计算机科学
COMPUTER SCIENCE

**GCE3S:基于进化搜索的自动驾驶安全关键场景生成方法**

孙乐乐, 黄松, 郑长友, 夏春艳, 阳真

引用本文

孙乐乐, 黄松, 郑长友, 夏春艳, 阳真. GCE3S:基于进化搜索的自动驾驶安全关键场景生成方法[J]. 计算机科学, 2025, 52(10): 275-286.

SUN Lele, HUANG Song, ZHENG Changyou, XIA Chunyan, YANG Zhen. GCE3S:A Method for Generating Safety-critical Scenarios in Autonomous Driving Based on Evolutionary Search [J]. Computer Science, 2025, 52(10): 275-286.

---

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240800030

# GCE3S:基于进化搜索的自动驾驶安全关键场景生成方法

孙乐乐　黄　松　郑长友　夏春艳　阳　真
中国人民解放军陆军工程大学指挥控制工程学院　南京 210007
(sunlele@aeu.edu.cn)

**摘　要**　自动驾驶技术的快速发展为交通出行带来了巨大的变革潜力,但在现实交通环境中,自动驾驶车辆的安全违规行为会导致巨大的损失。为了确保自动驾驶系统能够在各种复杂的交通环境中安全运行,在部署到实际道路之前必须对其进行充分的测试。由于自动驾驶测试场景空间的复杂性和高维性,现有安全关键场景生成方法存在成本高昂、效率低等问题。因此,提出了一种基于进化搜索的自动驾驶安全关键场景生成方法——GCE3S。GCE3S将场景中的障碍物及其属性映射为基因组成的染色体结构,从而对障碍物(车辆、天气、行人等)进行更加细致的扰动,构建具有对抗性的安全关键场景,并通过多个目标函数引导进化搜索算法生成多样化的安全关键场景。此外,在工业级自动驾驶系统百度Apollo和LGSVL模拟环境中对GCE3S进行了对比实验,实验结果表明,在相同的时间内,GCE3S生成的安全关键场景数量相较于最好的基准MOSAT方法提升了20.4%,生成的安全关键场景在多样性上增加了20%。

**关键词**:自动驾驶;仿真测试;测试场景;安全关键场景生成;多目标进化搜索
**中图分类号**　TP311

# GCE3S:A Method for Generating Safety-critical Scenarios in Autonomous Driving Based on Evolutionary Search

SUN Lele,HUANG Song,ZHENG Changyou,XIA Chunyan and YANG Zhen
College of Command and Control Engineering,Army Engineering University of PLA,Nanjing 210007,China

**Abstract**　The rapid development of automated driving technology has brought great potential for transforming mobility,but automated driving technology,as safety-critical software,will lead to huge losses due to safety violations of self-driving vehicles in real traffic environments. In order to ensure that autonomous driving systems can operate safely in various complex traffic environments,autonomous driving systems must be fully tested before being deployed on real roads. Due to the complexity and high dimensionality of the autonomous driving test scenario space,existing safety critical scenario generation methods suffer from high cost and low efficiency. Therefore,this paper proposes an evolutionary search-based safety-critical scenario generation method for autonomous driving－GCE3S. GCE3S constructs safety-critical scenarios with adversarial nature by mapping the obstacles and their attributes in the scenario to the chromosome structure of genetic composition,thus perturbing the obstacles(vehicles,weather,pedestrians,etc.) in a more detailed manner and guiding the evolutionary search algorithms through multiple objective functions to generate diverse safety critical scenarios. In addition,the GCE3S is experimentally evaluated in the simulated environments of Baidu Apollo,an industrial-grade autonomous driving system,and LGSVL. The experimental results show that the number of safety-critical scenarios generated by GCE3S improve by 20.4% and the generated safety-critical scenarios increase by 20% in terms of diversity in the same amount of time as compared to the best baseline MOSAT method.

**Keywords**　Autonomous driving,Simulation testing,Test scenario,Safety critical scenario generation,Multi-objective evolutionary search

## 1　引言

随着人工智能技术和各类高精度传感器的飞速发展以及国家政策的大力支持[1],近年来,自动驾驶技术取得了显著的进步。各大汽车厂商、科技公司和研究机构都在积极研发自动驾驶技术,如国外特斯拉旗下的Autopilot[2]和谷歌旗下的Waymo Driver[3],国内华为旗下的乾崑ADS 3.0[4]和百度旗下的Apollo[5]。然而,要确保自动驾驶汽车的安全性和可靠性,需要在各种复杂的交通场景中对自动驾驶系统进行验证,以应对不同的道路条件、交通规则和其他车辆的行为。

目前自动驾驶测试主要分为实车测试和仿真测试两种方法。实车测试是在真实道路环境中进行的测试,能够直接验证自动驾驶系统的性能和安全性。然而,实车测试成本高昂,需要耗费大量的时间和资源,且受到实际道路条件的限制,无

到稿日期:2024-08-05　返修日期:2024-11-26
通信作者:黄松(hs0317@163.com)

