the method. Each subplot shows *DS* at incremental percentages of the total simulation budget. Different curves represent different methods, compared in distinct fitness threshold (θ_f) and distance threshold (θ_d) settings. This figure highlights each method's efficiency: steeper or higher curves mean the algorithm finds more unique solutions earlier in the budget.

D. RQ2: Efficiency of CoCoMEGA

1) Methodology: To address RQ2, we compare *CoCoMEGA* with the baseline methods in terms of *DS* and *MRC* as a function of the search budget. In particular, we evaluate both *DS* and *MRC* across various search methods, gradually increasing the search budget from 10% to 100% in 10% increments. The methodology mirrors that of RQ1, using identical hyperparameters and performing 10 repetitions for each method, with the only difference being the varying search budget. By analyzing how *DS* and *MRC* values evolve across different methods and budget levels, we gain insights into the relationship between the search budget and the effectiveness of each method.

2) Results: We analyzed how the relationship between average *DS* and the percentage of the search budget used is affected by varying threshold values for θ_f and θ_d. Based on the experimental results in RQ1, we selected three representative $\theta_f \in \{0.5, 1.0, 1.5\}$ and $\theta_d \in \{0.5, 1.0, 1.5\}$ values that span the range of fitness and distance constraints. Each subplot of Fig. 6 tracks the effectiveness of these methods as the search budget incrementally increases from 10% to 100% of the total.

Across all subplots, a consistent trend is observed: *DS* values increase as the search budget grows. This is expected, as a higher budget provides more opportunities for each method to explore the search space, thus finding a greater number of *DS*. Across nearly all threshold settings, *CoCoMEGA* outperforms *SGA* and *RS*, achieving higher *DS* values even with smaller budgets. This indicates that *CoCoMEGA* is more efficient in exploring the search space and finding diverse complete solutions compared to the other two methods. Exceptions occur primarily for $\theta_d = 1.5$, where *CoCoMEGA* performs comparably to other methods. A possible explanation for this behavior is that a higher θ_d reduces the likelihood of forming complete solutions, thus limiting the advantage of *CoCoMEGA* in this specific setting.

The fitness threshold θ_f has a notable impact on the *DS* values across methods. However, the pattern of trends remains similar across all θ_f values. In general, with the increasing of θ_f, *DS* values drop for all methods, reflecting the difficulty in finding severe violations regardless of distance constraints.

The distance threshold θ_d also influences the efficiency of each method. At a low $\theta_d = 0.5$, *CoCoMEGA* exhibits a clear advantage over *SGA* and *RS*, almost consistently achieving higher *DS* values. This suggests that *CoCoMEGA* is highly effective in identifying complete solutions that are distinct. As θ_d increases to 1.0, *CoCoMEGA* maintains its lead above a 70% search budget, though the gap with *SGA* and *RS* starts

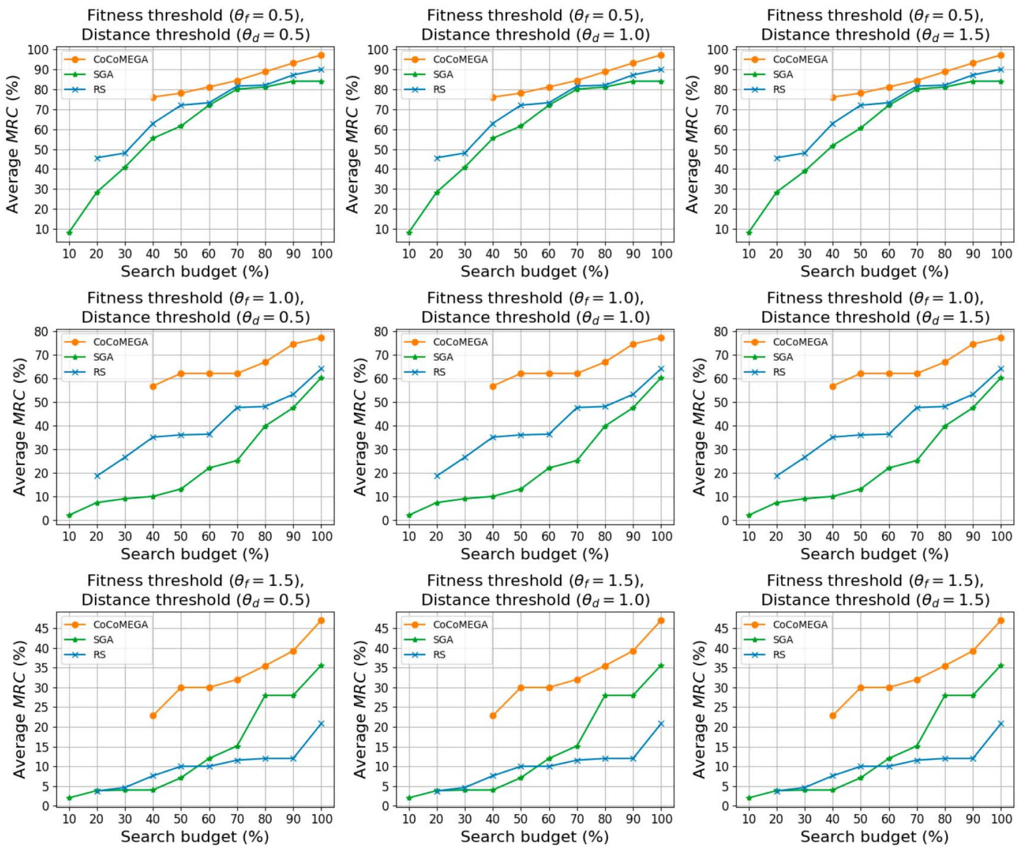

Fig. 7. *Metamorphic Relation Coverage (MRC)* vs. search budget. *MRC* indicates the percentage of defined MRs that are violated by at least one discovered solution. Higher *MRC* means the method finds solutions that violate more of the predefined MRs, indicating thorough behavioral exploration. The x-axis represents how much of the simulation budget has been consumed. This figure shows how quickly each method covers all relevant MR violations as the search progresses.

to narrow slightly. At the highest $\theta_d = 1.5$, the gap between *CoCoMEGA* and the baseline methods diminishes further. *CoCoMEGA* ultimately identifies more *DS* than the baseline methods, except when $\theta_f = 0.5$, where *RS* outperforms the other two methods. This suggests that *RS* is particularly effective at finding a greater number of diverse but trivial violations. For $\theta_f \geqslant 1.0$ and $\theta_d = 1.5$, all three methods display similarly low *DS* values, suggesting that a combination of high fitness and large distance requirements severely limits the number of viable complete solutions. In these cases, *RS* occasionally achieves *DS* values comparable to *CoCoMEGA*, likely due to its simpler approach to exploring the search space.

At higher search budgets (70–100%), *CoCoMEGA* achieves the highest *DS* values across most configurations. However, its *DS* values are sometimes similar to or slightly lower than those of the baseline methods. This may be due to the initial overhead of *CoCoMEGA*, as it exhaustively simulates potential complete solutions by combining scenarios and perturbations in the first generation, completing only one search generation while *SGA* completes seven or more. Nonetheless, *CoCoMEGA* continues to find new, distinct complete solutions as the budget increases. Although the maximum search budget was capped at 200 due to computational limits, *CoCoMEGA* shows a faster growth in *DS* values than the baseline methods, suggesting it would find even more complete solutions with a larger budget.

Fig. 7 illustrates how *MRC* evolves as the percentage of search budget consumed increases. Across all threshold settings, *CoCoMEGA* significantly and consistently outperforms *SGA* and *RS* in terms of *MRC*, indicating that *CoCoMEGA* can identify complete solutions covering more MRs than the other two baseline methods. Furthermore, when $\theta_f = 0.5$, the *MRC* of *CoCoMEGA* approaches nearly 100%. While *MRC* declines as θ_f increases, θ_d has minimal impact on *MRC*. At the highest $\theta_f = 1.5$, *CoCoMEGA* further widens the gap over *SGA* and *RS*, demonstrating its capability to find severe violations that cover a broader range of MRs.

