软件学报 ISSN 1000-9825, CODEN RUXUEW
2026,37(4):1671−1689 [doi: 10.13328/j.cnki.jos.007471] [CSTR: 32375.14.jos.007471]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于路网建模的自动驾驶关键场景生成与自适应演化方法*

田浩翔[1,2,3], 吴国全[1,2,3], 魏 峻[1,2,3], 郭 安[4], 韩星烁[5], 陈 伟[1,2,3], 王 伟[1,2,3], 叶 丹[1,2,3]

[1](中国科学院 软件研究所, 北京 100190)

[2](中国科学院大学, 北京 100049)

[3](计算机科学国家重点实验室 (中国科学院 软件研究所), 北京 100190)

[4](南京大学, 江苏 南京 210093)

[5](南京航空航天大学, 江苏 南京 211106)

通信作者: 吴国全, E-mail: gqwu@otcaix.iscas.ac.cn

摘 要: 自动驾驶系统的安全性对于自动驾驶汽车在现实世界中的实施非常重要. 因此, 自动驾驶系统在公开发布和部署之前必须进行充分的评估. 如何生成多样化的安全关键测试场景是自动驾驶系统测试的关键任务. 现有的自动驾驶系统关键场景生成方法, 包括再现现实世界的交通事故和基于搜索的关键场景生成. 然而, 前者由于自动驾驶与人类驾驶存在鸿沟, 现实世界交通事故大多无法发现自动驾驶系统的问题; 后者由于传统搜索算法的局限性, 发现的问题相似度较高. 此外, 由于测试场景的空间非常庞大, 二者生成关键场景的效率较低. 为了解决上述问题, 提出 LEADE, 一种基于道路网络建模的自动驾驶系统安全关键场景生成和自适应演化方法. 具体来说, 它根据用户的测试需求构建抽象场景, 并通过道路网络建模生成具体场景. 然后, LEADE 采用改进的自适应进化搜索来生成各种安全关键场景来测试自动驾驶系统. 在工业级全栈自动驾驶系统平台百度 Apollo 上实施和评估 LEADE. 实验结果表明, LEADE 可以有效和高效地生成安全关键场景, 并揭露 Apollo 的 10 种不同安全违背行为. 它通过识别同一道路上的 4 种新型安全关键场景, 优于两种最先进的基于搜索的自动驾驶系统测试技术.

关键词: 自动驾驶系统; 仿真测试; 场景程序生成

中图法分类号: TP311

中文引用格式: 田浩翔, 吴国全, 魏峻, 郭安, 韩星烁, 陈伟, 王伟, 叶丹. 基于路网建模的自动驾驶关键场景生成与自适应演化方法. 软件学报, 2026, 37(4): 1671–1689. http://www.jos.org.cn/1000-9825/7471.htm

英文引用格式: Tian HX, Wu GQ, Wei J, Guo A, Han XS, Chen W, Wang W, Ye D. Road-network-modeling-based Safety-critical Scenario Generation and Adaptive Evolution Approach for Autonomous Driving. Ruan Jian Xue Bao/Journal of Software, 2026, 37(4): 1671–1689 (in Chinese). http://www.jos.org.cn/1000-9825/7471.htm

Road-network-modeling-based Safety-critical Scenario Generation and Adaptive Evolution Approach for Autonomous Driving

TIAN Hao-Xiang[1,2,3], WU Guo-Quan[1,2,3], WEI Jun[1,2,3], GUO An[4], HAN Xing-Shuo[5], CHEN Wei[1,2,3], WANG Wei[1,2,3], YE Dan[1,2,3]

[1](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

[3](State Key Laboratory of Computer Science (Institute of Software, Chinese Academy of Sciences), Beijing 100190, China)

[4](Nanjing University, Nanjing 210093, China)

[5](Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

* 基金项目: 国家自然科学基金 (62472412); 中国科学院软件研究所创新基金重大重点项目 (ISCAS-ZD-202302); 中国科学院软件研究所基础研究项目 (ISCAS-JCMS-202402)
收稿时间: 2025-01-23; 修改时间: 2025-03-17, 2025-05-08; 采用时间: 2025-05-11; jos 在线出版时间: 2025-09-17
CNKI 网络首发时间: 2025-09-18

Abstract: The safety of autonomous driving systems (ADSs) is crucial for the implementation of autonomous vehicles (AVs). Therefore, ADSs must undergo thorough evaluation before being released and deployed publicly. Generating diverse, safety-critical test scenarios is a key task for ADS testing. Existing methods for generating ADS test scenarios include reproducing real-world traffic accidents or using search-based techniques. However, the accident-based scenario often fails to uncover safety violations in ADSs due to the gap between human driving and ADSs. The search-based approach tends to produce scenarios with high similarity because of the limitations of the search algorithm. To address these issues, this study proposes LEADE, a road network modeling-based safety-critical scenario generation and adaptive evolution method for ADSs. Specifically, it constructs abstract scenarios from user test requirements and generates concrete scenarios through road network modeling. LEADE then employs an improved adaptive evolutionary search to generate diverse safety-critical scenarios for testing the ADS. LEADE is implemented and evaluated on an industrial-grade full-stack ADS platform, Baidu Apollo. Experimental results demonstrate that LEADE can effectively and efficiently generate safety-critical scenarios and expose 10 diverse safety violations of Apollo. LEADE outperforms two state-of-the-art search-based ADS testing techniques by identifying 4 new types of safety-critical scenarios on the same roads.

Key words: autonomous driving system (ADS); simulation test; scenario program generation

自动驾驶系统的安全性和可靠性对于自动驾驶车辆的发展和应用至关重要[1]. 在将自动驾驶系统投入实际使用之前, 进行全面而彻底的测试显得尤为重要[2]. 这一测试过程需要涵盖大量多样化且全面的交通场景. 然而, 在公共道路上进行自动驾驶车辆测试不仅成本高昂, 而且可能增加事故风险. 与此相对, 仿真测试能够以极低的成本创建丰富的测试场景, 因此在自动驾驶系统测试中得到了广泛应用. 目前, 约 90% 的自动驾驶系统测试通过仿真平台完成, 9% 通过测试场地完成, 而只有 1% 通过实际道路测试完成[3].

然而, 由于现实交通的复杂性, “长尾”问题成为实现自动驾驶的主要限制[4]. 尽管现有技术能够处理超过 90% 的交通场景, 但如果不解决剩余的 10% 长尾场景, 实现真正公共道路上的自动驾驶将依然面临障碍[5]. 因此, 如何生成多样化的安全关键场景, 以有效应对长尾问题, 是自动驾驶车辆测试的关键任务[4,5].

传统的场景生成策略主要侧重于模拟自然驾驶环境[6,7]. 然而, 由于安全关键事件在自然条件下的稀有性, 从模拟的公共交通驾驶环境中生成的测试场景难以彻底揭示自动驾驶系统的安全隐患. 为了解决这一问题, 研究人员提出了新的生成安全关键场景的方法, 广义上可分为两类. 第 1 类方法基于交通事故场景生成, 通过从事故数据库中重现真实世界发生的交通事故场景来测试自动驾驶系统[8−11]. 然而, 这些测试场景数据集较为固定且难以覆盖场景空间, 被测试的自动驾驶系统可能在这些测试中表现良好, 但其在更广泛条件下的表现可能没有得到充分评估. 此外, 现实世界的交通事故大多是由人类驾驶员的失误造成的, 对自动驾驶系统缺乏挑战性. 第 2 类方法是更具前景的基于搜索的安全关键场景生成[12−15]. 其核心思想是在仿真空间中广泛搜索, 以发现安全关键场景[16−23]. 然而, 由于输入空间的高维度和复杂性, 这些方法通常需要大量时间才能发现自动驾驶系统的安全关键场景. 此外, 这些技术往往会重复发现与已知安全关键场景相似的场景, 从而导致其难以生成多样化的安全关键场景, 且只能识别有限类型的自动驾驶系统安全隐患.

通过对现有自动驾驶系统关键场景生成方法的分析, 本文总结出以下挑战.

(1) 如何设计和实现轻量级方法, 从抽象场景生成测试场景: 现有研究[19,24,25]基于预定义的规则和约束将抽象场景转化为测试场景. 这些方法需要为各种行为定义大量规则和约束, 以生成相应的轨迹. 例如, Scenic[24]、CRISCO[19]和 SoVAR[25]分别定义了数百条规则或约束来生成相应行为的轨迹, 而其实现往往需要成千上万行代码.

(2) 如何自适应地进化测试场景, 以生成多样化的安全关键场景: 许多自动驾驶系统测试方法采用多目标遗传算法来进化场景, 搜索仿真空间中的安全关键场景[17,18,20−23]. 然而, 这些方法中所有参与者的轨迹在选择父场景时都经过相同的变异概率, 这不利于高适应度场景中优秀特征的传播, 进而降低了多样化的安全关键场景的搜索效率和探索长尾场景空间的能力.

为了解决上述问题, 我们提出了 LEADE. 该方法根据用户的测试需求合成抽象场景, 并设计了一种轻量级方法从抽象场景生成具体测试场景. 随后, LEADE 利用改进的自适应遗传算法有效地进化这些场景, 引导它们更有效和更高效地探索场景空间, 生成多样化的安全关键场景. 具体来说: 为解决挑战 (1), LEADE 基于给定的抽象场

景, 采用基于道路网络的运动建模生成测试场景, 并将其脚本化为可执行的仿真程序, 从而避免为每种行为定义繁琐的轨迹生成规则. 为解决挑战 (2), LEADE 引入了自适应选择和细粒度自适应变异方法, 能够根据搜索过程的反馈和轨迹在不同目标上的得分, 动态调整变异和交叉的概率, 并使用自适应变异和交叉因子, 更高效地发现多样化的安全关键场景.

LEADE 由 3 个核心模块组成: ① 测试需求获取: 该模块通过与用户进行交互, 获取其各方面的测试需求, 并将这些需求转化为抽象场景. ② 场景程序生成: 基于抽象场景, LEADE 设计并实现了一种轻量级的具体场景生成方法, 通过基于道路网络的动作模型, 根据抽象场景的参与者动作, 在给定道路上生成符合动作语义的轨迹. ③ 场景自适应进化搜索: 该模块通过自适应选择方法, 筛选优秀个体并运用细粒度的自适应变异操作, 进行种群进化, 搜索多样化的安全关键场景. LEADE 工具源码提供在: https://github.com/ADStesting-test/Leade.