法覆盖所有安全关键场景。仿真测试则是在虚拟环境中进行的测试，通过模拟各种交通场景对自动驾驶系统进行评估[6]，其具有高效、可重复、成本低等优点，能够快速生成大量的测试场景，发现自动驾驶系统潜在的问题和风险。因此自动驾驶仿真测试是验证和评估自动驾驶系统安全性和可靠性的关键一环。在自动驾驶仿真测试中，测试场景应覆盖各种可能的交通场景和驾驶情况，包括常见的道路场景、特殊的天气条件和复杂的交通流等。通过生成多样化的测试场景来全面检验自动驾驶系统的感知、规划、决策和控制能力，确保其在各种情况下都能够安全可靠地运行[7]。综上所述，自动驾驶测试是确保自动驾驶技术安全性和可靠性的关键环节，而测试场景的生成是自动驾驶测试的核心，因此，如何快速高效地生成具有代表性和多样性的安全关键场景是至关重要的。

目前一部分研究是利用现实交通数据来构建测试场景[8-12]，通过收集和分析真实世界中自动驾驶车辆的交通数据，如车辆轨迹、交通信号和行人行为等，构建与真实世界类似的安全关键场景。然而，获取真实的交通数据并进行准确的标注是一项成本高昂且繁琐的工作，需要耗费大量的人力成本和时间。此外，由于真实环境下安全关键场景发生的概率很低，且现实交通数据具有多样性和复杂性等特点，仅依靠收集的现实交通数据无法覆盖所有的安全关键场景，从而影响自动驾驶系统的安全性和可靠性评估。

很多学者采用优化搜索技术搜索自动驾驶系统的安全关键场景[13-21]。例如，Li 等[14]提出了通过扰动交通参与者的架势操作来生成自动驾驶系统的安全关键场景方法 AV-FUZZER，AV-FUZZER 利用车辆动力学和遗传算法来搜索自动驾驶的安全关键场景。Feng 等[15]提出了一个通用框架以解决在不同操作设计域、CAV(Connected and Automated Vehicle)模型和性能指标的情况下的测试场景生成问题，通过考虑机动挑战和暴露频率两个因素来评估测试场景的重要性，然后使用优化搜索技术和种子填充技术来搜索安全关键场景。Luo 等[18]提出了一种名为 EMOOD 的测试场景生成方法，EMOOD 使用优先级技术将所有可能的模式进行排序，然后使用多目标优化算法迭代搜索安全关键场景，在每次迭代中，通过动态优先级技术确定目标模式，并给予那些具有更高重要性的模式更高的优先级。大部分研究将测试场景的输入定义为多个变量组成的向量，并利用各种算法来探索输入空间，从而生成自动驾驶系统的测试场景。然而，自动驾驶系统的输入空间具有高维性和复杂性，导致以上方法在不断变化的交通环境中搜索到安全关键场景的概率低。此外，基于优化搜索的技术设定的初始场景通常较为简单，不能探索到更加复杂的安全关键场景。

综上所述，自动驾驶测试是确保自动驾驶技术安全性和可靠性的关键环节，而测试场景的生成则是自动驾驶测试的核心，因此，如何快速高效地生成具有代表性和多样性的安全关键场景是本文研究的宗旨。本文提出了一种新颖的多目标进化搜索算法 GCE3S(Generating Safety-Critical Scenarios in Autonomous Driving Based on Evolutionary Search)，GCE3S 通过将测试场景中的障碍物及其属性建模为基因组成的染色体，从而对障碍物(车辆、天气、行人等)进行更加细致的扰动，构建具有对抗性的安全关键场景，并通过多个目标函数引导进化搜索算法生成多样化的安全关键场景，从而揭露自动驾驶系统的各种安全违规行为。在工业级自动驾驶系统百度 Apollo 和 LGSVL 模拟环境中对 GCE3S 进行了评估。

本文的主要贡献如下：

1)为了更好地对自动驾驶车辆进行扰动，将场景中的障碍物及其属性映射为基因组成的染色体结构，从而对其进行更加细致化的扰动；

2)采用多个目标函数对其指导搜索，包括最小碰撞距离、参与者行为对 EGO 的干扰程度以及场景的多样性；

3)在工业级自动驾驶系统百度 Apollo 上部署了 GCE3S，实验结果表明，在相同的时间内，GCE3S 生成的安全违规场景数量相较于基准 MOSAT 方法提升了 20.4%，生成的安全违规场景类型在多样性上增加了 20%。

## 2 背景与问题定义

### 2.1 自动驾驶系统

随着自动驾驶技术的飞速发展，亟需标准化的框架规范和指导自动驾驶的技术创新与应用。为了应对这一需求，各国政府和标准化组织相继推出了自动驾驶级别的划分标准。2020 年 3 月 9 日，中国工业和信息化部(Ministry of Industry and Information Technology, MIIT)发布了《汽车驾驶自动化分级》推荐性国家标准报批稿[22]，如表 1 所列，该标准根据自动驾驶系统的设计运行范围限制等六要素，将自动驾驶技术划分为 0 级至 5 级的 6 个级别[23]。这一分级标准的确定不仅体现在技术层面的规范和指导，更在于其对整个汽车产业生态和价值链体系的潜在重塑作用。中国、美国、欧洲和日本等国家地区均已将自动驾驶技术定位为交通领域的重点发展方向，并在国家战略层面进行了前瞻性的规划和布局。