To quantitatively compare the efficiency of *CoCoMEGA*, *SGA* and *RS* algorithms, we employed the *Area Under the Curve (AUC)* measure, a widely accepted and arguably the most useful performance aggregation method in optimization research [111]. Specifically, we computed the area under each algorithm's performance curve over the entire budget range (0%–100%) separately for each of the two metrics, i.e., *DS* over budget (Fig. 6) and *MRC* over budget (Fig. 7). This area can be interpreted as an average performance over the budget range and thus provides a comprehensive efficiency measure across all budget levels into a single value. To compare *CoCoMEGA* with the baselines, we calculated the average differences in their *AUC* values across different threshold configurations, and

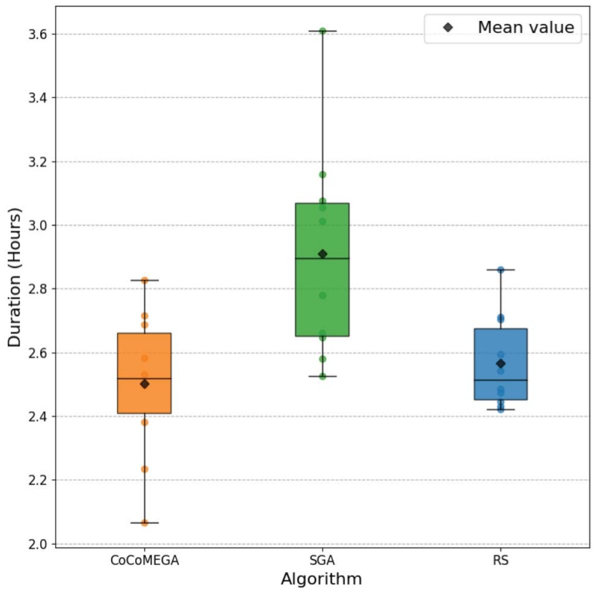

Fig. 8. Comparison of computational efficiency (execution time) for *CoCoMEGA, SGA,* and *RS*. Diamonds indicate mean execution times. Lower execution times indicate higher computational efficiency.

assessed the statistical significance of these improvements using Wilcoxon signed-rank tests.

For the *DS* over budget curves, *CoCoMEGA* achieved an average *AUC* improvement of 46.13% (p-value ≈ 0.023) compared to *SGA* and 31.95% (p-value ≈ 0.037) compared to *RS*, across moderate and high fitness thresholds ($\theta_f \geq 1$). Further, for the *MRC* over budget curves, the efficiency gains of *CoCoMEGA* were even more substantial, with average *AUC* improvements of 90.07% (p-value ≈ 0.036) over *SGA* and 85.86% (p-value ≈ 0.035) over *RS*. These results confirm that *CoCoMEGA* consistently outperforms *SGA* and *RS* in terms of overall efficiency.

Following the analysis of the *DS* and *MRC* metrics, we also compared the computational efficiency of *CoCoMEGA*, *SGA*, and *RS* directly in terms of execution time, within a fixed search budget. Fig. 8 illustrates the distribution of execution times (in hours) for each algorithm across 10 experimental runs. As depicted in Fig. 8, *CoCoMEGA* demonstrates superior computational efficiency with an average execution time of approximately 2.50 hours, compared to *RS* with 2.57 hours and *SGA* with 2.91 hours. In particular, *CoCoMEGA* consistently achieves faster execution times than *SGA*, indicating greater overall efficiency. The longer execution time observed in *SGA* is likely due to the generation of more invalid complete solutions throughout its search process. Invalid solutions are solutions whose parameters violate simulation constraints, such as overlapping object locations, causing the simulator to fail at the beginning of the simulation. While these invalid solutions do not count against the simulation budget and will then be ignored in the evolutionary process, the attempt to simulate them incurs additional computational overhead, thus contributing to increased overall execution time.

In summary, the results indicate that *CoCoMEGA* is particularly well-suited for tasks requiring diverse, severe violations across a range of search budgets and thresholds. The *AUC* analysis further reinforces this observation, showing a clear efficiency advantage for *CoCoMEGA* in terms of both *DS* and *MRC* metrics. Both *SGA* and *RS* struggle to maintain competitive *DS* values, especially at higher θ_f, where the search space becomes more constrained. *RS* occasionally performs comparably to *CoCoMEGA* at high θ_d and low θ_f, which is likely due to its pure exploratory nature.

Answer to RQ2

CoCoMEGA is more efficient than *SGA* and *RS* in identifying diverse, severe violations across a range of search budgets and thresholds, achieving an average *AUC* improvement of 46% over *SGA* and 32% over *RS* for the *DS* metric, and 90% over *SGA* and 86% over *RS* for the *MRC* metric. The performance advantage of *CoCoMEGA* persists across various threshold settings, with its lead slightly diminishing only at the highest distance thresholds (enforcing very high diversity) or under low-fitness thresholds.

E. RQ3: Effectiveness of Diversity Mechanisms in CoCoMEGA

1) Methodology: To answer RQ3, we evaluate the impact of the diversity mechanisms in *CoCoMEGA* by comparing the performance of the full method with a variant that excludes the diversity mechanisms, i.e., fitness clearing and diversity optimization for population archives. The variant, denoted as *CoCoMEGA*$\backslash d$, is identical to *CoCoMEGA* in all aspects except that it does not include the diversity mechanisms. We compare the two methods in terms of *DS* and *SD* across different fitness and distance thresholds. The methodology mirrors that of RQ1, using identical hyperparameters and performing 10 repetitions for each method, with the only difference being the presence of diversity mechanisms.

2) Results: Fig. 9 compares the *DS* values of *CoCoMEGA* with and without diversity mechanisms across various fitness and distance thresholds. The results indicate that *CoCoMEGA* consistently achieves higher *DS* values than *CoCoMEGA*$\backslash d$, highlighting the significant contribution of diversity mechanisms to identifying distinct solutions. This performance gap remains consistent across all fitness and distance thresholds. While both methods experience a decrease in *DS* values as θ_d increases, *CoCoMEGA* maintains its superiority, underscoring the effectiveness of its diversity mechanisms in preserving solution diversity. These findings suggest that diversity mechanisms are crucial for promoting a broadly distributed solution set and maintaining an expansive search space.

Fig. 10 depicts the *APD* between solutions across different fitness and distance thresholds. *CoCoMEGA* consistently produces solutions with higher *APD* compared to its counterpart without diversity, indicating a broader exploration of the