我们在百度 Apollo[26]上实现并评估了 LEADE. Apollo 是一个广泛用于控制公共道路上自动驾驶车辆的工业级全栈自动驾驶系统平台. 实验结果表明, LEADE 能够有效且高效地生成安全关键场景, 并成功发现 Apollo 系统中的多种不同的安全违背情况. LEADE 在 14 h 内揭示了 Apollo 的 10 种不同类型的安全违背场景, 并比目前最先进的基于搜索的自动驾驶测试方法, 多生成 4 种类型的新的安全关键场景.

本文的贡献如下.

1) 提出了一种轻量级的方法, 可以根据自由格式的用户需求文本, 生成可执行的测试场景, 并有效且高效地将其进化为多样化的安全关键场景.

2) 基于路网建模, 生成符合测试需求且易于演化搜索的场景程序; 并运用自适应选择和细粒度变异, 搜索更多样化的安全关键场景.

3) 在工业级四级自动驾驶系统平台上进行了有效性和效率评估, 并对 LEADE 及其发现的安全违例进行了全面分析.

1 相关工作

1.1 自动驾驶系统

自动驾驶系统是自动驾驶车辆的大脑, 通过集成多种传感器和算法, 负责执行包括导航、转向、刹车、加速以及应对动态路况和障碍物在内的多项驾驶任务. 自动驾驶系统在智能交通的安全性和效率中发挥着至关重要的作用.

为了确保研究工作的实用性, 我们选择了工业级自动驾驶系统——开源的全栈自动驾驶系统——百度 Apollo[26], 因为它具有代表性、实用性和先进性. ① 代表性: Apollo 是全球 4 大领先的工业自动驾驶系统开发者之一[27](其他 3 家分别是 Waymo、Ford 和 Cruise, 但这些系统未公开发布). ② 实用性: Apollo 可以轻松安装到车辆上, 并在公共道路上进行驾驶[28]. Apollo 已为真实车辆提供自动驾驶服务[29,30]. ③ 先进性: Apollo 系统持续积极更新, 持续加入最新的功能和特性.

1.2 自动驾驶仿真测试场景

在模拟测试中, 场景由可以在模拟器中执行的场景程序描述. 根据抽象级别[31], 自动驾驶系统仿真测试场景可分为抽象场景和具体场景. 抽象场景的特点是简单直接, 只需要描述场景测试需求, 如道路类型、参与者行为等, 而不需要指定具体的参数值. 具体场景需要根据抽象场景的要求, 在仿真环境空间内, 为场景每个元素, 详细准确地确定所有参数取值, 例如道路范围、速度、位置和时间等.

1.3 自动驾驶系统测试场景生成

考虑到自动驾驶系统的重要性, 在将其部署到实际道路之前, 全面的安全性测试是至关重要的. 关键思想是生成多样化的安全关键场景, 并在仿真中测试车辆在这些场景下的行为. 这样可以为自动驾驶系统的内部设计和算法提供有价值的反馈. 安全性测试的核心在于场景.

测试的有效性在很大程度上取决于生成场景的质量. 由于自动驾驶系统的输入空间巨大且功能复杂[32−34], 传统的软件测试方法难以从复杂的驾驶情境中捕捉到各种稀有事件[16]. 如何生成多样化的安全关键场景以全面测试自动驾驶系统, 近年来成为研究的热点[35−37]. 这些方法可以分为如下两类策略.

1.3.1 基于交通事故的场景生成方法

这类方法策略, 通过从事故数据库中重现交通事故作为测试场景[8,11,25,38]. 测试人员让主车 (车辆连接自动驾驶系统) 根据碰撞中车辆的路线行驶, 并通过事故中其他车辆和行人的轨迹定义场景中其他参与者的轨迹. 然而, 在真实的交通事故中[6], 超过 55% 的碰撞无法有效地发现自动驾驶系统的安全违背行为, 例如由以下原因引起的碰撞: 驾驶员的注意力不集中和分心、车辆硬件的组件、非法操作 (如逆行), 这些行为不会发生在自动驾驶系统上或不是自动驾驶系统的责任. 此外, 由于交通事故在实际交通中很少见, 而且大多数事故都不能对 ADS 造成有效的挑战, 因此通过再现交通事故来测试自动驾驶系统并不能充分评估其安全性和可靠性.

具体来说, AC3R[8]利用领域特定本体和自然语言处理 (NLP) 技术, 从警方事故报告中提取信息, 重建交通事故的边缘案例. SoVAR[25]利用大语言模型从警方报告中提取关键信息, 并根据这些信息生成相应的测试场景. Zhang 等人[11]训练了一个全景分割模型, 从事故视频中提取有效信息并恢复仿真中的交通参与者. 然而, 这需要大量时间来训练和优化信息提取模型. 此外, 由于交通事故在实际交通中相对稀少, 大多数事故并不能有效地挑战自动驾驶系统[39], 因此仅通过重现交通事故的场景无法全面测试工业级的自动驾驶系统. 与这些方法不同, LEADE 设计并实现了一种轻量级方法, 首先将用户需求合成抽象场景, 然后通过基于道路网络的运动建模, 从抽象场景生成可执行的测试场景.

1.3.2 基于搜索的场景生成方法

进化搜索技术在自动驾驶系统测试中被广泛应用[17,18,20−23], 通过引导生成的测试场景进入更具挑战性的情况. 一般来说, 现有的基于进化搜索的自动驾驶系统测试场景生成过程包括 3 个步骤: 1) 随机初始化生成第 1 代场景; 2) 运行并通过适应度函数评估每个生成的场景; 3) 选择适应度较高的场景并应用变异操作, 迭代生成新场景.

Abdessalem 等人[18,40]将多目标搜索与代理模型结合, 生成适用于自动驾驶系统的关键场景. AV-Fuzzer[22] 和 MOSAT[23]通过扰动参与者在动态交通环境中的驾驶行为, 生成面向安全违例的测试场景. 尽管这些方法有效, 但仍存在两个问题: 1) 第 1 代场景是通过随机初始化生成的; 2) 在进化过程中, 选择和变异的参数 (如变异和交叉概率) 是固定的, 没有根据每代的适应度水平和优秀个体的特征差异进行调整. 与这些方法不同, LEADE 解析用户需求并基于路网构建运动模型, 生成符合测试需求的参与者运动轨迹, 形成较高质量的初始种群; 进而通过自适应选择和自适应变异进行演化, 从而高效地发现多样化的安全关键场景.

2 方 法

后文图 1 显示了 LEADE 的整体框架, 包含 3 个主要模块: 测试需求提取、场景程序生成和自适应进化搜索. ① 测试需求提取: 该模块从用户提供的自由格式文本中提取测试需求, 并将其转化为抽象场景. ② 场景程序生成: 针对抽象场景, LEADE 不需要预定义大量规则来生成具体场景, 而是利用基于道路网络的运动建模方法, 生成可执行的具体场景程序. ③ 场景自适应演化: 基于生成的符合用户测试需求的场景程序, LEADE 构建初始种群, 采用改进的自适应多目标进化算法进行搜索, 进化出多样化且具备安全关键性的具体场景. 对于自动驾驶系统出现安全违背行为的场景, LEADE 记录并提供自动重放方法从而复现自动驾驶系统的安全违背情况.

2.1 测试需求提取

LEADE 通过自由格式的文本文件来捕捉用户的测试需求, 文件中可以包含场景的任意元素需求, 如天气、道路类型、路面状况、交通信号灯、标志以及交通参与者的类型 (例如车辆或行人) 及其动作. LEADE 使用大语言模型 (LLM, 如 GPT-4、LLaMA-7B 等) 从场景描述文本中, 提取关键信息. 为了保证场景元素定义的完备性, 我们参考现有的自动驾驶场景描述标准 OpenScenario[41,42]中, 对自动驾驶测试场景的定义 (在一定的时间和空间范围内, 自动驾驶汽车与行驶环境中的其他车辆、道路、交通设施、气象条件等元素综合交互过程的一种总体动态描

述), 结合美国国家公路交通管理局 (NHTSA)[6]对交通场景的文字描述, 确定了场景信息提取与定义所包含的关键元素: (1) 道路类型、车道数和路面条件; (2) 时间; (3) 天气; (4) 主车行为; (5) 从车数量、类型和行为; (6) 行人数量和行为; (7) 交通信号和标志. LEADE 使用的 prompt 提示如下所示.

You are an autonomous driving expert specializing in identifying dynamic objects in traffic scenarios. I will give you a scenario description file. Please use concise and structured language to extract the following elements (if given) in the scenario:

(1) Road: road type, number of lines, traffic signals, speed limits;

(2) Time and weather: time of the day, weather conditions;

(3) Ego vehicle: driving behavior;

(4) Vehicles: number of vehicles, vehicle types, driving behaviors of vehicles, lanes and relative positions of behaviors' starting and ending positions.

(5) Pedestrians: number of pedestrians, actions of pedestrians, lanes and relative positions of actions, starting and ending positions.

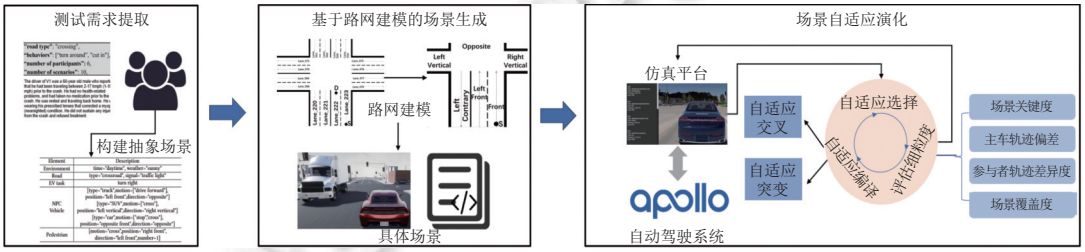


图 1 LEADE 框架图

以下通过一个示例来表达测试需求: 在一个晴天的交叉路口, 主车 (场景中连接被测自动驾驶系统的车辆) 需向右转, 一辆卡车在旁边对向车道行驶, 一辆 SUV 从左侧横穿路口, 一辆车停在路口对面, 一个行人从右侧过马路. 基于输入的文本, LEADE 将每个场景元素被解析为一个元组. 所给示例的抽象场景的结构化内容如图 2 所示.

Environment	time="daytime", weather="sunny"
Road	type="crossroad", signal="traffic light"
EV task	turn right
NPC Vehicle	[type="truck",motion=["drive forward"], position="left front",direction="opposite"] [type="SUV",motion=["cross"], position="left vertical",direction="right vertical"] [type="car",motion=["stop","cross"], position="opposite front",direction="opposite"]
Pedestrian	[motion="cross",position="right front", direction="left front",number=1]

图 2 抽象场景示例

抽象场景的关键元素如下.