目前自动驾驶技术已经开始由 L1，L2 级向 L3，L4 级发展，2023 年 11 月 17 日，L3 级自动驾驶迈出了落地应用的关键一步，工信部等 4 个部门联合发布《关于开展智能网联汽车准入和上路通行试点工作的通知》[24]，其中明确提出在智能网联汽车道路测试与示范应用基础上，遴选具备量产条件的 L3 及 L4 级别的自动驾驶汽车开展准入试点，我国首批确定由 9 个汽车生产企业和 9 个使用主体组成的联合体，将在北京、上海、广州等 7 个城市展开智能网联汽车准入和上路通行试点，试点产品涵盖乘用车、客车以及货车三大类。截至 2024 年 4 月底，我国共开放智能网联汽车测试道路 29 000 多公里，发放测试示范牌照 6 800 多张，道路测试总里程超过 8 800 万公里[25]。

特斯拉的 Autopilot 系统可以实现自动辅助导航驾驶、自动辅助转向和智能召唤等功能，并计划在今后实现更高级别的自动驾驶，最终达到 L5 级完全自动驾驶。谷歌 Waymo 公司是自动驾驶技术的领先者之一，它已经在限定区域内实现了 L4 级自动驾驶。国内车企方面，百度的 Apollo 自动驾驶

系统通过复杂传感器套装实现自动驾驶功能。随着百度全面推进自动驾驶规模化应用，“萝卜快跑”已在10多个城市开展常态化出行服务，累计出行订单量超过330万单，其中北京、武汉、重庆、深圳4个城市的用户可以用“萝卜快跑”App方便地叫到车内无安全员的Robotaxi，同时百度也在稳步提升自动驾驶等级。华为也入局自动驾驶领域，推出了华为乾崑智驾ADS 3.0高阶智能驾驶系统。华为ADS 3.0是一项接近于L3级别的自动驾驶技术，采用了多传感器结合的技术，通过摄像头、毫米波雷达和超声波等传感器获取道路和周围环境的信息，率先实现了不依赖于高精地图的高速、城区高阶智能驾驶，可以应对更多复杂的驾驶场景，提升驾驶安全性。比亚迪、小鹏等都在稳步推进自动驾驶技术研发，广汽日产、上汽大众等传统车企也在积极布局智能网联汽车。

综上所述，目前各大车企实现的自动驾驶技术大多处于L1至L2＋的水平，还达不到真正意义上的自动驾驶。但随着激光雷达、高精地图以及端到端模型等技术的进步，L3级乃至更高级别的自动驾驶指日可待。总体来看，汽车智能化正在稳步发展，但要实现完全自动驾驶还有很长的路要走。汽车厂商需要在技术积累和法规允许的前提下，逐步提升自动驾驶等级，最终实现无人驾驶。

表1　自动驾驶级别划分

Table 1　Classification of autonomous driving level

| 级别 | 名称 | 含义 | 持续的车辆横向和纵向运动控制 | 目标和事件探测与响应 |
| --- | --- | --- | --- | --- |
| L0级 | 应急辅助 | 所有操作都需要驾驶员执行，但可以配备一些辅助驾驶的保护系统，如刹车辅助、盲区监测等 | 驾驶员 | 驾驶员、系统 |
| L1级 | 部分驾驶辅助 | 拥有辅助系统，如防抱死系统、车身电子稳定系统、定速巡航等 | 驾驶员、系统 | 驾驶员、系统 |
| L2级 | 组合驾驶辅助 | 可以自动完成某些驾驶任务，如自适应巡航和车道保持辅助等 | 系统 | 驾驶员、系统 |
| L3级 | 有条件自动驾驶 | 在特定条件下完全控制驾驶任务，但需要驾驶员在必要时接管 | 系统 | 系统 |
| L4级 | 高度自动驾驶 | 在特定的环境和条件下完全自主执行驾驶任务，无需驾驶员干预 | 系统 | 系统 |
| L5级 | 完全自动驾驶 | 在任何环境、任何道路都能由自动驾驶系统进行控制，驾驶员可以完全不参与驾驶 | 系统 | 系统 |

### 2.2　仿真测试

自动驾驶仿真测试通过虚拟环境来模拟各种驾驶场景，以评估自动驾驶系统[26]。自动驾驶仿真测试凭借其测试场景丰富、计算速度快、测试效率高、资源消耗低、可重复性好、可嵌入汽车开发的各个环节等优势，能够很好地在实际路面行驶之前，对车辆进行全面、安全的测试。仿真测试可以提高自动驾驶系统的安全性和可靠性，减少实际测试的风险和成本，加速自动驾驶技术的发展和应用。

自动驾驶仿真测试作为一种高效、安全、低成本的测试方法，会针对自动驾驶系统的功能、性能、安全性、可靠性等进行全面、充分的测试，以保证自动驾驶汽车能够达到路面行驶的要求。自动驾驶汽车的功能测试是测试自动驾驶汽车是否能够按照预期执行各种功能，如遵守交通规则、感知道路标志和障碍物等。Zhu等[27]将功能测试分为感知模块测试、决策模块测试、综合模块功能测试及整车测试。除了功能测试以外，一些其他的测试类型也被广泛应用于自动驾驶仿真测试中，如安全测试、性能测试等。安全测试主要用于测试自动驾驶汽车在紧急情况下的反应能力和安全性。安全测试的目的是检查自动驾驶汽车的安全性，例如，在急刹车或避让等威胁场景中是否能安全应对。