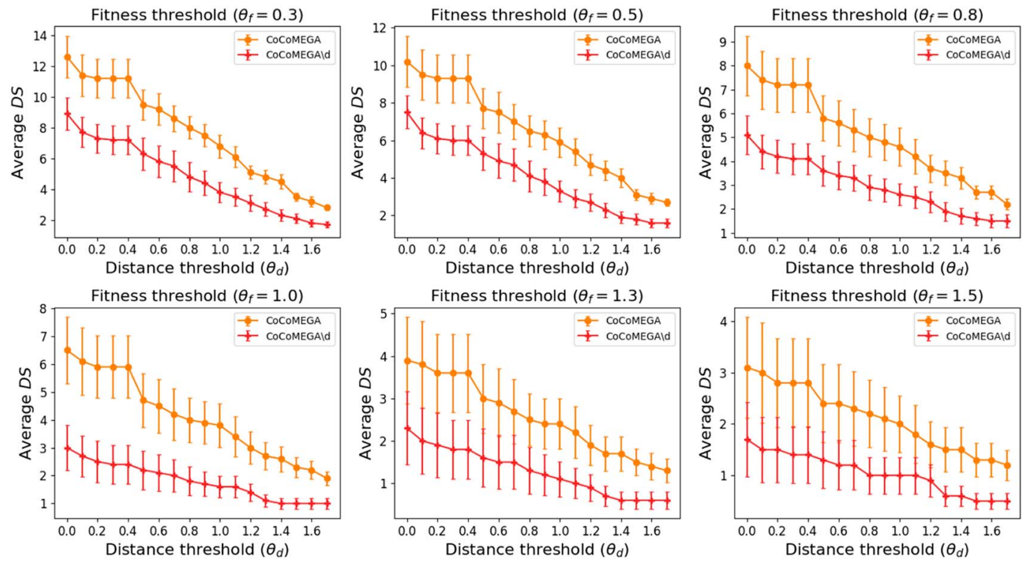

Fig. 9. Comparison of *CoCoMEGA* with and without diversity mechanisms: *Distinct Solutions (DS)* vs. distance threshold (θ_d) across different fitness thresholds (θ_f). A higher *DS* indicates more distinct violating solutions discovered by the method. Each curve plots the average *DS* at different θ_d settings, under a specific fitness threshold θ_f. This figure reveals how diversity mechanisms impact the quantity and diversity of solutions as θ_f varies.

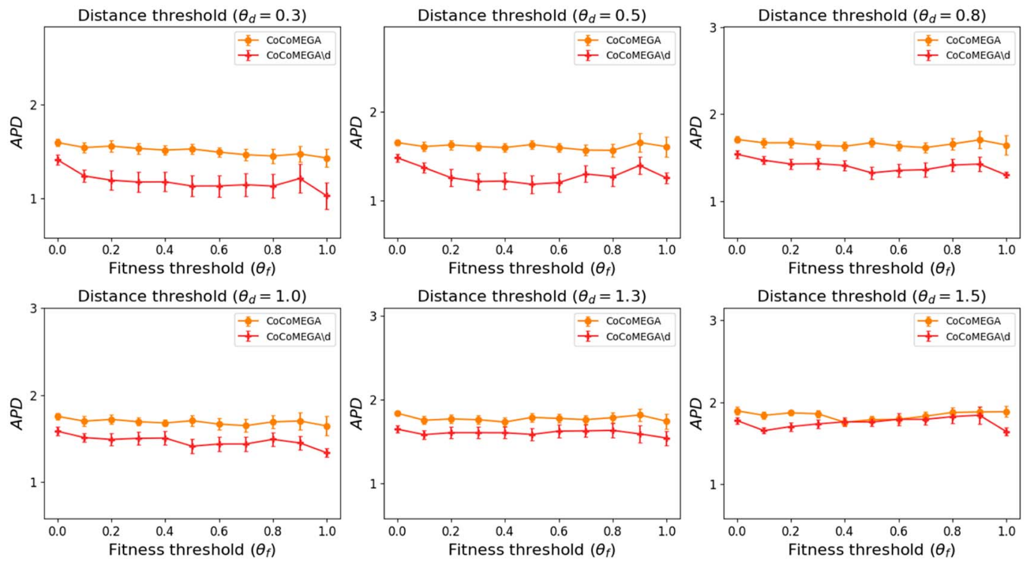

Fig. 10. Comparison of *CoCoMEGA* with and without diversity mechanisms: *Average Pairwise Distance (APD)* vs. fitness threshold (θ_f) across different distance thresholds (θ_d). *APD* quantifies how spread out the solutions are. A higher *APD* indicates greater diversity among the found solutions. Each subplot corresponds to a certain distance threshold θ_d, and the y-axis shows the *APD* of discovered solutions under various θ_f values. This figure demonstrates how diversity mechanisms maintain diversity while striving for higher fitness (i.e., more severe violations).

solution space and a more distributed solution set. The difference is especially significant at lower θ_d values, where *CoCoMEGA*$\backslash d$ produces solutions with lower *APD*, suggesting solutions that are more clustered together in the search space. This confirms that the diversity mechanisms contribute to exploring more varied solutions.

Table III presents the *MRC* and *CMR* values for both *CoCoMEGA* and *CoCoMEGA*$\backslash d$. *CoCoMEGA* consistently achieves higher *MRC* across various θ_f values than *CoCoMEGA*$\backslash d$. The absence of diversity mechanisms results in a even larger decrease in *MRC* at higher θ_f values, highlighting their importance in preserving solution diversity. Similarly, *CMR* values are notably higher with diversity enforcement. These findings confirm the effectiveness of diversity mechanisms in improving both solution coverage and variability.

TABLE III
MRC and CMR Values for CoCoMEGA with and Without Diversity Mechanisms at Different Values of θ_f and θ_d

θ_d	Method	Average *MRC*$\pm CI_{0.95}$ (Average *CMR*$\pm CI_{0.95}$)					
		$\theta_f = 0.3$	$\theta_f = 0.5$	$\theta_f = 0.8$	$\theta_f = 1.0$	$\theta_f = 1.3$	$\theta_f = 1.5$
$\theta_d = 0.0$	*CoCoMEGA*	**100.0 $\pm$ 0.0** (3.0 $\pm$ 0.3)	**98.0 $\pm$ 1.8** (2.6 $\pm$ 0.3)	**90.0 $\pm$ 7.2** (2.1 $\pm$ 0.3)	**78.0 $\pm$ 10.9** (1.8 $\pm$ 0.4)	**66.0 $\pm$ 12.3** (1.3 $\pm$ 0.3)	**56.0 $\pm$ 13.1** (1.0 $\pm$ 0.3)
	CoCoMEGA$\setminus d$	86.0 $\pm$ 7.6 (2.9 $\pm$ 0.4)	86.0 $\pm$ 7.6 (2.5 $\pm$ 0.4)	82.0 $\pm$ 9.1 (2.2 $\pm$ 0.3)	56.0 $\pm$ 12.0 (1.4 $\pm$ 0.4)	34.0 $\pm$ 11.7 (1.1 $\pm$ 0.4)	34.0 $\pm$ 11.7 (0.9 $\pm$ 0.3)
$\theta_d = 1.0$	*CoCoMEGA*	**100.0 $\pm$ 0.0** (2.9 $\pm$ 0.3)	**98.0 $\pm$ 1.8** (2.6 $\pm$ 0.3)	**90.0 $\pm$ 7.2** (2.1 $\pm$ 0.3)	**78.0 $\pm$ 10.9** (1.8 $\pm$ 0.4)	**66.0 $\pm$ 12.3** (1.3 $\pm$ 0.3)	**56.0 $\pm$ 13.1** (1.0 $\pm$ 0.3)
	CoCoMEGA$\setminus d$	84.0 $\pm$ 8.4 (2.3 $\pm$ 0.3)	84.0 $\pm$ 8.4 (2.1 $\pm$ 0.3)	80.0 $\pm$ 9.7 (1.9 $\pm$ 0.3)	54.0 $\pm$ 12.0 (1.2 $\pm$ 0.3)	32.0 $\pm$ 11.5 (0.9 $\pm$ 0.3)	32.0 $\pm$ 11.5 (0.8 $\pm$ 0.3)
$\theta_d = 1.7$	*CoCoMEGA*	**98.0 $\pm$ 1.8** (2.0 $\pm$ 0.3)	**94.0 $\pm$ 3.8** (2.1 $\pm$ 0.2)	**86.0 $\pm$ 7.6** (1.6 $\pm$ 0.1)	**78.0 $\pm$ 10.9** (1.3 $\pm$ 0.2)	**64.0 $\pm$ 12.0** (1.1 $\pm$ 0.2)	**56.0 $\pm$ 13.1** (1.0 $\pm$ 0.3)
	CoCoMEGA$\setminus d$	66.0 $\pm$ 9.3 (1.5 $\pm$ 0.2)	70.0 $\pm$ 9.0 (1.4 $\pm$ 0.1)	68.0 $\pm$ 10.1 (1.2 $\pm$ 0.2)	50.0 $\pm$ 11.8 (0.9 $\pm$ 0.2)	28.0 $\pm$ 10.5 (0.6 $\pm$ 0.2)	28.0 $\pm$ 10.5 (0.5 $\pm$ 0.2)

Answer to RQ3

Diversity mechanisms play a crucial role in enhancing solution diversity and exploration. *CoCoMEGA* with diversity mechanisms consistently outperforms its counterpart without such mechanisms, identifying 91% more MR violations in terms of *DS* across various fitness and distance thresholds. Additionally, these mechanisms improve *MRC* and *CMR* values by 50% and 33%, respectively, ensuring broader solution coverage and variability.