(1) 环境: 描述场景中的时间 (如白天或夜晚) 和天气 (如晴天或雨天), 用于定义具体场景的驾驶环境.

(2) 道路条件: 包括道路类型 (如交叉路口或高速公路) 和交通信号 (如交通信号灯), 定义场景中的道路特征.

(3) 主车任务: 描述主车 (连接被测自动驾驶系统的车辆) 在场景中的驾驶任务, 包括驾驶动作 (如转弯) 和行驶方向 (如向右转). 生成的具体场景将模拟主车与自动驾驶系统之间的交互.

(4) 从车: 描述场景中的非自动驾驶控制的车辆 (从车), 包括车辆类型 (如 SUV 或卡车)、运动动作 (如变道)、相对于主车的位置和方向 (如左前方).

(5) 行人描述场景中的行人, 包括行人数量、动作 (如穿越交叉口), 以及行人相对于主车的位置和行进方向 (如从左到右).

当用户输入的场景需求文本包含的信息不完整时, LEADE 会提示用户从可行的属性中进行选择. 例如, 当用户无特定测试需求时, LEADE 支持用户仅输入道路类型 (包括直路、十字路口、T 型路口), 而后输出该道路类型支持的从车和行人的行为类型, 供用户选择, 该约束如表 1 道路类型支持的行为类型所示.

表 1 道路类型支持的行为类型

行为类型及约束	描述
从车行为类型全集	沿车道行驶, 向左变道, 向右变道, 左转, 右转, 横穿, 停车
行人行为类型全集	沿道路行走, 横穿道路, 驻足等待
道路上的行为约束	If 道路类型 = = 高速直路, 可选行为类型 = 从车行为类型全集 – {左转、右转、横穿}; If 道路类型 = = 高速路口, 可选行为类型 = 从车行为类型全集 – {横穿}; If 道路类型 = = 城区直路, 可选行为类型 = 从车行为类型全集 – {左转、右转、横穿} + 行人行为类型全集; If 道路类型 = = 城区直路, 可选行为类型 = 从车行为类型全集 + 行人行为类型全集;

2.2 场景程序生成

根据每个抽象场景, LEADE 生成可执行的具体场景程序. 该过程涉及 3 个主要任务: 1) 定义主车的起始位置和目标位置 (即主车连接到自动驾驶系统的位置); 2) 定义从车和行人的运动轨迹; 3) 定义时间和天气参数. 为解决上述任务, LEADE 采用基于道路网络的建模方法来生成具体场景. 以下是生成过程的详细步骤.

2.2.1 基于主车起始位置和终点位置的路网建模

LEADE 对地图文件进行解析, 获取地图中各道路的类型属性值; 而后根据用户指定的道路类型, 匹配地图中的道路; 从匹配到的道路中, 随机选择一条道路 R, 并提取该道路的车道信息, 包括车道 ID、车道方向和车道长度. 然后, 基于主车的驾驶任务, 确定主车的起始位置 S 和终点位置 D.

LEADE 首先选择合适的车道并为主车标记起始和目标位置. 基于选择的道路, LEADE 根据车道结构和主车的起始位置, 将道路划分为多个相对区域 (如图 3 所示), 包括正前方 (front)、左前方 (left front)、右前方 (right front)、正后方 (behind)、左后方 (left behind)、右后方 (right behind)、旁侧对向 (left opposite)、左侧垂直车道 (left vertical)、右侧垂直车道 (right vertical)、对侧 (opposite), 这些区域构成了路网模型, 如表 2 所示. 基于路网模型, LEADE 定义了一系列确定主车起始位置和终点位置的规则, 如表 3 所示. 其中 L_i 表示 ID 为 i 的车道, LD_i 表示车道 L_i 的方向, LE_i 表示车道 L_i 的长度, LT_S 表示位置 S 的车道距离. 以主车任务“向左变道”为例, 其起点在一条通向路口的车道上, 终点在左侧垂直车道的区域中, 入口与该路口的车道连接, 主车起点和终点如图 3 的 S 和 D 所示.

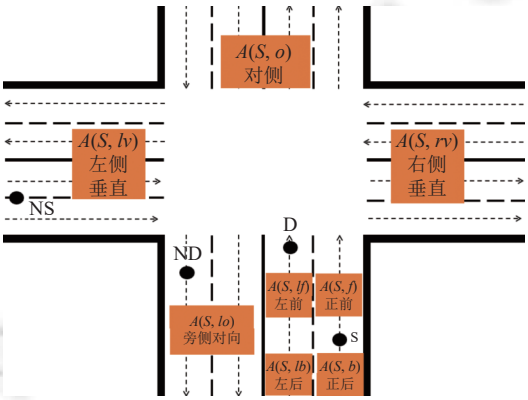


图 3 路网结构图

表 2 路网区域划分

路网区域	划分规则
$A(S,f)$	与主车起点车道相同且位于主车前方的区域
$A(S,lf)$	位于主车起点前方的左侧、通向路口且与主车起点所在车道方向相同的区域
$A(S,rf)$	位于主车起点前方的右侧、通向路口且与主车起点所在车道方向相同的区域
$A(S,b)$	与主车起点车道相同且位于主车后方的区域
$A(S,lb)$	位于主车起点后方的左侧、通向路口且与主车起点所在车道方向相同的区域
$A(S,rb)$	位于主车起点后方的右侧、通向路口且与主车起点所在车道方向相同的区域
$A(S,lo)$	位于主车起点左侧、与主车起点所在车道方向相反的路口前的区域
$A(S,lv)$	位于主车起点所在车道通向路口的左侧垂直车道
$A(S,rv)$	位于主车起点所在车道通向路口的右侧垂直车道
$A(S,o)$	位于主车起点所在车道通向路口的对侧车道

表 3 主车起终点定义规则

主车行驶任务	起终点规则
沿车道行驶	$S \in L_i,\ LT_S < LE_i$ $D \in L_i,\ LT_S < LT_D \leqslant LE_i$
向左变道	$S \in L_i,\ LT_S < LE_i$ $D \in A(S,lf),\ LT_D > LT_S$
向右变道	$S \in L_i,\ LT_S < LE_i$ $D \in A(S,rf),\ LT_D > LT_S$
横穿	$S \in L_i,\ LT_S < LE_i$ $D \in L_j,\ L_j \in A(S,o),\ LD_j = LD_i$
右转	$S \in L_i,\ LT_S < LE_i$ $D \in L_j,\ L_j \in A(S,rv),\ \sin\left(LD_i, LD_j\right) = 1$
左转	$S \in L_i,\ LT_S < LE_i$ $D \in L_j,\ L_j \in A(S,lv),\ \sin\left(LD_i, LD_j\right) = -1$

2.2.2 基于路网建模的参与者 (从车和行人) 行为轨迹生成

根据构建的路网模型, LEADE 对参与者行为的轨迹进行建模. 我们从当前最通用的交通数据库 NHTSA (美国国家公路交通管理局[6], 被广泛使用于交通仿真场景构建) 与行人行为基准数据集[43]中, 总结了 7 种从车的行为类型和 3 种行人的行为类型, 如表 1 所示. 这些类型的行为, 能够覆盖各种各样的交通场景的定义需求. 对于每种行为, 我们从其轨迹起终点和中间路点的如下方面: 相对于主车的方位、距离、车道、运动方向、速度, 制定其轨迹生成规则. 这里以表 4 所示的“转向”这一从车行为 (包括左转和右转) 为例, 进行说明, 其余运动类型可在 https://github.com/ADStesting-test/Leade 进行查阅. 在参与者行为轨迹模型中, W_0 表示从车或行人的初始位置, W_f 表示其结束位置, i 是主车起始位置所在车道的 ID. POS 为位置, DIR 为方向. 以表 4 第 1 行为例, 位于主车左侧垂直车道的从车, 进行右转时, 其起点选在主车左侧垂直车道且车道通向路口, 终点选在主车左前方道路且入口与该路口的车道连接, 从车起点和终点如图 3 的 NS 和 ND 所示.

在生成参与者行为轨迹 $(W_0, W_1, \ldots, W_f)$ 时, 有以下几条通用约束条件.

(1) 禁止从车逆向行驶: 对于从车的两个路径点 W_p, W_{p+1}, 若 $W_p \in L_m$ 同时 $W_{p+1} \in L_n$, 如果 $L_m == L_n$, 则 $LT_{W_{p+1}} \geqslant LT_{W_p}$.

(2) 车辆行驶方向约束: 车辆的行驶方向应与车道方向一致.

(3) 车辆速度约束: 车辆的行驶速度不得超过道路的最大速度限制.

(4) 行人速度约束: 行人的行走速度不得超过 3 m/s.

2.2.3 时间和天气的参数定义

时间类别是离散的, 而天气类别是一个字典变量, 其键值对分别表示天气类型和天气类型的量化值 (该值将

在 [0, 1] 区间内转化为实际值). 因此, LEADE 将一天中的时间编码为离散向量, 将天气编码为向量 $[0,1]^{|wk|}$, 其中 $|wk|$ 表示模拟器中定义的天气种类数量.