需要注意的是，自动驾驶汽车的测试不仅是仿真测试，还需要在实际路况下进行测试。实际测试可以验证仿真测试的结果，同时也可以发现一些仿真测试无法模拟的特殊情况。因此，自动驾驶汽车的测试需要结合仿真测试和实际测试，以保证测试结果的准确性和可靠性。

## 3　研究方法

### 3.1　场景的定义与建模

在自动驾驶仿真测试中，场景指在特定时间和空间内，自动驾驶车辆与其周围环境交互的描述，一个场景可以通过一系列可配置的静态属性和动态特征来描述。静态属性设定场景的静态对象，如地图、天气、一天中的时间等；动态特征定义场景中的交通流情况，涵盖了非玩家角色（Non-Player Character，NPC）、车辆的状态（如位置、方向等）、行驶轨迹和行为模式（如速度变化、加速度变化等）等。如表2所列，场景中主要有8类对象，即自我车辆、NPC车辆、行人、障碍物、时间、天气、道路、交通标志。场景是一个元组$(e,n,p,o,t,w,r,c)$，其中每个组件都是相应对象的一系列操作，这些操作取决于交通场景的相应对象。具体来说，$e$是自我车辆的操作序列，如设置起点、终点；$n$是NPC车辆的操作序列，如设置起始速度或起点位置；$p$是行人的操作序列，其设置与NPC类似；$o$是障碍物的操作序列，如设置障碍物的大小或位置；$t$是时间的操作序列，设置小时或分钟，$w$是天气的操作序列，如设置雨、雪等天气；$r$是道路的操作序列，如设置地图为旧金山地图；$c$是交通标志的操作序列，如设置限速、交通灯等标志。一般来说，一个场景中有多个NPC车辆、行人和障碍物。面对不同的测试目的，研究人员将重点放在一些特定属性上。例如，评估感知模块在不同天气条件下的鲁棒性时，研究人员更关注不同天气的设置，如雨和雾；评估ADS的安全性时，研究人员更关注NPC车辆的轨迹和行为配置。

表 2 场景对象

Table 2 Scene objects

| 场景对象 | 场景属性 | 场景描述 |
|---|---|---|
| Ego($e$) | {position, destination} | 自动驾驶车辆 |
| Npc($n$) | {type, goal, position} | 其他车辆 |
| $P$ | {type, goal, position} | 行人 |
| $O$ | {type, position} | 障碍物 |
| $T$ | time | 时间 |
| $W$ | {Rain, fog, wetness, cloudiness, damage} | 天气 |
| Map($m$) | MapName | 道路 |
| $c$ | {type, position} | 交通标志 |

图 1 为 json 编写的场景脚本示例。

```
1. "ego":{
2.    "position":{
3.       "x":90.6,
4.       "y":100.2,
5.       "z":60.5 },
6.    "destination":{
7.       "method":"forward_right",
8.       "value":{
9.          "v1":225,
10.         "v2":-1,
11.         "v3":0 }
12.   }
13. },
14. "npcs":[
15.   {
16.      "type":"Sedan",
17.      "goal":"mutated",
18.      "position":{
19.         "x":90.1,
20.         "y":90.2,
21.         "z":50.2
22.   },
23. {
24.      "type":"SUV",
25.      "goal":"mutated",
26.      "position":{
27.         "x":100.1,
28.         "y":109.2,
29.          "z":90.4
30.   }
31. ],
32. "environment":{
33.   "rain":0,
34.   "fog":0,
35.   "wetness":0,
36.   "cloudiness":0,
37.   "damage":0
38. },
39. "time":0,
40. "mapName":"SanFrancisco_correct"
```

图 1 场景脚本

Fig. 1 Scene Script

该脚本定义的 $n$ 是由两辆 NPC 车辆组成的场景。具体来说，第 1－13 行制定了自动驾驶车辆的起点位置和终点位置；第 14－31 行制定了两辆 NPC 车辆的类型和起点位置，NPC 车辆的类型可以设置为模拟器支持的{BoxTruck，Hatchback，Jeep，SUV，SchoolBus，Sedan}集合中的任意值；第 32－38 行制定了雨、雾、湿度、云等天气情况；第 39 行说明了仿真模拟运行的时刻；第 40 行说明了地图的类型。所有语句构成了一个测试场景，该场景可以通过不同的方式发生变异，如 NPC 车辆的起点位置和终点位置、天气的情况，以及 NPC 车辆的速度和加速度。

在测试场景中，与被测自动驾驶系统连接的车辆称为自我车辆（EGO），其他车辆称为非玩家角色车辆（NPC）。GCE3S 通过在不断变化的环境中干扰 NPC 车辆的驾驶操作，来寻找 EGO 车辆发生安全违规的行为，从而生成安全关键场景。本文将对 NPC 车辆扰动的搜索视为优化搜索问题，通过进化搜索算法进行求解。在 GCE3S 中，每个测试场景对应进化搜索算法中的一个个体，图 2 展示了场景的基因建模表示。