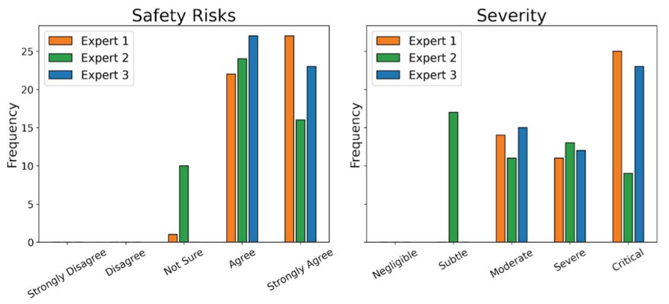

Fig. 11. Combined histograms of expert ratings for the 50 test cases: Left subplot shows responses to "This test case entail safety risks" and the right subplot shows responses to "How severe is the issue in terms of its potential impact on driving safety?" Each bar color represents a different expert (orange for Expert 1, green for Expert 2, and blue for Expert 3).

VI. Discussion

A. Developer Feedback

To complement our automated evaluation, we sought feedback from three domain experts with extensive safety engineering experience in ADSs. Our objective was to determine whether the detected MR violations represented safety risks from the experts' perspective, and if so, to what extent. Informed consent was obtained from the domain experts for the use of their feedback in this study.

We supplied the experts with 50 test cases representing the most severe violations found by CoCoMEGA across 10 executions. For each test case, we provided source and follow-up videos along with velocity-change diagrams, highlighting where the violations occurred. We then asked two questions to be answered on a Likert scale [112]: (i) "This test case entail safety risks" (Rate from 1 = Strongly Disagree to 5 = Strongly Agree), and (ii) "How severe is the issue in terms of its potential impact on driving safety?" (Rate from 1 = Negligible to 5 = Critical).

As seen in Fig. 11, experts tended to rank the test cases on the higher end of both scales. For the question on safety risks (left subplot), a large proportion of ratings fell into "Agree" or "Strongly Agree" (i.e., 4 or 5), with average scores of 4.52, 4.12, and 4.46 for expert 1, 2, and 3, respectively. This suggests that most of the MR-violating scenarios were perceived as posing genuine safety concerns. Similarly, for severity (right subplot), many violations were deemed "Severe" or "Critical" (i.e., 4 or 5), with average scores of 4.22, 3.28, and 4.16, respectively, underlining that these issues could have a substantial impact on driving safety. These expert assessments confirm that the test cases generated by our approach are indeed viewed by practitioners as indicative of noteworthy safety risks, thereby underscoring the practical importance of the discovered violations.

B. Interpretation of Solution Diversity

The metrics of *APD*, *MRC*, and *CMR* provide complementary perspectives on the diversity of identified complete solutions, together forming a robust framework for assessing diversity in the test cases generated for ADSs.

The *APD* metric, representing the average distance between pairs of complete solutions, provides a straightforward quantification of diversity in the scenario space. Higher *APD* values indicate that complete solutions are spread out and capture a broader array of possible scenarios. This metric is particularly useful for understanding the variation in complete solutions regarding scenario representation, as it directly reflects how far solutions diverge from one another. However, *APD* has limitations. For instance, methods like *RS* may yield a few widely spaced solutions, inflating *APD* despite limited diversity. Thus, *APD* alone may not fully capture diversity when solutions are few and highly dispersed.

The *MRC* and *CMR* metrics, on the other hand, measure the percentage of predefined MRs covered by the complete solutions and the unique combinations of MRs that the solutions violate, providing different aspects of diversity by assessing how comprehensively the solutions explore the possible ways in which system behaviors can lead to MR violations and examining how solutions interact with multiple MRs simultaneously. Unlike *MRC*, which is concerned with the total coverage of individual MRs, *CMR* captures further the complexity and richness of the testing scenarios, indicating that complete solutions cover not only isolated violations but also complex, interrelated ones. This capability is crucial for understanding potential compounded risks in ADSs that may emerge when multiple MRs are violated simultaneously. High values in *MRC* or *CMR* reflect a robust exploration of critical behaviors, essential for evaluating system resilience under more nuanced and complex conditions.

In summary, together, *APD*, *MRC*, and *CMR* offer a multi-faceted approach to evaluating solution diversity. *APD* provides a measure of structural diversity, while *MRC* and *CMR* focus on behavioral diversity, reflecting how comprehensively the complete solutions challenge the system's response to diverse conditions and potential violations. This combined approach ensures that the test cases not only cover a wide range of possible scenarios but also uncover complex, high-risk scenarios, which are critical for a comprehensive assessment of ADSs.

C. Threats to Validity

In this section, we analyze possible threats to validity of our results. These threats are grouped into four categories: Internal, External, Construct, and Conclusion [113].

1) Internal validity: Internal validity refers to the degree to which a study reliably demonstrates a cause-and-effect relationship between the experiment and its outcome. In our research, one of the primary threats to internal validity is the sensitivity of population-based methods (i.e., *CoCoMEGA* and *SGA*) to hyperparameter values, as the current settings may not reflect optimal configurations. To address this, we adopted widely recommended hyperparameter values [110], which are frequently employed in the literature. For parameters without recommended values, we determined them through a pilot experiment.

Another internal validity threat concerns the correctness of the MRs used in the experiment, as they play a central role in identifying MR violations. Any incorrect or overly restrictive MRs could lead to false positives or negatives, potentially affecting the reliability of test results. For instance, an inaccurate threshold in an output relation could either fail to detect a true violation or detect one where there is not. To mitigate this, we selected MRs that are well-established in the literature and validated them through a pilot experiment before full implementation in experiments. This phase allowed us to identify and analyze discrepancies or unexpected results, refining the MRs as needed, e.g., thresholds in output relations were adjusted. This process ensured the MRs were well-calibrated and reliable for the extensive experiments.

A further threat to internal validity arises from minor discrepancies between the actual number of executed simulations and the allocated simulation budget for the population-based methods. Ensuring consistent budgets across different methods is essential for a fair comparison and we addressed this issue by using linear interpolation to align simulation budgets. However, linear interpolation introduces its own risks, as it assumes that the relationship between the search budget and metrics is linear and continuous. This assumption may not hold in cases where metrics exhibit non-linear or abrupt changes across budgets, potentially introducing inaccuracies in the aligned results. To mitigate this threat, we validated the interpolation approach by examining trends in metrics and confirming that the linear assumption holds within the tested budget range.

2) External validity: External validity addresses the extent to which the results can be generalized to various contexts and examines the gap between the simulation environment and real-world conditions. In our research, the primary concern regarding external validity was the potential limitations arising from the use of a specific ADS (INTERFUSER) and simulator (CARLA). However, CARLA is highly regarded and widely used open-source simulator known for its high fidelity, and INTERFUSER was a top-performing ADS on the CARLA leaderboard [102] during our evaluation period. Moreover, the considerable implementation overhead required to enable scenario execution with CARLA and INTERFUSER significantly constrained our ability to evaluate additional technologies. As part of our future research, we intend to extend our approach to other ADSs and explore its applicability to domains such as autonomous mobile robotic systems and unmanned aerial systems. This will contribute to a more comprehensive understanding of the effectiveness and limitations of our methodology in various real-world contexts.