表 4 从车转向动作的轨迹模型

相对位置	行为方向	轨迹规则
左侧垂直	向右	$W_0 \in L_m$, $L_m \in A(S,lv)$, $\sin(LD_i,LD_m)=-1$ $W_f \in L_n$, $L_n \in A(S,lf)$, $\cos(LD_i,LD_n)=-1$
	向左	$W_0 \in L_m$, $L_m \in A(S,lv)$, $\sin(LD_i,LD_m)=-1$ $W_f \in L_n$, $L_n \in A(S,o)$, $LD_n=LD_i$
右侧垂直	向右	$W_0 \in L_m$, $L_m \in A(S,rv)$, $\sin(LD_i,LD_m)=-1$ $W_f \in L_n$, $L_n \in A(S,o)$, $LD_n=LD_i$
	向左	$W_0 \in L_m$, $L_m \in A(S,rv)$, $\sin(LD_i,LD_m)=-1$ $W_f \in L_n$, $L_n \in A(S,lf)$, $\cos(LD_i,LD_n)=-1$
对侧	向右	$W_0 \in L_m$, $L_m \in A(S,o)$, $\cos(LD_i,LD_m)=-1$ $W_f \in L_n$, $L_n \in A(S,rv)$, $\sin(LD_i,LD_n)=1$
	向左	$W_0 \in L_m$, $L_m \in A(S,o)$, $\cos(LD_i,LD_m)=-1$ $W_f \in L_n$, $L_n \in A(S,lv)$, $\sin(LD_i,LD_n)=-1$
正前	向右	$W_0 \in L_i$, $LT_{W_0} > LT_S$ $W_f \in L_n$, $L_n \in A(S,rv)$, $\sin(LD_i,LD_n)=1$
	向左	$W_0 \in L_i$, $LT_{W_0} > LT_S$ $W_f \in L_n$, $L_n \in A(S,lv)$, $\sin(LD_i,LD_n)=-1$
左前	向右	$W_0 \in L_m$, $L_m \in A(S,lf)$, $LD_m=LD_i$, $LT_{W_0} > LT_S$ $W_f \in L_n$, $L_n \in A(S,rv)$, $\sin(LD_i,LD_n)=1$
	向左	$W_0 \in L_m$, $L_m \in A(S,lf)$, $LD_m=LD_i$, $LT_{W_0} > LT_S$ $W_f \in L_n$, $L_n \in A(S,lv)$, $\sin(LD_i,LD_n)=-1$
右前	向右	$W_0 \in L_m$, $L_m \in A(S,rf)$, $LT_{W_0} > LT_S$ $W_f \in L_n$, $L_n \in A(S,rv)$, $\sin(LD_i,LD_n)=1$
	向左	$W_0 \in L_m$, $L_m \in A(S,rf)$, $LT_{W_0} > LT_S$ $W_f \in L_n$, $L_n \in A(S,lv)$, $\sin(LD_i,LD_n)=-1$
正后	向右	$W_0 \in L_i$, $LT_{W_0} < LT_S$ $W_f \in L_n$, $L_n \in A(S,rv)$, $\sin(LD_i,LD_n)=1$
	向左	$W_0 \in L_i$, $LT_{W_0} < LT_S$ $W_f \in L_n$, $L_n \in A(S,lv)$, $\sin(LD_i,LD_n)=-1$
左后	向右	$W_0 \in L_m$, $L_m \in A(S,lf)$, $LD_m=LD_i$, $LT_{W_0} < LT_S$ $W_f \in L_n$, $L_n \in A(S,rv)$, $\sin(LD_i,LD_n)=1$
	向左	$W_0 \in L_m$, $L_m \in A(S,lf)$, $LD_m=LD_i$, $LT_{W_0} < LT_S$ $W_f \in L_n$, $L_n \in A(S,lv)$, $\sin(LD_i,LD_n)=-1$
右后	向右	$W_0 \in L_m$, $L_m \in A(S,rf)$, $LT_{W_0} < LT_S$ $W_f \in L_n$, $L_n \in A(S,rv)$, $\sin(LD_i,LD_n)=1$
	向左	$W_0 \in L_m$, $L_m \in A(S,rf)$, $LT_{W_0} < LT_S$ $W_f \in L_n$, $L_n \in A(S,lv)$, $\sin(LD_i,LD_n)=-1$

2.2.4 场景检查

同时由于生成的场景需要遵循现实道路操作, LEADE 检查生成的参与者轨迹的可行性. 这一过程确保从车在正确的位置、方向和速度下操作, 并且不违反交通规则. 为此, LEADE 根据前述的通用约束条件检查轨迹的可行性.

(1) 车辆行驶方向约束: 车辆的行驶方向应与车道方向一致, 并且不能在同一车道上逆行.

(2) 空间约束: 每辆从车的初始位置不能部分重叠, 并且车辆之间至少要保持 5 m 的间距.

(3) 车速约束: 车辆的行驶速度不得超过道路的限速.

(4) 时间约束: 每辆从车的速度和位置变化应遵循轨迹状态的顺序, 确保运动的时序一致.

对于每个抽象场景, LEADE 会生成 k 个可执行且可行的具体场景作为初代种群.

2.3 场景自适应演化

基于生成的场景程序, LEADE 执行并测试自动驾驶系统. 若场景执行过程中, 没有发现主车的安全违背, 则开始后续的演化过程: LEADE 根据主车在该场景中的具体行驶轨迹, 对其他参与者的轨迹进行针对性的自适应修改, 以提高场景参与者对主车行驶过程的干扰和挑战. LEADE 采用改进的自适应遗传算法进行搜索, 以发现更多具有安全关键性和多样性的具体场景. 算法过程如算法 1 所示. 在启动进化搜索时, LEADE 基于上一步生成的符合测试需求的场景程序, 构建初代场景. 接下来, LEADE 通过多目标进化搜索方法进行场景搜索, 包括细粒度评估、自适应选择、自适应变异操作, 旨在提高高质量场景的遗传能力, 并增强低质量场景的变异性. LEADE 执行生成的场景来测试自动驾驶系统, 并记录自动驾驶系统中引发安全违背的场景, 这些场景可以自动重现并用于重新测试自动驾驶系统.

算法 1. 安全关键场景下改进的自适应进化搜索算法.

输入: 第 1 代个体 P (第 2.2 节生成的场景程序);
输出: 安全关键场景集合 SC.

1. $SC, TP, TS, EP \leftarrow \varnothing, \varnothing, \varnothing, \varnothing$
2. **while** not *TerminationCondition*() **do**
3. $EX, BN \leftarrow \varnothing, \varnothing$
4. **for** $x_i \in |TP[-1]|$ **do**
5. $ex_i = execute(x_i), EX \leftarrow EX \cup ex_i$
6. **if** *isEgoCollision*() **then**
7. $SC \leftarrow SC \cup x_i$
8. **end if**
9. $S \leftarrow \arg_{x \in EX} \{\min f_c(x), \min f_e(x), \min f_d(x), \min f_o(x)\}$
10. $TS \leftarrow TS \cup S, PN \leftarrow \varnothing$
11. **if** $|TP|==1$ **then**
12. $PN \leftarrow P$
13. **else**
14. $PN \leftarrow sort(TP, TS)$
15. $update(TP, TS, PN)$
16. $threshold \leftarrow generate\_variation\_threshold(S)$
17. **end if**
18. **for** $S_i \in PN$ **do**
19. $PM_i \leftarrow generate\_mutation\_probability(S_i, TS[-1])$
20. **if** $PM_i > threshold$ **then**
21. $x_i' \leftarrow scenario\_mutation(x_i)$
22. $BN \leftarrow BN \cup x_i'$
23. **end if**
24. **end for**

```
25.     for each two individuals S_i, S_j ∈ PN do
26.       PN_i ← generate_mutation_probability(S_i, S_j, TS[−1])
27.       if PM_i > threshold then
28.         x'_i, x'_j ← scenario_crossover(x_i, x_j)
29.         BN ← BN ∪ x'_i, x'_j
30.       end if
31.     end for
32.   TP ← TP ∪ BN
33. return SC
```

2.3.1 自适应进化搜索算法

基于生成的具体场景, LEADE 采用改进的自适应进化搜索算法, 旨在发现具有安全关键性和多样性的具体场景. 每个具体场景被编码为一个个体 $S_i = \{C_1, C_2, \ldots, C_n\}$, 其中 C_n 表示场景中第 n 个参与者的轨迹. 与现有的自动驾驶系统具体场景进化搜索方法不同, 现有方法通常让选定的父代个体的参与者接受相同程度的变异和交叉操作, 而 LEADE 则采用自适应选择和自适应变异策略, 这能够提高搜索安全关键场景的效率, 并发现更多类型的自动驾驶系统安全违例. 改进后的进化搜索流程如算法 1 所示.

细粒度评估具体场景 (第 4–9 行): 在每一代的进化过程中, LEADE 会根据多个目标构建改进的多帕累托 (Pareto) 最优解, 考虑的目标包括: 场景的关键性、主车的偏差、参与者的多样性以及场景覆盖度. 这些目标通过一系列精细化的度量指标来量化, 全面评估对自动驾驶系统的对抗性与多样化扰动.

自适应选择 (第 10–16 行): 基于多个目标的适应度评估, LEADE 会从当前种群中选出表现优异的个体作为父代, 进行交叉与变异操作, 从而逐步产生 Pareto 最优解. 在个体选择过程中, LEADE 会根据当前种群的适应度值自适应地调整交叉和变异的概率, 以确保探索的广度和深度.

自适应变异 (第 18–29 行): 自适应变异包括自适应交叉和自适应变异操作. LEADE 会根据种群中各个轨迹的适应度排名, 动态地选择不同的变异操作类型, 以确保在每一代中更好地探索和优化场景. 这种自适应变异机制能够提高安全违例场景搜索的效率, 并增加场景的多样性, 帮助发现更多潜在的安全隐患.

当连续 3 代保留的优秀个体未发生变化时, 重新生成初代种群, 并重启进化进程.

2.3.2 细粒度目标评估

为了寻找更多具有安全关键性和多样性的具体场景, 本文设计了一种包含上述目标的适应度函数, 并通过该函数对生成的场景进行评估.

场景关键性: 该指标用于识别能够暴露安全违例的具体场景. 为了更好地评估场景运行过程中, 主车与参与者进行交互时出现碰撞的可能性, 改进了最小预计碰撞时间 (MiTTC) 指标, 即场景运行过程所有时刻预计碰撞时间 (iTTC) 的最小值. iTTC 是指考虑了主车和从车的边界框、实时相对速度和距离的碰撞时间. 计算公式如下所示:

$$f_c = \min_{\substack{i\in[1,k]\\ t\in(0,s)}} \left\{ \frac{DX(b_t^E, b_t^{N_i})}{vx_{N_t^i} - vx_{E_t}} + \frac{vx_{N_t^i} - vx_{E_t}}{ax_{N_t^i} - ax_{E_t}}, \frac{DY(b_t^E, b_t^{N_i})}{vy_{N_t^i} - vy_{E_t}} + \frac{vy_{N_t^i} - vy_{E_t}}{ay_{N_t^i} - ay_{E_t}} \right\},$$

其中, N_i 表示场景中的第 i 个参与者, E 代表主车. $DX(b_t^E, b_t^{N_i})$ 和 $DY(b_t^E, b_t^{N_i})$ 分别表示主车和第 i 个参与者在时刻 t 的边界框的横向和纵向欧几里得距离, vx 和 vy 分别表示横向和纵向速度, ax 和 ay 分别表示横向和纵向加速度. 具体场景中有 k 个参与者, 场景的执行持续时间为 s 秒. f_c 值越小, 场景的适应度分数越高.