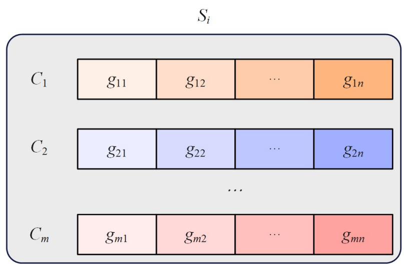



图 2 场景 $S_i$ 的基因建模表示

Fig. 2 Gene modelling representation of scene $S_i$

每个个体（测试场景）由一个或多个染色体 $S_i=\{C_1, C_2, \cdots, C_m\}$组成。$C_i$ 为第 $i$ 个染色体，$m$ 为场景中染色体的数量。将 NPC 驾驶操作序列编码为一个染色体 $C_i=\{g_{i1}, g_{i2}, \cdots, g_{in}\}$，它由一个个基因组成，在执行过程中，根据 EGO 与 NPC 车辆的相对位置，将驾驶操作序列转换为基因序列。

### 3.2 方法框架

GCE3S 的目标是通过高效生成对抗性的、多样化的安全关键场景来发现自动驾驶系统的交通违规行为。在自动驾驶中，安全关键场景指在模拟或实际道路条件下，自动驾驶系统可能面临对安全至关重要的复杂、紧急或异常的测试场景。如图 3 所示，本文采用基于进化搜索算法来生成自动驾驶交通违规的安全关键场景。首先，定义一个初始场景，设定初始参数，然后将初始场景作为进化搜索算法的初始种群。在进化过程中，根据场景的适应度分数，通过选择、交叉、变异等操作进行变换，计算适应度分数，保留导致安全违规的可能性较大的场景。重复此过程，直到发现自动驾驶汽车安全违规行为的安全关键场景。当适应度分数没有随着时间的推移而提高时，则认为目前的搜索达到一个局部极值点。此外，本文设计了一个重新搜索的机制，当搜索进入一个局部机制区域时，系统会随机生成一个与进化搜索算法生成的过去场景最不类似的场景，来确保进化搜索算法能够走出局部极值点，这种方式可以有效地发现不同类型的安全关键场景。

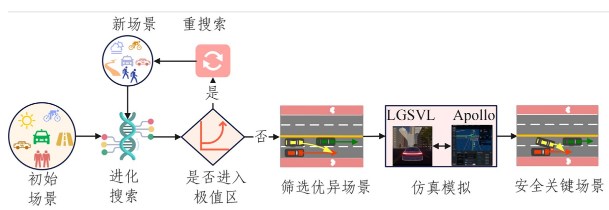



图 3 GCE3S 方法框架

Fig. 3 Framework of GCE3S method

GCE3S 方法的主要算法如算法 1 所示。它以一组初始种子场景 $S_0$ 和一个目标自动驾驶系统 $A$ 作为输入，输出测试场景 $T$、安全关键场景 $C$ 以及一些可配置参数 $P$。

**算法 1** GCE3S Algorithm

```
Input: Initial scenario S0, autonomous driving System A, parameter set P
Result: Test Scenarios T, Safety critical scenarios C
1. S← {}, T ← {}, C ← {}
2. IntiEnvironment(P)
3. for i in range(0, P. pop_size) do
4.     S′ ← randomMutation(S0)
5.     result ← Simulator(S′, A)
6.     S′. fitness ← getfitness(S′)
7.     S ← S ∪ S′
8.     if result. isViolation() then
9.         C ← C ∪ S′
10.    end if
11. end for
12. Repeat
13.    Sc ← multPointCrossover(S)
14.    Sm ← adptiveMutation(S)
15.    for i in range(0, Sm. pop) do
16.        result ← Simulator(Sm. pop[i], A)
17.        Sm. pop[i]. fitness ← getfitness(Sm. pop[i])
18.        if result. isViolation() then
19.            C ← C ∪ Sm. pop[i]
20.        end if
21.    end for
22.    S ← selectTournament(Sm)
23.    if S. isNoProcess() then
24.        restart(S)
25.    end if
26. until given time budget expires
27. return T, C
```

1）初始化：GCE3S 框架首先需要人工构建一个初始场景 $S_0$ 并设置参数 $P$，参数 $P$ 主要包括种群大小 *pop_size*、NPC 数量 *npc_size*、速度与加速度范围 *bounds*、变异算子 *mutation*、交叉算子 *crossover*，以及最大迭代数 *max_gen* 等；其次，初始化所需要的变量以及仿真环境（第 1—2 行），并根据初始场景 $S_0$ 和设置参数 $P$ 进行种群 $S$ 的初始化操作（第 3—11 行）；然后，对初始场景 $S_0$ 随机变异产生一个新的场景 $S'$（第 4 行），将场景 $S'$ 输入仿真环境中运行，得到其运行结果（第 5 行）；最后，计算其相应的适应度分数值，并将其加入种群 $S$ 中（第 6—7 行），如果场景 $S'$ 发生交通违规，则将其加入安全关键场景集 $C$ 中（第 8—10 行）。