Another potential threat to the external validity of our study lies in the limited variety of MRs we considered, as we focused only on two types of common output relations, i.e., *invariance* and *decreasing* output relations. These MRs are not representative of more complex (e.g., nonlinear or non-monotonic) relationships, thus potentially restricting the generalizability of our findings, as other types of input and output relations exist that could uncover different types of MR violations. However, we selected the most widely used MRs in system-level testing of ADSs, from the literature, thus capturing common types of system behaviors relevant to ADSs. Expanding the study to include a broader variety of MRs, which remain to be defined, could provide a more comprehensive assessment and further validate the robustness of our approach across diverse testing scenarios.

3) Construct validity: Construct validity addresses the extent to which the object of study truly reflects the underlying theory behind it. In our research, a key concern related to construct validity is the suitability of the *DS* and *SD* metrics for representing the effectiveness of each method in identifying severe MR violations and exploring diverse regions of the search space. While these metrics are chosen to capture the concept of effectiveness, they may not fully account for all dimensions of solution quality and diversity.

4) Conclusion validity: Conclusion validity refers to the extent to which a research conclusion could be trusted. In our experiments, we encountered limitations due to the relatively small number of repetitions due to enormous computation costs, i.e., 10 runs per method were conducted. To address the potential impact of statistical error arising from this limited sample size, we have systematically reported descriptive statistics in conjunction with their corresponding confidence intervals, providing insights into the central tendency and variations of results.

VII. Conclusion

In this paper, we introduce *CoCoMEGA*, a novel automated testing method that effectively combines, for the first time, Metamorphic Testing (MT) and advanced search-based testing to support the system-level assessments of ADSs. By integrating MT, *CoCoMEGA* effectively detects test cases that violate Metamorphic Relations (MRs), capturing desirable properties expected to hold in ADSs, even when safety requirements are not violated. This helps uncover subtle safety risks. Our evaluation on the Carla simulator using the Interfuser ADS demonstrated that *CoCoMEGA* outperforms baseline methods across various budget and threshold configurations, efficiently generating severe and diverse MR violations that explore wider regions of the search space. Expert assessments of these violations confirmed that most pose real safety risks, with many deemed severe or critical, underscoring their practical relevance to improving ADS safety. These results suggest *CoCoMEGA* is an effective and efficient solution to the challenges of ADS testing by decomposing the high-dimensional search space using a Cooperative Co-Evolutionary Algorithm (CCEA), thus expanding the test coverage of potential safety risks. Future work could extend this approach to additional simulators, broaden its scope to other complex autonomous systems, and further explore strategies for improved testing efficiency such as surrogate modeling.

Data Availability

The replication package for our experiments, which includes the implementation of our approach, baseline methods, configuration files, and experimental results, is publicly available on Figshare at https://doi.org/10.6084/m9.figshare.27910677.

References

[1] Z. Tahir and R. Alexander, "Coverage based testing for V&V and safety assurance of self-driving autonomous vehicles: A systematic literature review," in *Proc. IEEE Int. Nat. Conf. Artif. Intell. Testing (AITest)*, Piscataway, NJ, USA: IEEE Press, Aug. 2020, pp. 23–30, doi: 10.1109/AITEST49225.2020.00011.

[2] Y. Tian, K. Pei, S. Jana, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proc. 40th Int. Conf. Softw. Eng. (ICSE)*, New York, NY, USA: ACM, May 2018, pp. 303–314, doi: 10.1145/3180155.3180220.

[3] M. Bojarski, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.

[4] G. Lou, Y. Deng, X. Zheng, M. Zhang, and T. Zhang, "Testing of autonomous driving systems: where are we and where should we go?" in *Proc. 30th ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, New York, NY, USA: ACM, Nov. 2022, pp. 31–43, doi: 10.1145/3540250.3549111.

[5] A. Nasr, K. Inkol, and J. McPhee, "Safety in wearable robotic exoskeletons: Design, control, and testing guidelines," *J. Mechan. Robot.*, vol. 17, no. 5, p. 050801, Nov. 2024, doi: 10.1115/1.4066900.

[6] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Jun. 2018, pp. 586–597, doi: 10.1109/DSN.2018.00066.

[7] "List of self-driving car fatalities," Accessed: Dec. 04, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Self-driving_car#Incidents

[8] Carsurance. "24 self-driving car statistics & facts." 2022. Accessed: Dec. 04, 2024. [Online]. Available: https://carsurance.net/insights/self-driving-car-statistics/

[9] J. Garcia, Y. Feng, J. Shen, S. Almanee, Y. Xia, and A. Q. Chen, "A comprehensive study of autonomous vehicle bugs," in *Proc. ACM/IEEE 42nd Int. Conf. Softw. Eng. (ICSE)*, New York, NY, USA: ACM, Jun. 2020, pp. 385–396, doi: 10.1145/3377811.3380397.

[10] J. D. Musa, *Software Reliability Engineering: More Reliable Software Faster and Cheaper.* Bloomington, IN: Authorhouse, 2004.

[11] S. Tang et al., "A survey on automated driving system testing: Landscapes and trends," *ACM Trans. Softw. Eng. Method.*, vol. 32, no. 5, pp. 1–62, Jul. 2023, doi: 10.1145/3579642.

[12] M. Cheng, Y. Zhou, X. Xie, J. Wang, G. Meng, and K. Yang, "Evaluating decision optimality of autonomous driving via metamorphic testing," 2024, *arXiv:2402.18393*.

[13] A. Corso and M. J. Kochenderfer, "Interpretable safety validation for autonomous vehicles," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC),* Sep. 2020, pp. 1–6, doi: 10.1109/ITSC45102.2020.9294490.

[14] C. E. Tuncali and G. Fainekos, "Rapidly-exploring random trees for testing automated vehicles," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Piscataway, NJ, USA: IEEE Press, Oct. 2019, pp. 661–666, doi: 10.1109/ITSC.2019.8917375.

[15] H. Wakabayashi, Y. Takahashi, S. Niimi, and K. Renge, "Traffic conflict analysis using vehicle tracking system/digital VCR and proposal of a new conflict indicator," *INFRAStruct. Planning Rev.*, vol. 20, pp. 949–956, Sep. 2003, doi: 10.2208/journalip.20.949.

[16] A. Tamke, T. Dang, and G. Breuel, "A flexible method for criticality assessment in driver assistance systems," in *Proc. IEEE Intell. Vehicles Symp. (IV),* Piscataway, NJ, USA: IEEE Press, Jun. 2011, pp. 697–702, doi: 10.1109/IVS.2011.5940482.

[17] C. Birchler, S. Khatiri, P. Rani, T. Kehrer, and S. Panichella, "A roadmap for simulation-based testing of autonomous cyber-physical systems: Challenges and future direction," 2024, *arXiv:2405.01064*.

[18] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE Int. J. Transp. Saf.*, vol. 4, no. 1, pp. 15–24, Apr. 2016. [Online]. Available: http://www.jstor.org/stable/26167741

[19] F. U. Haq, D. Shin, and L. Briand, "Efficient online testing for DNN-enabled systems using surrogate-assisted and many-objective optimization," in *Proc. 44th Int. Conf. Softw. Eng., Ser. ICSE '22.* New York, NY, USA: ACM, May 2022, pp. 811–822, doi: 10.1145/3510003.3510188.

[20] Z. Zhong, G. Kaiser, and B. Ray, "Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles," *IEEE Trans. Softw. Eng.*, vol. 49, no. 4, pp. 1860–1875, Apr. 2023, doi: 10.1109/TSE.2022.3195640.