主车轨迹偏差: 该指标用于衡量主车实际行驶轨迹与预期路线之间的偏差. 在没有扰动的情况下, 主车的预期路线应沿着车道行驶并保持给定的速度. 指标 f_e 定义为主车行驶过程中最大的位置偏移量和加速度变化, 其计算公式如下:

$$f_e=\max_{t\in[2,s]}\left\{\frac{D(E_t,P_t)-D(E_{t-1},P_{t-1})}{D(E_{t-1},P_{t-1})}\right\}+\max_{t\in[2,s]}\left\{\frac{V_{E_t}-V_{E_{t-1}}}{V_{E_{t-1}}}\right\},$$

其中, E_t 表示 t 时刻主车的实际位置, P_t 表示预期路线在 t 时刻的预期位置, D 计算的是两者之间的距离, V 表示速度. f_e 越高, 场景的适应度分数越高.

参与者轨迹差异度: 该指标关注参与者轨迹的差异度. LEADE 通过计算参与者轨迹之间的欧氏距离来评估同一行为的参与者轨迹的差异度, 欧氏距离是广泛用于计算不同具体场景中参与者轨迹相似度的方法[22]. 对于具体场景 s_i, 多样性指标 f_d 的计算方式为: $f_d=\left(\sum_{j=1}^{r}ED_{s_i,s_j}\right)\Big/r$. 其中, ED 表示不同具体场景中参与者轨迹的欧氏距离. LEADE 计算了场景 s_i 中参与者轨迹与之前迭代中发现的 r 个安全违例场景 (记作 s_j) 中参与者轨迹的平均欧氏距离. ED_{s_i,s_j} 的计算方式如下:

$$ED_{s_i,s_j}=\frac{\sum_{n=1}^{l}\sum_{m=1}^{c}TD_{s_i^n,s_j^m}}{l\times c},$$

$$TD_{s_i^n,s_j^m}=\sum_{k=0}^{\mu}\sqrt{(x_{s_i^n.k}-x_{s_j^m.k})^2+(y_{s_i^n.k}-y_{s_j^m.k})^2},$$

其中, TD 表示不同参与者轨迹之间的欧氏距离. s_i^n 和 s_i^m 表示场景 s_i 中第 n 和 m 个参与者的轨迹. l 和 c 分别表示场景 s_i 和 s_j 中从车和行人的数量, $(x_{s_i^n.k},y_{s_i^n.k})$ 表示场景 s_i 中第 n 个参与者轨迹的第 k 个路点的坐标, μ 表示两条轨迹 s_i^n 和 s_j^m 中最小的路点数. f_d 越高, 场景的适应度分数越高.

场景覆盖率: 该指标关注场景如何覆盖之前生成的安全违例场景未覆盖的空间. 场景覆盖性通过位置和速度覆盖来评估. 每个路点包含位置和速度, 分别表示为 w 和 v. 场景覆盖性指标 f_o 的计算公式如下:

$$f_o=\left(\sum_{n=1}^{l}\sum_{k=1}^{\mu}\{w_k,v_k\}\in Tr_n\cap\{w_k,v_k\}\notin R\right)\Big/(l\times\mu),$$

其中, l 是参与者的数量, Tr_n 表示第 n 个参与者的轨迹, μ 表示第 n 个参与者轨迹中的路点数, R 表示在之前迭代中找到的安全违例场景中, 参与者轨迹覆盖的路点集合. f_o 越高, 具体场景的适应度分数越高.

每个场景的适应度分数在执行结束时进行计算. 当本代的所有场景都执行完毕后, LEADE 会选择当前代和上一代中适应度最高的 k 个个体作为下一代的父代个体. 父代个体的选择采用最优 Pareto 原则, 并通过拥挤距离 (crowding distance) 对个体进行排序.

2.3.3 自适应选择

对于父代个体, LEADE 根据其适应度值和种群的适应度水平自适应地决定哪些个体进行交叉和变异. 适应度较高的场景具有更大的交叉概率, 而适应度较低的场景则具有更大的变异概率.

对于种群 p_n 中具有适应度 f_i 的场景 s_i. 其中 $f_{\max}$ 和 $f_{\min}$ 分别表示 p_n 中的最大适应度值和最小适应度值. 对于每个种群 p_n, LEADE 计算适应度的平均值, 记为 $\bar{f}$. 场景 p_n 的变异概率为 PM_i. 对于两个场景 s_i 和 s_j, 它们的交叉概率表示为 $PC_{i,j}$, $f'_{i,j}$ 为它们的较大适应度值.

交叉个体的选择: 个体的适应度值越大, 它将其特征传递给其他个体创造更多不同优秀个体的概率越高. 对于场景 s_i 和 s_j, 它们的交叉概率 $PC_{i,j}$ 计算公式如下:

$$PC_{i,j}=\min\{k_1(f'_{i,j}-f_{\min})/(\bar{f}-f_{\min}),k_3\},\ 0<k_1,k_3\leqslant 1.$$

如果 $PC_{i,j}\geqslant threshold_n$, LEADE 将在场景 s_i 和 s_j 中各自随机选择一个参与者, 在两个场景之间进行交换.

变异场景的选择: 个体的适应度值越低, 其参数发生变异的概率越大. 场景 s_i 的变异概率计算公式如下:

$$PM_i=\min\{k_2(f_{\max}-f_i)/(f_{\max}-\bar{f}),k_4\},\ 0<k_2,k_4\leqslant 1.$$

如果 $PM_i\geqslant threshold_m^n$, LEADE 将选择主车行驶过程中与主车的 MiTTC 的参与者路点, 并改变该路点的参数值.

为了扰乱适应度值高于平均水平的个体, 以便在全局最优区域中进行空间搜索, 并确保所有适应度值低于平

均水平的个体强制进行变异, 我们将 k_1 和 k_2 的值设置为 0.6, 将 k_3 和 k_4 的值设置为 1.0, 其他值可以根据实际需要进行调整.

由于变异阈值 $threshold_n$ 对种群 p_n 的整体变异性影响较大, 当 p_n 中个体的适应度值趋于收敛时, 应适当降低 $threshold$ 以便促进个体的交叉和变异, 从而生成更多不同的后代. $threshold_n$ 的计算公式如下:

$$threshold_n = c_1(f_{\max} - \bar{f}) + m_1(\bar{f} - f_{\min}),\ 0 < c_1, m_1 \leqslant 1.$$

2.3.4 自适应进化操作

自适应进化包括自适应交叉和自适应变异.

自适应交叉: 对于要进行交叉的个体, 会根据个体在种群中的适应度排名, 选择目标函数得分最高的参与者轨迹进行交叉. 通过对不同个体之间的轨迹进行单点交叉, 来更好地传播优秀的染色体, 并生成更多不同的染色体, 这有助于发现更多类型的安全违背具体场景. 例如, 在图 4 中, 场景 P1 和 P2 被选中进行自适应交叉 (在 P1 和 P2 的执行过程中, 主车没有发生安全违例). P1 在当前种群中在多样性目标上的得分最高, 且车辆 a 的轨迹对 P1 的多样性贡献最大. LEADE 通过交换车辆 a 和车辆 m 的轨迹, 进行 P1 和 P2 的交叉. O2 是通过 P1 和 P2 的自适应交叉生成的新场景, 暴露了主车的新安全违背.

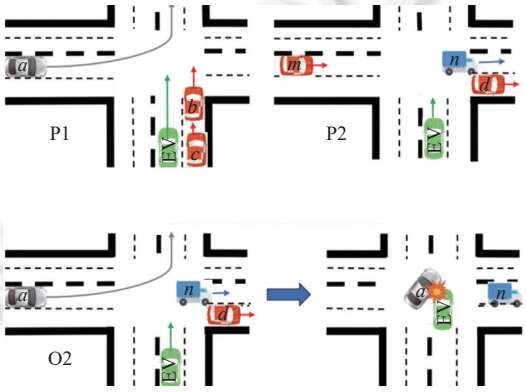


图 4 两个场景自适应交叉示例

自适应变异: 对于要突变的个体, LEADE 会根据参与者轨迹的适应度动态地确定不同类型的变异操作. 如果某一轨迹在种群中的某个目标上得分最高, LEADE 会轻微调整其参数 (如在其原始车道内修改速度和位置值), 以便更好地探索周围空间. 否则, LEADE 会对该轨迹进行较大的改动 (如将其位置改变到另一条车道, 或对其速度进行显著修改). 例如, 在图 5 中, 从车 b 的轨迹, 在该代种群中的关键度得分最高, 因此进行较小改动; 从车 c 的轨迹在各目标的得分都不是种群最高, 因此进行较大改动. 对于突变后生成的新轨迹, LEADE 会通过第 2.2.2 节提到的一般约束条件来检查其可行性.

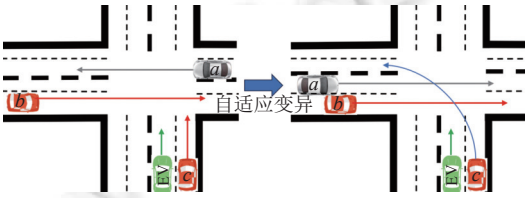


图 5 场景自适应变异示例

3 实验分析

为了展示 LEADE 的能力, 我们将其应用于测试工业级全栈自动驾驶系统百度 Apollo 7.0[26,28], 该系统广泛用

于控制公共道路上的自动驾驶车辆. 为评估 LEADE 的效率和效果, 我们回答以下研究问题.

问题 1: LEADE 在抽象场景构建和具体场景生成方面的全面性和准确性如何?

问题 2: LEADE 在发现 Apollo 的多样化安全违背方面的效果和效率如何?

问题 3: LEADE 相比于其他先进的具体场景演化方法, 其发现 Apollo 安全违背的有效性和效率如何?

3.1 实验设置

我们在 Ubuntu 20.04 操作系统上进行实验, 硬件配置包括 500 GB 内存、Intel Core i7 CPU 和 NVIDIA GTX 2080 Ti 显卡. 实验使用的仿真平台为 SORA-SVL (一个支持与 Apollo 连接的端到端自动驾驶仿真系统), 并选择了旧金山地图执行生成的场景. 在实验过程中, Apollo 的所有模块均处于启用状态, 包括感知、定位、预测、路径规划、规划和控制模块. Apollo 配备了多种传感器, 包括两个摄像头 (一个安装在车顶, 另一个位于主车前方)、GPS、雷达和激光雷达.