2）进化搜索：在进化搜索过程中，GCE3S 重复执行进化搜索引导的安全关键场景生成，直到运行时间达到给定时间（第 12—26 行）。在每一次迭代中，GCE3S 首先对种群 $S$ 进行多点交叉操作，从而得到交叉后的种群 $S_c$（第 13 行）；其次，通过自适应变异得到种群 $S_m$（第 14 行）；然后，对变异后的种群 $S_m$ 中的每个场景进行仿真模拟（第 15—21 行），得到其仿真模拟结果 *result*，并计算种群中每个场景的适应度分数值 $S_m.pop[i].fitness$（第 16—17 行），如果场景 $S_m.pop[i]$ 发生安全违规，则将其加入安全关键场景集 $C$（第 18—20 行）；最后，根据锦标赛选择从种群 $S_m$ 中重新构建下一轮迭代的种群 $S$（第 22 行），并计算种群 $S$ 的适应度分数是否随着时间的推移而变大，若不是，则认为当前进化搜索达到局部极值区域，重新构建一个与过去不相似的场景 $S_0'$ 并开始新一轮的进化搜索（第 23—25 行）。

3）结束：实验结束之后，会返回生成的测试场景集 $T$ 及安全关键场景集 $C$（第 27 行）。

### 3.3 种群初始化

本文的目的是寻找导致自动驾驶系统产生安全违规的场景。在进化搜索算法的应用过程中，首先需要解决的问题是种群初始化，在测试场景解空间中根据初始场景 $S_0$ 进行随机变异，产生新的场景 $S'$，并将其输入仿真环境中模拟运行，运行结束之后，计算其适应度分数，并将其加入初始种群 $S$ 中，如果场景 $S'$ 发生安全违规，则将其加入安全关键场景 $C$ 中，经过 *pop_size* 轮迭代，形成最终的初始种群 $S$。在确定了自动驾驶车辆的初始位置后，每个测试场景通过参数化描述交通参与者的初始位置、速度和加速度。在进化搜索算法的解集中，用向量组 $\boldsymbol{S}_i=\{C_1,C_2,\cdots,C_m\}$ 来表示一个测试场景，其中，$C_i$ 为第 $i$ 个染色体，表示测试场景的第 $i$ 个障碍物，如车辆、行人等；$m$ 为场景中染色体的数量。每个染色体被描述为一个向量 $\boldsymbol{C}_i=\{g_{i1},g_{i2},\cdots,g_{in}\}$，其中，$g_{ij}$ 为第 $i$ 个障碍物的第 $j$ 个属性值，如位置、速度等；$n$ 为第 $i$ 个障碍物具有的属性数。

### 3.4 适应度函数设计

在自动驾驶安全关键场景生成中，适应度函数用于评估每个测试场景，以指导进化搜索算法的进化方向。适应度函数指标需要考虑发生交通违规的可能性、多样性等因素。根据自动驾驶车辆与其他车辆、行人或障碍物之间的交互，设计基于领域知识和车辆安全的适应度函数指标。

根据以上目标，本文的适应度函数由以下指标组成：最小碰撞估计距离（Minimum Collision Estimated Distance, MCED）、EGO 加速度变化（Variation of Acceleration, VOA），以及 NPC 车辆轨迹平均欧氏距离（Average Euclidean Distance, AED）。GCE3S 在对抗性扰动的情况下最大化 VOA 与 AEDF，并在高碰撞风险的情况下最小化 MCED。

1）MCED：即最小碰撞估计距离。每个测试场景的 MCED 是 EGO 车辆与 NPC 车辆之间的最小距离的二范数，因此选择基于 EGO 与 NPC 之间的最小距离的二范数作为适应度函数指标之一，如式(1)所示。

$$MCED=\min\{\sqrt{(x_i-x)^2+(y_i-y)^2}\mid i\in m\} \tag{1}$$

其中，$(x_i,y_i)$ 是 NPC 车辆的位置，$(x,y)$ 是 EGO 车辆所在的位置，$m$ 为测试场景中障碍物的数量。GCE3S 采用车辆动力学模型应用于碰撞中违反的安全约束条件，计算每个 NPC 与 EGO 的碰撞估计距离，其中最小碰撞估计距离即为 MCED。当 EGO 车辆发生安全违规时，相应的 MCED 值为 0，MCED 越小，适应度得分越高。