[21] N. Rajabli, F. Flammini, R. Nardone, and V. Vittorini, "Software verification and validation of safe autonomous cars: A systematic literature review," *IEEE Access*, vol. 9, pp. 4797–4819, 2021, doi: 10.1109/ACCESS.2020.3048047.

[22] R. Stefan, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87456–87477, May 2020, doi: 10.1109/ACCESS.2020.2993730.

[23] M. A. Potter and K. A. Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving from Nature — PPSN III*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 249–257, doi: 10.1007/3-540-58484-6_269.

[24] ISO. "ISO 21448:2022, road vehicles–safety of the intended functionality." Accessed: Dec. 04, 2024. [Online]. Available: https://www.iso.org/standard/77490.html

[25] A. for Standardization of Automation and M. S. (ASAM). ASAM OpenSCENARIO. Accessed: Dec. 04, 2024. [Online]. Available: https://www.asam.net/standards/detail/openscenario/v200/

[26] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *Proc. 31st IEEE/ACM Int. Conf. Automat. Softw.*

Eng. (ASE), New York, NY, USA: ACM, Aug. 2016, pp. 63–74, doi: 10.1145/2970276.2970311.

[27] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, “Testing autonomous cars for feature interaction failures using many-objective search,” in *Proc. 33rd ACM/IEEE Int. Conf. Automated Softw. Eng. (ASE)*, New York, NY, USA: ACM, Sep. 2018, pp. 143–154, doi: 10.1145/3238147.3238192.

[28] T. Dreossi, A. Donzé, and S. A. Seshia, “Compositional falsification of cyber-physical systems with machine learning components,” *J. Automat. Reason.*, vol. 63, no. 4, pp. 1031–1053, Jan. 2019, doi: 10.1007/s10817-018-09509-5.

[29] G. Li et al., “AV-FUZZER: Finding safety violations in autonomous driving systems,” in *Proc. IEEE 31st Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2020, pp. 25–36, doi: 10.1109/ISSRE5003.2020.00012.

[30] “Baidu.” Apollo. Accessed: Dec. 04, 2024. [Online]. Available: https://en.apollo.auto/apollo-self-driving

[31] N. Kolb, F. Hauer, and A. Pretschner, “Fitness function templates for testing automated and autonomous driving systems in intersection scenarios,” in *Proc. IEEE Int. Intell. Transp. Syst. Conf.*, Sep. 2021, pp. 217–222, doi: 10.1109/ITSC48978.2021.9564591.

[32] F. Hauer, A. Pretschner, and B. Holzmüller, “Fitness functions for testing automated and autonomous driving systems,” *Comput. Saf., Rel., Secur.*, New York, NY, USA: Springer Int. Publishing, Aug. 2019, pp. 69–84, doi: 10.1007/978-3-030-26601-1_5.

[33] Y. Luo et al., “Targeting requirements violations of autonomous driving systems by dynamic evolutionary search,” in *Proc. 36th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE),* Nov. 2021, pp. 279–291, doi: 10.1109/ASE51524.2021.9678883.

[34] C. Birchler, N. Ganz, S. Khatiri, A. Gambi, and S. Panichella, “Cost-effective simulation-based test selection in self-driving cars software,” *Sci. Comput. Program.*, vol. 226, Mar. 2023, Art. no. 102926. doi: 10.1016/j.scico.2023.102926.

[35] C. Birchler, S. Khatiri, P. Derakhshanfar, S. Panichella, and A. Panichella, “Single and multi-objective test cases prioritization for self-driving cars in virtual environments,” *ACM Trans. Softw. Eng. Method.*, vol. 32, no. 2, pp. 1–30, Apr. 2023, doi: 10.1145/3533818.

[36] V. Riccio and P. Tonella, “Model-based exploration of the frontier of behaviours for deep learning system testing,” in *Proc. 28th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ES-EC/FSE)*, New York, NY, USA: ACM, Nov. 2020, pp. 876–888, doi: 10.1145/3368089.3409730.

[37] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, “DeepHyperion: Exploring the feature space of deep learning-based systems through illumination search,” in *Proc. 30th ACM SIGSOFT Int. Symp. Softw. Testing Anal. (ISSTA)*, New York, NY, USA: ACM, Jul. 2021, pp. 79–90, doi: 10.1145/3460319.3464811.

[38] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, “Efficient and effective feature space exploration for testing deep learning systems,” *ACM Trans. Softw. Eng. Method.*, vol. 32, no. 2, pp. 1–38, Mar. 2023, doi: 10.1145/3544792.

[39] J. Mouret and J. Clune, “Illuminating search spaces by mapping elites,” 2015, *arXiv:1504.04909*.

[40] E. Castellano, A. Cetinkaya, and P. Arcaini, “Analysis of road representations in search-based testing of autonomous driving systems,” in *Proc. IEEE 21st Int. Conf. Softw. Qual., Rel. Secur. (QRS)*, Piscataway, NJ, USA: IEEE Press, Dec. 2021, pp. 167–178, doi: 10.1109/QRS54544.2021.00028.

[41] A. Gambi, M. Mueller, and G. Fraser, “Automatically testing self-driving cars with search-based procedural content generation,” in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Testing Anal. (ISSTA)*, vol. 4, New York, NY, USA: ACM, Jul. 2019, pp. 318–328, doi: 10.1145/3293882.3330566.

[42] Q. Goss and M. I. Akbaş, “Eagle strategy with local search for scenario based validation of autonomous vehicles,” in *Proc. Int. Conf. Connected Vehicle Expo (ICCVE)*, Piscataway, NJ, USA: IEEE Press, Mar. 2022, pp. 1–6, doi: 10.1109/ICCVE52871.2022.9743067.

[43] X. Zheng, H. Liang, B. Yu, B. Li, S. Wang, and Z. Chen, “Rapid generation of challenging simulation scenarios for autonomous vehicles based on adversarial test,” in *Proc. IEEE Int. Conf. Mechatron. Automat. (ICMA),* Piscataway, NJ, USA: IEEE Press, Oct. 2020, pp. 1166–1172, doi: 10.1109/ICMA49215.2020.9233535.

[44] F. Batsch, A. Daneshkhah, V. Palade, and M. Cheah, “Scenario optimisation and sensitivity analysis for safe automated driving using Gaussian processes,” *Appl. Sci.*, vol. 11, no. 2, Jan. 2021, Art. no. 775, doi: 10.3390/app11020775.

[45] H. Beglerovic, M. Stolz, and M. Horn, “Testing of autonomous vehicles using surrogate models and stochastic optimization,” in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–6, doi: 10.1109/ITSC.2017.8317768.

[46] J. Sun, H. Zhou, H. Zhang, Y. Tian, and Q. Ji, “Adaptive design of experiments for accelerated safety evaluation of automated vehicles,” in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC),* Sep. 2020, pp. 1–7, doi: 10.1109/ITSC45102.2020.9294429.

[47] T. Y. Chen et al., “Metamorphic testing: A review of challenges and opportunities,” *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–27, Jan. 2018, doi: 10.1145/3143561.

[48] T. Y. Chen, S. Cheung, and S. Yiu, “Metamorphic testing: A new approach for generating next test cases,” Feb. 2020, *arXiv:2002.12543*.

[49] Z. Yang, S. Huang, C. Zheng, X. Wang, Y. Wang, and C. Xia, “MetaLiDAR: Automated metamorphic testing of lidar-based autonomous driving systems,” *J. Softw.: Evol. Process.*, vol. 36, no. 7, Dec. 2023, Art. no. e2644. doi: 10.1002/smr.2644.

[50] Z. Yang et al., “MetaSem: Metamorphic testing based on semantic information of autonomous driving scenes,” *Softw. Testing, Verification Rel.*, vol. 34, no. 5, May 2024, Art. no. e1878, doi: 10.1002/stvr.1878.

[51] Z. Q. Zhou and L. Sun, “Metamorphic testing of driverless cars,” *Commun. ACM*, vol. 62, no. 3, pp. 61–67, 2019.