为了解决这一问题, 我们调研了现有的可以为 Apollo 描述和定义测试场景的领域专用语言[24,41,44], 选择了 AVUnit[44]作为基础工具, 因为它能够准确地定义并确定性地执行具体场景中参与者的运动轨迹. 在 AVUnit 的场景程序中, 每个轨迹由一系列状态定义, 每个状态包括位置、航向和速度. 位置以“lane_id→dist”表示, 其中 dist 为当前位置到车道起点的距离, 也称为车道距离.

LEADE 关注的主要安全要求包括: 自动驾驶系统应避免与场景中的其他参与者发生碰撞, 并确保顺利到达目的地. 为此, 我们定义了两个关键指标来监控主车的安全表现: 一是场景执行过程中主车与其他参与者之间的最小距离; 二是场景结束时主车与目的地之间的距离. 需要指出的是, AVUnit 在场景执行期间会确保从车和行人沿预定路线行驶, 同时遵循交通规则 (例如, 避免与其他车辆或主车发生碰撞, 保持车速不超过道路限速, 遵守车道规则). 因此, 主车与其他参与者的碰撞通常是由主车本身的安全违背行为引起的.

LEADE 中需要定义一些参数: $threshold_m$、$threshold_c$ 和 k. $threshold_m$ 是个体变异的阈值, $threshold_c$ 是个体交叉的阈值. 为了确定这些阈值, 我们测试了现有遗传算法推荐的阈值范围[45], 并分别选择了 0.3 作为 $threshold_m$, 0.7 作为 $threshold_c$. k 是每代中选择的最优个体数量. 为了平衡搜索效果与进化成本, LEADE 在自适应进化过程中随机生成 k, 其范围值是 [4, 8].

3.2 实验设计

针对问题 1, 我们使用 LEADE 从不同格式的测试需求文件中构建抽象场景, 并生成相应具体场景. 为了评估抽象场景构建的有效性. 我们根据自动驾驶的测试需求文件, 并从中随机选择了 100 个不同的场景需求文件, 这些文件涵盖了各种类型的道路、主车驾驶任务和参与者行为. 随后, 我们运行 LEADE 构建抽象场景. 为了验证 LEADE 在具体场景生成中的有效性, 我们使用 CRISCO[19]作为基准, 为相同的抽象场景, 生成具体场景. CRISCO 通过求解一系列约束来在抽象场景基础上生成具体场景. 4 位作者独立分析并交叉检查对每个测试需求文件的信息提取及为每个抽象场景生成的具体场景. 若检查结果存在不一致, 另一位作者参与小组讨论, 解决冲突并达成一致.

针对问题 2, 我们运行 LEADE 生成场景并在仿真平台模拟器上执行, 以测试 Apollo. 在发现的安全违背场景中, 我们设计了一套分类标准对其进行分类, 并分析 Apollo 各模块的潜在缺陷和正确操作.

针对问题 3, 我们将 LEADE 与 3 种采用遗传算法寻找 Apollo 安全违背场景的最新技术进行比较: AV-Fuzzer[22]、MOSAT[23]、BehAVExplor[46]. AV-Fuzzer 通过进化参与者行为来揭示安全违背, 而 MOSAT 则使用原子操作和模式生成具体场景, BehAVExplor 基于种子场景, 采用主车轨迹差异度引导的多目标遗传算法, 对具体场景进行演化. 这 3 种技术均使用遗传算法进行场景优化. 我们在相同的时间内, 对同样的初代场景, 运行 LEADE、AV-Fuzzer、MOSAT 和 BehAVExplor, 对场景进行演化, 并从以下几个方面比较它们的有效性和效率.

(1) 能发现多少种 Apollo 的安全违背类型?

(2) 生成一个场景需要多少时间?

(3) 平均生成多少个场景才能找到一个 Apollo 的安全违背?

(4) 平均需要多长时间才能暴露第 1 个安全违背及所有已发现的安全违背类型?

为了回答上述研究问题, 我们每个实验运行 24 h. 需要注意的是, 为了考虑基于搜索的技术的随机性, 所有实验重复进行了 5 次, 并对结果进行了统计分析和比较.

3.3 实验分析-问题 1

表 5 展示了 LEADE 在从交通视频中提取信息的准确性. 在提取静态信息 (如道路结构) 和简单的动态信息 (如主车行驶任务) 方面表现最佳, 在提取复杂的动态信息 (如参与者行为、相对位置) 时, 准确性稍有下降, 但基本能够准确地提取测试场景需求的关键信息, 正确构建抽象场景.

表 5 抽象场景构建的准确性 (%)

提取信息类型	道路	主车行驶任务	参与者行为	相对位置
准确性	98	96.6	94.2	89.8

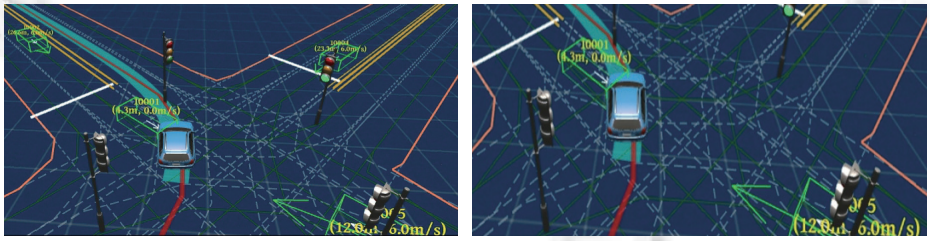
表 6 展示了 LEADE 和 CRISCO 在从抽象场景生成具体场景方面的准确性. 相比之下, LEADE 在结构相对简单的道路 (如直道) 上的准确性略高于 CRISCO, 随着道路结构复杂度升高, LEADE 生成具体场景的准确性略有下降, 但下降幅度显著低于 CRISCO. 此外 CRISCO 不支持生成弯路的具体场景, 并且 LEADE 无需定义大量约束条件.

表 6 具体场景生成的正确性 (%)

方法	直道场景	十字路口	T字路口	弯路
LEADE	95.5	91.4	90	89
CRISCO	93.2	85.6	75	—

3.4 实验分析-问题 2

在每次实验中, LEADE 平均生成 3 756 个场景 (最少 3 346 个, 最多 4 015 个), 其中 313 个场景 (最少 292 个, 最多 355 个) 包含主车的安全违背. 为了更好地分析 Apollo 的潜在缺陷, 对于每个发现的安全违背场景, 我们识别出导致主车安全违背的关键参与者. 例如, 图 6(a) 展示了一个安全违背场景, 而图 6(b) 展示了只包含关键参与者的安全违背场景, 这意味着如果从图 6(b) 中的场景中移除任何参与者, 主车的安全违背将不再发生. 为了识别每个安全违背场景的关键参与者, LEADE 首先通过逐一移除场景中的参与者并检查是否能够重现主车的安全违背来回放该场景.



(a) 演化生成的安全违背场景 (b) 只包含关键参与者的安全违背场景

图 6 安全违背具体场景示例

对于每个发现的安全违背场景, 我们都会检查安全违背是否是由场景中参与者的非法行为造成的, 并分析主车造成的安全违背. 图 7 显示了一个不是由主车引起的安全违背行为. 从车非法驶入本车车道并与本车相撞. 对于已发现的由主车引起的安全违背, 我们使用相同的标准对其进行分类.

图 7 非主车问题引发的安全违背示例

为了将发现的安全违背场景分类, 我们设计了一套分类标准, 考虑以下方面: 道路类型、主车的驾驶任务、参与者的行为、Apollo 的驾驶错误. 基于这些标准, 包含关键参与者的安全违背场景被分类为 10 种不同类型 (见

表 7), 所有类型均在每次实验的前 14 h 内揭示. EV 代表主车, NV 代表从车 (2 NV 表示两辆从车), P 代表行人.

由于篇幅限制, 以下将选择两种类型的安全违背场景进行说明 (EV 代表主车). 其他类型的安全违背说明可在 https://github.com/ADStesting-test/Leade 中查阅.

表 7　发现的违反 Apollo 安全的类型

类型	道路类型	主车驾驶任务	参与者			自动驾驶系统安全违背
			类型	相对主车初始位置	行为	
1	十字路口	左转 右转	从车×2 从车×2	左侧/右侧垂直车道 左侧垂直车道	横穿 横穿	对前后连续行驶的车辆速度, 产生误判: 当前车速度不变但后车速度变化时, 自动驾驶系统会误判后车的速度
2	多车道直路	左/右变道	从车×2	左/右前方 左/右后方	沿车道行驶 加速	变道时, 忽略邻车道后方从车的加速, 导致未避让直行有路权的车辆
3	十字路口	横穿	从车×2	左侧垂直车道	横穿 左转	将并排行驶在垂直车道上的两辆车误识别为一个实体, 忽略了减速后第2个抵达路口的从车的行为变化
4	十字路口	左转	从车	左侧垂直车道	左转	在路口连接区域转弯时, 对连接区内的从车路权等复杂信息的处理和响应, 存在明显的延迟
5	多车道直路	直行	从车×2	正前方 左/右前方	刹车 减速	行驶过程中, 变道后遇到阻塞, 未能二次变道
6	十字路口	右转	从车	右侧垂直车道	停车 (主车进入车道入口)	转向过程中, 转入车道存在阻塞时, 未能调整路线进入另一条可转入的车道
7	十字路口	左转 横穿	从车×2 从车×2	对侧车道 左侧垂直车道	横穿 横穿	对连续通过路口的从车之间的距离计算不准确
8	十字路口	横穿	从车 行人	左/右前方 左/右侧垂直车道	沿车道行驶, 刹车 横穿	当与主车并排或在主车侧前方行驶的从车, 出现紧急制动时, 自动驾驶系统未具备风险预测的经验
9	多车道直路	直行	从车	正前方	慢速行驶	应对前方多辆低速行驶从车的能力不足, 超车决策过慢, 在慢速超车时与前方车辆发生碰撞
10	十字路口	右转 横穿	大车 大车	左侧垂直车道 右侧垂直车道	横穿 右转	路口停车或转向时, 对大型车辆的尺寸识别不够准确, 导致主车与相近的大型车辆之间, 没有保持安全的横向间距

安全违背类型 7 示例: 如图 8 所示, 在第 1 个场景中, 主车正在左转. 车辆 a 正在从对面车道穿过交叉口, 车辆 b 跟随 a 穿过交叉口. 在主车开始左转之前, 它识别到 a 拥有优先通行权, 并停车让 a 通过. 当 a 通过交叉口时, 主车由于错误地计算了 b 的距离和速度而加速, 导致与 b 发生碰撞. 第 2 个场景的安全违背由主车同样的错误引起.