2）VOA：VOA 为行驶过程中 EGO 车辆加速度变化的最大值。GCE3S 记录汽车行驶过程中的实际速度，每隔 1 s 计算一次加速度，如式(2)所示。

$$VOA=\max|(v_n-v_{n-1})-(v_{n-1}-v_{n-2})| \tag{2}$$

其中，$v_n$代表 EGO 车辆在 $n$ 时刻的速度，VOA 越大，适应度得分越高。

3）AED：为了计算两个 NPC 车辆轨迹（在不同场景中）之间的欧氏距离，首先将 EGO 车辆的初始位置转换为地图上的原点坐标；然后根据相对于 EGO 车辆初始位置的位置将 NPC 车辆轨迹转换为相应坐标，GCE3S 计算 NPC 车辆轨迹与每个记录的安全关键场景中 NPC 车辆轨迹之间的欧氏距离 $D(T_{\rm npc},T_{\rm scs})$，如式(3)所示；最后计算 NPC 轨迹与所有记录的安全关键场景轨迹的平均欧氏距离 AED，如式(4)所示。

$$D(T_{\rm npc},T_{\rm scs})=\sqrt{\sum_{i=1}^{n}((x_{{\rm npc}_i}-x_{{\rm scs}_i})^2+(y_{{\rm npc}_i}-y_{{\rm scs}_i})^2)} \tag{3}$$

$$AED=\frac{1}{m}\sum_{j=1}^{m}D(T_{\rm npc},T_{{\rm scs}_j}) \tag{4}$$

其中，$T_{\rm npc}$为此时测试场景中 NPC 的轨迹；$T_{\rm scs}$为已生成安全关键场景中 NPC 车辆的轨迹；$n$ 为车辆轨迹的坐标数，AED 为所有记录的安全关键场景中，NPC 车辆轨迹与 NPC 轨迹之间的欧几里得距离的平均值；$m$ 为记录的安全关键场景的数量。AED 值越大，则适应度分数越高。

### 3.5 操作算子

#### 3.5.1 交叉算子

本文采用多点交叉的方法，在相互配对的两个染色体中随机设置多个交叉点，然后交换两个个体在所设定的两个交叉点之间的部分染色体，形成新的子代基因，交叉过程如图 4 所示。如果多点交叉只选择一个交叉点，那么多点交叉就变成了单点交叉。对于两个父代个体 $P_1=\{C_{11},C_{12},\cdots,C_{1n}\}$，$P_2=\{C_{21},C_{22},\cdots,C_{2n}\}$，$P_1$ 和 $P_2$的染色体长度为 $n$，选择 $m$ 个交叉点，即为 $d_1,d_2,\cdots,d_m$，其中 $1\leqslant d_1<d_2<\cdots<d_m\leqslant n$。多点交叉复制父代个体 $P_1$ 和 $P_2$ 到子代个体 $P_1{}'$，$P_2{}'$，对于每个交叉点 $d_j(1\leqslant j\leqslant m)$，将 $P_1$ 和 $P_2$ 在交叉点处的基因进行交换，从而得到两个新的子代个体 $P_1{}'$，$P_2{}'$。$P_1{}'$ 和 $P_2{}'$ 的计算式如式(5)和式(6)所示。

$$P_1{}'=P_1[1:d_1-1]\oplus P_2[d_1]\oplus P_1[d_1+1:d_2-1]\oplus\cdots\oplus P_2[d_m]\oplus P_1[d_m+1:n] \tag{5}$$

$$P_2{}'=P_2[1:d_1-1]\oplus P_1[d_1]\oplus P_2[d_1+1:d_2-1]\oplus\cdots\oplus P_1[d_m]\oplus P_2[d_m+1:n] \tag{6}$$

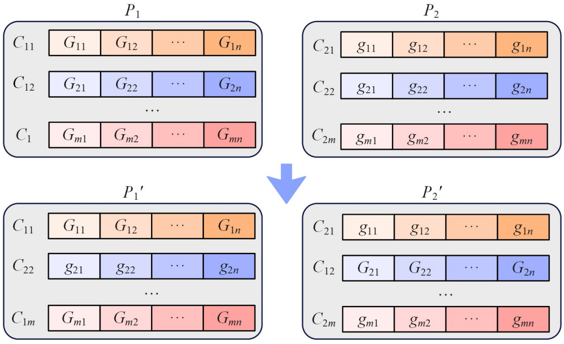



图 4 交叉操作

Fig. 4 Crossover operation

#### 3.5.2 变异算子

均匀变异是一种进化搜索算法的变异操作，通过在个体的染色体上随机选择一个基因位，并在该基因位上以均匀分布的方式（概率 $p_m$）产生一个随机的变异值，从而得到一个新的个体。对于单个个体 $S_i=\{c_1,c_2,\cdots,C_m\}$，其中的一个染色体为 $C_i=\{g_{i1},g_{i2},\cdots,g_{in}\}$，$g_{in}$代表第 $i$ 个染色体上的第 $n$ 个基因位上的值。

如图 5 所示，均匀变异首先随机选择一个基因位 $j(1\leqslant j\leqslant n)$，然后在基因位 $j$ 上产生一个随机的变异值 $\Delta g_x$，$\Delta g_x$ 服从均匀分布 $U(-\Delta_{\max},\Delta_{\max})$，其中 $\Delta$ 是一个给定的变异范围。