[52] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, “DeepRoad: Gan-based metamorphic testing and input validation framework for autonomous driving systems,” in *Proc. 33rd ACM/IEEE Int. Conf. Automat. Softw. Eng. (ASE)*, New York, NY, USA: ACM, Sep. 2018, pp. 132–142, doi: 10.1145/3238147.3238187.

[53] Y. Pan, H. Ao, and Y. Fan, “Metamorphic testing for autonomous driving systems in fog based on quantitative measurement,” in *Proc. IEEE 21st Int. Conf. Softw. Qual., Rel. Secur. Companion, (QRS-C),* Piscataway, NJ, USA: IEEE Press, Dec. 2021, pp. 30–37, doi: 10.1109/QRS-C55045.2021.00015.

[54] J. C. Han and Z. Q. Zhou, “Metamorphic fuzz testing of autonomous vehicles,” in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. Workshops (ICSEW)*, New York, NY, USA: ACM, Jun. 2020, pp. 380–385, doi: 10.1145/3387940.3392252.

[55] R. Li et al., “Metamorphic relation generation: State of the art and visions for future research,” 2024, *arXiv:2406.05397*.

[56] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer, “Adaptive stress testing for autonomous vehicles,” in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Piscataway, NJ, USA: IEEE Press, Jun. 2018, pp. 1–7, doi: 10.1109/IVS.2018.8500400.

[57] A. Corso, P. Du, K. R. Driggs-Campbell, and M. J. Kochenderfer, “Adaptive stress testing with reward augmentation for autonomous vehicle validation,” in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Piscataway, NJ, USA: IEEE Press, Oct. 2019, pp. 163–168, doi: 10.1109/ITSC.2019.8917242.

[58] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a formal model of safe and scalable self-driving cars,” Oct. 2018, *arXiv:1708.06374*.

[59] D. Baumann, R. Pfeffer, and E. Sax, “Automatic generation of critical test cases for the development of highly automated driving functions,” in *Proc. IEEE 93rd Veh. Technol. Conf. (VTC)*, Piscataway, NJ, USA: IEEE Press, Apr. 2021, pp. 1–5, doi: 10.1109/VTC2021-Spring51267.2021.9448686.

[60] D. Karunakaran, S. Worrall, and E. Nebot, “Efficient statistical validation with edge cases to evaluate highly automated vehicles,” in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC),* Piscataway, NJ, USA: IEEE Press, Sep. 2020, pp. 1–8, doi: 10.1109/ITSC45102.2020.9294590.

[61] X. Ma et al., “A survey on cooperative co-evolutionary algorithms,” *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 421–441, Jun. 2019, doi: 10.1109/TEVC.2018.2868770.

[62] Z. Yang, K. Tang, and X. Yao, “Large scale evolutionary optimization using cooperative coevolution,” *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, Aug. 2008, doi: 10.1016/j.ins.2008.02.017.

[63] L. Panait, S. Luke, and J. F. Harrison, “Archive-based cooperative coevolutionary algorithms,” in *Proc. 8th Annu. Conf. Genet. Evol. Comput. (GECCO)*, New York, NY, USA: ACM, Jul. 2006, pp. 345–352, doi: 10.1145/1143997.1144060.

[64] Y. Deng et al., “A declarative metamorphic testing framework for autonomous driving,” *IEEE Trans. Softw. Eng.*, vol. 49, no. 4, pp. 1964–1982, Apr. 2023, doi: 10.1109/TSE.2022.3206427.

[65] F. U. Haq, D. Shin, S. Nejati, and L. Briand, “Can offline testing of deep neural networks replace their online testing? A case study of automated driving systems,” *Empirical Softw. Eng.*, vol. 26, no. 90, Jul. 2021, doi: 10.1007/s10664-021-09982-4.

[66] S. Sharifi, D. Shin, L. C. Briand, and N. Aschbacher, “Identifying the hazard boundary of ml-enabled autonomous systems using cooperative coevolutionary search,” *IEEE Trans. Softw. Eng.*, vol. 49, no. 12, pp. 5120–5138, Dec. 2023, doi: 10.1109/TSE.2023.3327575.

[67] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proc. 1st Annu. Conf. Robot Learn., Ser. Proc. Mach. Learn. Res.*, vol. 78. PMLR, Nov. 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[68] Y. Deng, X. Zheng, M. Zhang, G. Lou, and T. Zhang, “Scenario-based test reduction and prioritization for multi-module autonomous driving systems,” in *Proc. 30th ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, New York, NY, USA: ACM, Nov. 2022, pp. 82–93, doi: 10.1145/3540250.3549152.

[69] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series,” in *Proc. 3rd Int. Conf. Knowl. Discovery Data Mining (AAAIWS)*, vol. 10, Jul. 1994, pp. 359–370.

[70] H. Sakoe and S. Chiba, “Dynamic programming algorithm optimization for spoken word recognition,” *IEEE Trans. Acoust., Speech Signal Process.*, vol. 26, no. 1, pp. 43–49, Feb. 1978, doi: 10.1109/TASSP.1978.1163055.

[71] L. Bull, “Evolutionary computation in multi-agent environments: Partners,” in *Proc. 7th Int. Conf. Genetic Algorithms*, San Mateo, CA, USA: Morgan Kaufmann, Jul. 1997, pp. 370–377.

[72] L. Bull, *Evolutionary Computing in Multi-Agent Environments: Operators.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 43–52, doi: 10.1007/BFb0040758.

[73] R. P. Wiegand, W. C. Liles, and K. A. D. Jong, “An empirical analysis of collaboration methods in cooperative coevolutionary algorithms,” in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, vol. 2611, San Mateo, CA, USA: Morgan Kaufmann, Jul. 2001, pp. 1235–1245.

[74] A. D. Cioppa, C. D. Stefano, and A. Marcelli, “On the role of population size and niche radius in fitness sharing,” *IEEE Trans. Evol. Comput.*, vol. 8, no. 6, pp. 580–592, Dec. 2004, doi: 10.1109/TEVC.2004.837341.

[75] D. E. Goldberg and J. Richardson, “Genetic algorithms with sharing for multimodal function optimization,” in *Proc. Genet. Algorithms Their Appl. Proc. Second Int. Conf. Genet. Algorithms*, vol. 4149, Hillsdale, NJ, USA: Lawrence Erlbaum, 1987.

[76] A. Petrowski, “A clearing procedure as a niching method for genetic algorithms,” in *Proc. of IEEE Int. Conf. Evol. Comput. (ICEC),* Piscataway, NJ, USA: IEEE Press, May 1996, pp. 798–803, doi: 10.1109/ICEC.1996.542703.

[77] B. Sareni and L. Krahenbuhl, “Fitness sharing and niching methods revisited,” *IEEE Trans. Evol. Comput.*, vol. 2, no. 3, pp. 97–106, Sep. 1998, doi: 10.1109/4235.735432.

[78] M. Eigen, “Selforganization of matter and the evolution of biological macromolecules,” *Die Naturwissenschaften*, vol. 58, no. 10, pp. 465–523, Oct. 1971, doi: 10.1007/BF00623322.

[79] J. Horn and D. E. Goldberg, *A Timing Analysis of Convergence to Fitness Sharing Equilibrium.* Berlin, Heidelberg: Springer Sci. and Bus. Media LLC, 1998, pp. 23–33, doi: 10.1007/BFb0056846.

[80] S. W. Mahfoud, “Population size and genetic drift in fitness sharing,” in *Foundations of Genetic Algorithms.* New York, NY, USA: Elsevier, 1995, vol. 3, pp. 185–223, doi: 10.1016/B978-1-55860-356-1.50014-5.