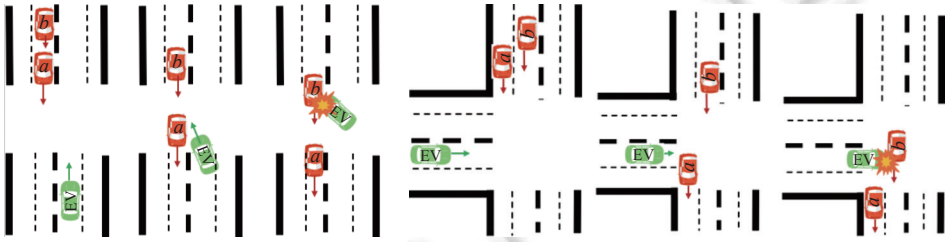


图 8　安全违背类型 7 的两个示例

安全违背类型 10 示例: 如图 9 所示, 主车正在右转. 左侧垂直车道上的卡车 a 正在穿越. 当主车开始转弯时, a 正在靠近主车目的地旁边的车道. 主车减速并靠近右角右转, 与 a 保持安全距离. 然而, 当主车完成转弯后, 主车加速并靠近车道标线, 没有考虑到 a 的体型过大, 导致发生碰撞.

3.5　实验分析-问题 3

为了比较 LEADE 与现有的采用遗传算法查找 Apollo 安全违背的最新技术的效果与效率, 我们在与 LEADE 相同的旧金山道路上运行 MOSAT、AV-Fuzzer 和 BehAVExplor. 为了公平起见, 在每次 24 h 的运行中, 3 种方法

每代中的个体数量相同 (在 5 次实验中, 个体数量分别在 4–8 之间). 需要注意的是, 对于每个发现的安全违背场景, 我们检查安全违背是否由场景中参与者的非法行为引起, 并分析由主车引发的安全违背. 如图 7 所示, 某个安全违背并非由主车引起, 而是从车 d 非法变道并与主车发生碰撞. 对于由主车引起的安全违背, 我们使用相同的标准进行分类. 实验结果见表 8.

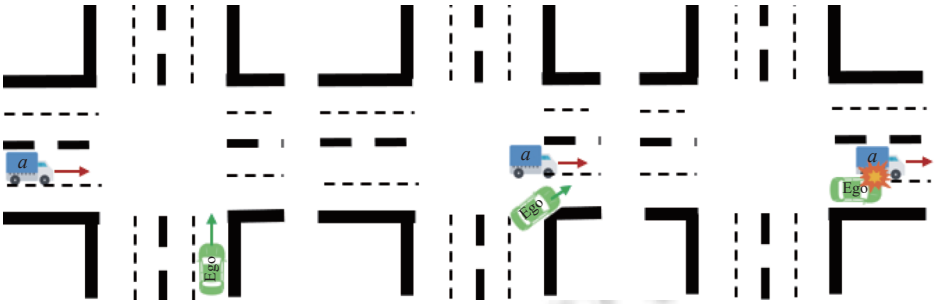


图 9　安全违背类型 10 示例

表 8　LEADE 与对比方法的实验结果

指标	类型	LEADE	MOSAT	AV-Fuzzer	BehAVExplor
发现的安全违背类型数量	—	10	6	3	8
场景生成用时 (s)	最小	21.5	23.2	64	35.4
	最大	25.8	26.9	49.8	44.5
	平均	23	24.4	55	40.8
安全违背场景发现频率	最小	10.1	61	96.4	21
	最大	13.7	65.9	192.6	107
	平均	12	62.1	131	54
发现首个安全违背场景时间 (h)	最小	1	2	89	8
	最大	12	28	122	16
	平均	7	16	98	12
发现所有类型安全违背场景的时间 (h)	最小	11.3	16	15.7	10.6
	最大	13.6	18.9	20.4	13.2
	平均	12.9	18.1	19.2	12.8

在每次 24 h 的运行中, MOSAT 能够找到 6 种 Apollo 的安全违背类型, AV-Fuzzer 能够找到 3 种安全违背类型, BehAVExplor 能够找到 8 种不同的 Apollo 安全违背. MOSAT 平均生成 3 541 个具体场景 (最少 3 208 个, 最多 3 719 个), 其中 130 个场景 (最少 109 个, 最多 148 个) 发生了主车的安全违背. AV-Fuzzer 平均生成 1 572 个具体场景 (最少 1 350 个, 最多 1 734 个), 其中 12 个场景 (最少 9 个, 最多 14 个) 发生了主车的安全违背. BehAVExplor 平均生成 2 160 个具体场景 (最少 1 942 个, 最多 2 441 个), 其中 21 个场景 (最少 6 个, 最多 44 个) 发生了主车的安全违背.

LEADE 发现的 10 种不同的 Apollo 安全违背类型, 在前 14 小时内被揭示. MOSAT 发现的 6 种安全违背类型, 在前 19 h 内揭示. AV-Fuzzer 发现的 3 种安全违背类型, 在前 21 h 内揭示. BehAVExplor 发现的 8 种不同类型的安全违背, 在前 14 h 被揭示. LEADE 发现所有安全违背类型的时间与 BehAVExplor 接近, 比其他两个方法更短; 但 LEADE 相比于 BehAVExplor, 在相同时间内能够发现更多类型的主车安全违背. 此外, LEADE 能够比其他 3 种方法更快地找到第 1 个安全违背. 因此, 与 3 种方法相比, LEADE 能够在更短的时间内找到更多种类的 Apollo 安全违背.

平均而言, LEADE 每生成 12 个具体场景就能发现 1 个 Apollo 的安全违背, BehAVExplor 每生成 54 个具体场景, 发现 1 个 Apollo 的安全违背. 而 MOSAT 需要生成 62 个场景, AV-Fuzzer 需要生成 131 个场景才能发现 1

个安全违背. LEADE 的安全违背暴露频率更高, 这表明 LEADE 能够更高效地生成更多具有安全风险的场景. 在具体场景生成的时间成本上, LEADE 每生成并运行一个具体场景需要 23 s, BehAVExplor 需要 41 s, 而 MOSAT 为 24 s, AV-Fuzzer 为 55 s. LEADE 生成并执行具体场景的时间成本几乎与 MOSAT 相同, 低于 AV-Fuzzer 和 BehAVExplor. 对比结果表明, LEADE 能够更有效地生成更多样化的安全违背场景.

4 总 结

本文提出了一种基于路网建模的自动驾驶系统关键场景生成与自适应演化方法——LEADE. 通过解析用户测试需求, 构建抽象场景; 基于路网建模, 生成具体场景. 运用改进的自适应遗传算法, 包括细粒度评估、自适应选择、自适应变异, LEADE 能够有效地生成多样化的安全关键场景, 从而克服现有方法在生成效率和场景多样性上的局限. 实验结果表明, LEADE 在工业级全栈自动驾驶系统平台上的应用表现优异, 不仅高效地发现了自动驾驶系统中的多种安全违背, 还揭示了当前技术未能覆盖的新类型的安全关键场景. 在生成和执行具体场景时, 场景中的从车和行人必须严格遵守交通法规. 虽然这一约束可能导致 LEADE 遗漏一些由于参与者异常行为导致的安全违背场景, 但考虑到这些异常行为在现实交通中属于极端情况, 缺少实际意义; 此外, 目前自动驾驶系统存在的安全问题较多, 因此当前测试的首要目标是发现由自动驾驶车辆引起的安全违背, 尚未进入测试自动驾驶系统面对极端情况的应急能力的评估阶段. 故而保证测试场景中从车和行人的行为轨迹遵守交通法规, 能够确保发现的安全违背问题, 是由自动驾驶系统导致的. 未来, LEADE 可以进一步扩展以适应更复杂的场景要求, 例如动态交通环境和多模态传感器输入, 从而更全面地支持自动驾驶系统的安全评估与优化.

References

[1] Bertoncello M, Wee D. Ten ways autonomous driving could redefine the automotive world. 2015. https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world#/

[2] Barr ET, Harman M, McMinn P, Shahbaz M, Yoo S. The oracle problem in software testing: A survey. IEEE Trans. on Software Engineering, 2015, 41(5): 507–525. [doi: 10.1109/TSE.2014.2372785]

[3] Zofka MR, Kuhnt F, Kohlhaas R, Rist C, Schamm T, Zöllner JM. Data-driven simulation and parametrization of traffic scenarios for the development of advanced driver assistance systems. In: Proc. of the 18th Int'l Conf. on Information Fusion (FUSION). Washington: IEEE, 2015. 1422–1428.

[4] Singh S, Saini BS. Autonomous cars: Recent developments, challenges, and possible solutions. IOP Conf. Series: Materials Science and Engineering, 2021, 1022(1): 012028. [doi: 10.1088/1757-899X/1022/1/012028]

[5] Chen L, Wu PH, Chitta K, Jaeger B, Geiger A, Li HY. End-to-end autonomous driving: Challenges and frontiers. arXiv:2306.16927, 2024.

[6] Pre-crash scenario typology for crash avoidance research. 2022. https://www.nhtsa.gov/sites/nhtsa.gov/files/pre-crash_scenario_typology-final_pdf_version_5-2-07.pdf

[7] Feng S, Yan XT, Sun HW, Feng YH, Liu HX. Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment. Nature Communications, 2021, 12: 748. [doi: 10.1038/s41467-021-21007-8]

[8] Gambi A, Huynh T, Fraser G. Generating effective test cases for self-driving cars from police reports. In: Proc. of the 27th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Tallinn: ACM, 2019. 257–267. [doi: 10.1145/3338906.3338942]

[9] Bashetty SK, Amor HB, Fainekos G. DeepCrashtest: Turning dashcam videos into virtual crash tests for automated driving systems. In: Proc. of the 2020 IEEE Int'l Conf. on Robotics and Automation (ICRA). Paris: IEEE, 2020. 11353–11360. [doi: 10.1109/ICRA40945.2020.9197053]

[10] Althoff M, Lutz S. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In: Proc. of the 2018 IEEE Intelligent Vehicles Symp. (IV). Changshu: IEEE, 2018. 1326–1333. [doi: 10.1109/IVS.2018.8500374]

[11] Zhang XD, Cai Y. Building critical testing scenarios for autonomous driving from real accidents. In: Proc. of the 32nd ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Seattle: ACM, 2023. 462–474. [doi: 10.1145/3597926.3598070]

[12] Harman M. Automated test data generation using search based software engineering. In: Proc. of the 2nd Int'l Workshop on Automation of Software Test (AST 2007). Minneapolis: IEEE, 2007. 2. [doi: 10.1109/AST.2007.4]

[13] Harman M, Mansouri SA, Zhang YY. Search-based software engineering: Trends, techniques and applications. ACM Computing Surveys (CSUR), 2012, 45(1): 11. [doi: 10.1145/2379776.2379787]

[14] McMinn P. Search-based software test data generation: A survey. Software Testing, Verification and Reliability, 2004, 14(2): 105–156. [doi: 10.1002/stvr.294]

[15] Thibeault Q, Anderson J, Chandratre A, Pedrielli G, Fainekos G. PSY-TaLiRo: A Python toolbox for search-based test generation for cyber-physical systems. arXiv:2106.02200, 2021.