[81] E. C. Osuna and D. Sudholt, “Analysis of the clearing diversity-preserving mechanism,” in *Proc. 14th ACM/SIGEVO Conf. Found. Genet. Algorithms (FOGA)*, New York, NY, USA: ACM, Jan. 2017, pp. 55–63, doi: 10.1145/3040718.3040731.

[82] D. R. Wilson and T. R. Martinez, “Improved heterogeneous distance functions,” *J. Artif. Intell. Res.*, vol. 6, pp. 1–34, Jan. 1997, doi: 10.1613/jair.346.

[83] C. A. C. Coello, “A comprehensive survey of evolutionary-based multiobjective optimization techniques,” *Knowl. Inf. Syst.*, vol. 1, no. 3, pp. 269–308, Aug. 1999, doi: 10.1007/BF03325101.

[84] K. C. Tan, E. F. Khor, T. H. Lee, and R. Sathikannan, “An evolutionary algorithm with advanced goal and priority specification for multi-objective optimization,” *J. Artif. Intell. Res.*, vol. 18, pp. 183–215, Feb. 2003, doi: 10.1613/jair.842.

[85] S. Luke, *Essentials of Metaheuristics*, 2nd ed. Lulu, 2013. Accessed: Dec. 04, 2024. [Online]. Available: http://cs.gmu.edu/∼sean/book/metaheuristics/

[86] M. Li and X. Yao, “Quality evaluation of solution sets in multiobjective optimisation: A survey,” *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–38, Mar. 2019, doi: 10.1145/3300148.

[87] J. R. Schott, “Fault tolerant design using single and multicriteria genetic algorithm optimization,” Master’s thesis, Massachusetts Institute of Technology, 1995.

[88] D. V. Veldhuizen and G. Lamont, “On measuring multiobjective evolutionary algorithm performance,” in *Proc. Congr. Evol. Comput. CEC00 (Cat. No.00TH8512)*, vol. 1, Piscataway, NJ, USA: IEEE Press, Jul. 2000, pp. 204–211, doi: 10.1109/CEC.2000.870296.

[89] E. Zitzler, K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: Empirical results,” *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000, doi: 10.1162/106365600568202.

[90] H. Wang, Y. Jin, and X. Yao, “Diversity assessment in many-objective optimization,” *IEEE Trans. Cybern.*, vol. 47, no. 6, pp. 1510–1522, Jun. 2017, doi: 10.1109/TCYB.2016.2550502.

[91] Y. Tian, R. Cheng, X. Zhang, M. Li, and Y. Jin, “Diversity assessment of multi-objective evolutionary algorithms: Performance metric and benchmark problems [research frontier],” *IEEE Comput. Intell. Mag.*, vol. 14, no. 3, pp. 61–74, Aug. 2019, doi: 10.1109/MCI.2019.2919398.

[92] G. G. Yen and Z. He, “Performance metric ensemble for multiobjective evolutionary algorithms,” *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 131–144, Feb. 2014, doi: 10.1109/TEVC.2013.2240687.

[93] D. Whitley, *Next Generation Genetic Algorithms: A User’s Guide and Tutorial.* New York, NY, USA: Springer Int. Publishing, Sep. 2018, pp. 245–274, doi: 10.1007/978-3-319-91086-4_8.

[94] K. Deb and R. B. Agrawal, “Simulated binary crossover for continuous search space,” *Complex Syst.*, vol. 9, no. 2, pp. 115–148, Apr. 1995.

[95] G. Fraser and A. Arcuri, “EvoSuite: Automatic test suite generation for object-oriented software,” in *Proc. 19th ACM SIGSOFT Symp. 13th Eur. Conf. Found. Softw. Eng. (ESEC/FSE)*, New York, NY, USA: ACM, Sep. 2011, pp. 416–419, doi: 10.1145/2025113.2025179.

[96] A. Panichella, F. M. Kifetew, and P. Tonella, “Reformulating branch coverage as a many-objective optimization problem,” in *Proc. IEEE 8th Int. Conf. Softw. Testing, Verification Validation (ICST),* Apr. 2015, pp. 1–10, doi: 10.1109/ICST.2015.7102604.

[97] G. Fraser and A. Arcuri, “EvoSuite: On the challenges of test case generation in the real world,” in *Proc. IEEE 6th Int. Conf. Softw. Testing, Verification Validation,* Mar. 2013, pp. 362–369, doi: 10.1109/ICST.2013.51.

[98] A. Panichella, F. M. Kifetew, and P. Tonella, “Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets,” *IEEE Trans. Softw. Eng.*, vol. 44, no. 2, pp. 122–158, Feb. 2018, doi: 10.1109/TSE.2017.2663435.

[99] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* Cambridge, MA, USA: MIT Press, 1992.

[100] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proc. 1st Annu. Conf. Robot Learn.*, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, Nov. 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html.

[101] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, “Safety-enhanced autonomous driving using interpretable sensor fusion transformer,” in *Proc. 6th Conf. Robot Learn.*, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, Dec. 2023, pp. 726–737. [Online]. Available: https://proceedings.mlr.press/v205/shao23a.html.

[102] CARLA. “CARLA autonomous driving leaderboard.” Accessed: Dec. 04, 2024. [Online]. Available: https://leaderboard.carla.org/leaderboard/

[103] Y. Deng et al., “BMT: Behavior driven development-based metamorphic testing for autonomous driving models,” in *Proc. IEEE/ACM 6th Int. Workshop Metamorphic Testing (MET)*, Piscataway, NJ, USA: IEEE Press, Jun. 2021, pp. 32–36, doi: 10.1109/MET52542.2021.00012.

[104] M. Iqbal, “Metamorphic Relations for Testing Automated and Autonomous Driving Systems and Simulation Platforms,” Ph.D. dissertation, School of Computing and Information Technology, University of Wollongong, Feb. 2024. Doctor of Philosophy thesis. [Online]. Available: https://ro.uow.edu.au/theses1/1758.

[105] R. B. Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, “Automated repair of feature interaction failures in automated driving systems,” in *Proc. 29th ACM SIGSOFT Int. Symp. Softw. Testing Anal. (ISSTA)*, New York, NY, USA: ACM, Jul. 2020, pp. 88–100, doi: 10.1145/3395363.3397386.

[106] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen, “A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms,” *Soft Comput.*, vol. 23, no. 9, pp. 3137–3166, Dec. 2017, doi: 10.1007/s00500-017-2965-0.

[107] Y. Lavinas, C. Aranha, T. Sakurai, and M. Ladeira, “Experimental analysis of the tournament size on genetic algorithms,” in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC),* Piscataway, NJ, USA: IEEE Press, Oct. 2018, pp. 3647–3653, doi: 10.1109/SMC.2018.00617.

[108] P. Kaelo and M. Ali, “Integrated crossover rules in real coded genetic algorithms,” *Eur. J. Oper. Res.*, vol. 176, no. 1, pp. 60–76, Jan. 2007, doi: 10.1016/j.ejor.2005.07.025.

[109] S. Sette and L. Boullart, “Genetic programming: principles and applications,” *Eng. Appl. Artif. Intell.*, vol. 14, no. 6, pp. 727–736, Dec. 2001, doi: 10.1016/S0952-1976.(02)00013-1.

[110] S. Mirjalili, *Genetic Algorithm.* Springer Int. Publishing, Jun., 2018, pp. 43–55, doi: 10.1007/978-3-319-93025-1_4.

[111] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, “Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009,” in *Proc. 12th Annu. Conf. Companion Genet. Evol. Comput. (GECCO)*, New York, NY, USA: ACM, Jul., 2010, pp. 1689–1696, doi: https://doi.org/10.1145/1830761.1830790.

[112] R. Likert, “A technique for the measurement of attitudes,” *Arch. Psychol.*, vol. 22, no. 140, p. 55, 1932.

[113] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering.* Berlin, Heidelberg: Springer Sci. and Bus. Media LLC, Sep. 2024, doi: 10.1007/978-3-662-69306-3.