[16] Guo A, Feng Y, Cheng YZ, Chen ZY. Semantic-guided fuzzing for virtual testing of autonomous driving systems. Journal of Systems and Software, 2024, 212: 112017. [doi: 10.1016/j.jss.2024.112017]

[17] Koren M, Alsaif S, Lee R, Kochenderfer MJ. Adaptive stress testing for autonomous vehicles. In: Proc. of the 2018 IEEE Intelligent Vehicles Symp. (IV). Changshu: IEEE, 2018. 1–7. [doi: 10.1109/IVS.2018.8500400]

[18] Ben Abdessalem R, Nejati S, Briand LC, Stifter T. Testing advanced driver assistance systems using multi-objective search and neural networks. In: Proc. of the 31st IEEE/ACM Int'l Conf. on Automated Software Engineering. Singapore: ACM, 2016. 63–74. [doi: 10.1145/2970276.2970311]

[19] Tian HX, Wu GQ, Yan JR, Jiang Y, Wei J, Chen W, Li S, Ye D. Generating critical test scenarios for autonomous driving systems via influential behavior patterns. In: Proc. of the 37th IEEE/ACM Int'l Conf. on Automated Software Engineering. Rochester: ACM, 2022. 46. [doi: 10.1145/3551349.3560430]

[20] Onieva E, Hernández-Jayo U, Osaba E, Perallos A, Zhang X. A multi-objective evolutionary algorithm for the tuning of fuzzy rule bases for uncoordinated intersections in autonomous driving. Information Sciences, 2015, 321: 14–30. [doi: 10.1016/j.ins.2015.05.036]

[21] Klück F, Zimmermann M, Wotawa F, Nica M. Genetic algorithm-based test parameter optimization for ADAS system testing. In: Proc. of the 19th IEEE Int'l Conf. on Software Quality, Reliability and Security (QRS). Sofia: IEEE, 2019. 418–425. [doi: 10.1109/QRS.2019.00058]

[22] Li GP, Li YR, Jha S, Tsai T, Sullivan M, Hari SKS, Kalbarczyk Z, Iyer R. AV-Fuzzer: Finding safety violations in autonomous driving systems. In: Proc. of the 31st IEEE Int'l Symp. on Software Reliability Engineering (ISSRE). Coimbra: IEEE, 2020. 25–36. [doi: 10.1109/ISSRE5003.2020.00012]

[23] Tian HX, Jiang Y, Wu GQ, Yan JR, Wei J, Chen W, Li S, Ye D. MOSAT: Finding safety violations of autonomous driving systems using multi-objective genetic algorithm. In: Proc. of the 30th ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Singapore: ACM, 2022. 94–106. [doi: 10.1145/3540250.3549100]

[24] Fremont DJ, Dreossi T, Ghosh S, Yue XY, Sangiovanni-Vincentelli AL, Seshia SA. Scenic: A language for scenario specification and scene generation. In: Proc. of the 2019 ACM SIGPLAN Conf. on Programming Language Design and Implementation. Phoenix: ACM, 2019. 63–78. [doi: 10.1145/3314221.3314633]

[25] Guo A, Zhou Y, Tian HX, Fang CR, Sun YJ, Sun WS, Gao XY, Luu AT, Liu Y, Chen ZY. SoVAR: Build generalizable scenarios from accident reports for autonomous driving testing. In: Proc. of the 39th IEEE/ACM Int'l Conf. on Automated Software Engineering. Sacramento: ACM, 2024. 268–280. [doi: 10.1145/3691620.3695037]

[26] An open autonomous driving platform. 2013. https://github.com/ApolloAuto/apollo

[27] Navigant research names Waymo, ford autonomous vehicles, cruise, and Baidu the leading developers of automated driving systems. 2020. https://www.businesswire.com/news/home/20200407005119/en/Navigant-Research-Names-Waymo-Ford-Autonomous-Vehicles

[28] Hersey F. Baidu launches their open platform for autonomous cars—And we got to test it. 2017. https://technode.com/2017/07/05/baidu-apollo-1-0-autonomous-cars-we-test-it/

[29] Autoware self-driving vehicle on a highway. 2025. https://www.youtube.com/watch?v=npQMzH3jd8

[30] Baidu launches public robotaxi trial operation. 2019. https://www.globenewswire.com/news-release/2019/09/26/1921380/0/en/Baidu-Launches-Public-Robotaxi-Trial-Operation.html

[31] Menzel T, Bagschik G, Maurer M. Scenarios for development, test and validation of automated vehicles. In: Proc. of the 2018 IEEE Intelligent Vehicles Symp. (IV). Changshu: IEEE, 2018. 1821–1827. [doi: 10.1109/IVS.2018.8500406]

[32] Shin SY, Nejati S, Sabetzadeh M, Briand LC, Zimmer F. Test case prioritization for acceptance testing of cyber physical systems: A multi-objective search-based approach. In: Proc. of the 27th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Amsterdam: ACM, 2018. 49–60. [doi: 10.1145/3213846.3213852]

[33] Nejati S. Testing cyber-physical systems via evolutionary algorithms and machine learning. In: Proc. of the 12th IEEE/ACM Int'l Workshop on Search-based Software Testing (SBST). Montreal: IEEE, 2019. 1. [doi: 10.1109/SBST.2019.00008]

[34] Arrieta A, Wang S, Markiegi U, Sagardui G, Etxeberria L. Search-based test case generation for cyber-physical systems. In: Proc. of the 2017 IEEE Congress on Evolutionary Computation (CEC). Donostia: IEEE, 2017. 688–697. [doi: 10.1109/CEC.2017.7969377]

[35] Wang S, Ali S, Yue T, Li Y, Liaaen M. A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering. In: Proc. of the 38th IEEE/ACM Int'l Conf. on Software Engineering (ICSE). Austin: IEEE, 2016. 631–642. [doi: 10.1145/2884781.2884880]

[36] Hekmatnejad M, Hoxha B, Fainekos G. Search-based test-case generation by monitoring responsibility safety rules. In: Proc. of the 23rd IEEE Int'l Conf. on Intelligent Transportation Systems (ITSC). Rhodes: IEEE, 2020. 1–8. [doi: 10.1109/ITSC45102.2020.9294489]

[37] Cao YM, Thibeault Q, Chandratre A, Fainekos G, Pedrielli G, Castillo-Effen M. Work-in-progress: Towards assurance case evidence generation through search based testing. In: Proc. of the 2021 Int'l Conf. on Embedded Software (EMSOFT). Austin: IEEE, 2021. 41–42.

[38] Tang SC, Zhang ZY, Zhou JX, Lei L, Zhou Y, Xue YX. LeGEND: A top-down approach to scenario generation of autonomous driving systems assisted by large language models. In: Proc. of the 39th IEEE/ACM Int'l Conf. on Automated Software Engineering. Sacramento: ACM, 2024. 1497–1508. [doi: 10.1145/3691620.3695520]

[39] Almaskati D, Kermanshachi S, Pamidimukkula A. Autonomous vehicles and traffic accidents. Transportation Research Procedia, 2023, 73: 321–328. [doi: 10.1016/j.trpro.2023.11.924]

[40] Abdessalem RB, Panichella A, Nejati S, Briand LC, Stifter T. Testing autonomous cars for feature interaction failures using many-objective search. In: Proc. of the 33rd IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE). Montpellier: IEEE, 2018. 143–154. [doi: 10.1145/3238147.3238192]

[41] Jullien JM, Martel C, Vignollet L, Wentland M. OpenScenario: A flexible integrated environment to develop educational activities based on pedagogical scenarios. In: Proc. of the 9th IEEE Int'l Conf. on Advanced Learning Technologies. Riga: IEEE, 2009. 509–513. [doi: 10.1109/ICALT.2009.24]

[42] ASAM OpenScenario®. 2025. https://www.asam.net/standards/detail/openscenario/v200/

[43] Rasouli A, Kotseruba I, Tsotsos JK. Are they going to cross? A benchmark dataset and baseline for pedestrian crosswalk behavior. In: Proc. of the 2017 IEEE Int'l Conf. on Computer Vision Workshops. Venice: IEEE, 2017. 206–213. [doi: 10.1109/ICCVW.2017.33]

[44] Zhou Y, Sun Y, Tang Y, Chen YQ, Sun J, Poskitt CM, Liu Y, Yang ZJ. Specification-based autonomous driving system testing. IEEE Trans. on Software Engineering, 2023, 49(6): 3391–3410. [doi: 10.1109/TSE.2023.3254142]

[45] Haupt RL. Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors. In: Proc. of the 2000 IEEE Antennas and Propagation Society Int'l Symp. Salt Lake City: IEEE, 2000. 1034–1037. [doi: 10.1109/APS.2000.875398]

[46] Cheng MF, Zhou Y, Xie XF. BehAVExplor: Behavior diversity guided testing for autonomous driving systems. In: Proc. of the 32nd ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Seattle: ACM, 2023. 488–500. [doi: 10.1145/3597926.3598072]

作者简介

田浩翔, 博士, CCF 学生会员, 主要研究领域为自动驾驶系统, 仿真测试, 软件安全.

吴国全, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为软件工程, 软件测试与维护, 服务计算.

魏峻, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为软件工程, 网络分布式计算, 系统安全与可靠性.

郭安, 博士, 主要研究领域为自动驾驶测试, 智能软件工程.

韩星烁, 博士, 教授, 博士生导师, 主要研究领域为人工智能安全, 自动驾驶攻防, 软件系统安全.

陈伟, 博士, 研究员, 博士生导师, CCF 专业会员, 主要研究领域为智能软件工程, 服务计算, 网络分布式计算.

王伟, 博士, 研究员, 博士生导师, CCF 高级会员, 主要研究领域为分布式系统软件.

叶丹, 博士, 研究员, 博士生导师, 主要研究领域为网络分布式系统, 软件工程.