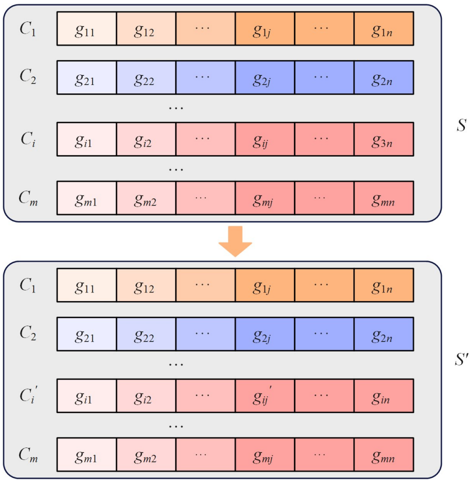



图 5 变异操作

Fig. 5 Variant operation

将变异值 $\Delta g_x$ 加到基因位 $j$ 上，得到新的基因位值 $g'_{in}$，如式(7)所示，用新的基因位值 $g_{in}{}'$替换原基因位值 $g_{in}$，得到新的染色体 $C_i{}'=\{g'_{i1},g'_{i2},\cdots,g'_{in}\}$。通过均匀变异操作，可以增加种群的多样性，避免算法陷入局部最优解，提高算法的搜索能力。

$$g'_{in}=g_{in}+\Delta g_x \tag{7}$$

高斯变异也是进化搜索算法中一种常用的变异操作，通过在个体的染色体上添加一个服从高斯分布的随机扰动来产生新的个体，这种变异操作可以在一定程度上保持种群的多样性，同时有助于算法跳出局部最优解。

高斯变异首先对每个基因 $g_{in}$ 生成一个服从高斯分布的随机数 $r_j\sim N(\mu,\sigma^2)$，其中 $N(\mu,\sigma^2)$表示均值为 $\mu$、方差为 $\sigma^2$ 的高斯分布；其次将随机数 $r_j$ 添加到基因位 $g_{in}$ 上，得到变异后的基因值 $g'_{in}$，如式(8)所示，用新的基因位值 $g'_{in}$ 替换原基因位值 $g_{in}$，得到新的染色体 $C_i{}'=\{g'_{i1},g'_{i2},\cdots,g'_{in}\}$。在上述变异中，$\sigma$ 是高斯变异的标准差，它控制变异的强度，较大的 $\sigma$ 会导致更大的变异幅度，从而增加种群的多样性；较小的 $\sigma$ 则会使变异更加局部化，有助于在当前解的附近进行精细搜索。

$$g'_{in}=g_{in}+r_j \tag{8}$$

由于均匀变异在整个解空间中搜索较为广泛，可以避免算法过早地陷入局部最优解，因此，在 GCE3S 算法搜索前期采用均匀变异算子。高斯变异通过对现有基因值添加一个遵循高斯分布的随机变量来引入较小的变化，这适用于进化搜索算法后期达到全局最优解。随着进化搜索过程的推进，需要在这些安全违规潜力大的区域进行更加细致的搜索，因此 GCE3S 后期采用高斯变异进行搜索。

#### 3.5.3 选择算子

在本文中，多目标进化搜索算法的选择操作采用锦标赛方法，如图 6 所示。锦标赛选择策略为每次从场景种群 $S$ 中取一半场景个体 $S'$，然后在这些场景中选择其中最好的一个个体 $s_t$ 进入子代种群。重复该操作，直到新的种群规模达到原来的种群规模。具体的操作步骤如下：假设随机产生 $N$ 个个体作为第一代种群，接下来开始锦标赛选择，从种群中随机选择 $N/2$ 个体 $S_n$（每个个体入选概率相同），根据个体 $S_n$ 的适应度值，选择其中适应度值最好的个体 $s_{best}$，如式(9)所示，随后将个体 $S_{best}$ 加入子代种群中，即 $S_{new}=S_{new}\cup S_{best}$。重复此步骤，得到的 $N$ 个个体构成新一代种群。

$$S_{best}=\underset{i\in\left\{1,2,\cdots\frac{N}{2}\right\}}{\operatorname{argmax}} f(S_i) \tag{9}$$

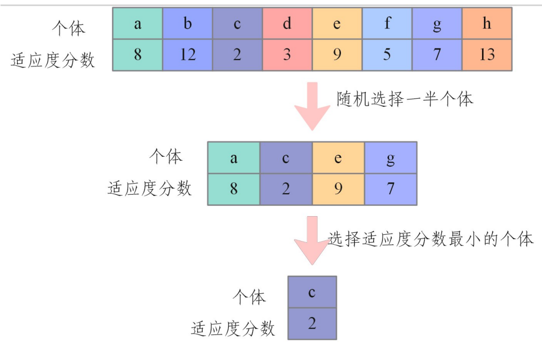



图 6 选择操作

Fig. 6 Select operation

### 3.6 重启机制

为了解决搜索过程中陷入局部极值点的问题，GCE3S 设计了重启机制，即当种群适应度分数随时间停滞不前时会触发重启机制（我们认为进化不太可能产生新的安全关键场景），重新选取一个与过去最不相似的场景作为初始种子场景，开始新一轮的进化搜索过程。也就是说，存在 $t$，使得第 $t$ 代种群 $S_t$ 的适应度分数与种群 $S_{t-1}$ 的适应度分数的差值小于等于阈值 $\varepsilon$，即 $|F(S_t)-F(S_{t-1})|\leqslant\varepsilon$，$F(S)$ 的计算式如式(10)所示，则认为目前搜索进度达到了局部极值点，此时，引入一个新的场景 $s_{new}\in S$，该场景与过去的场景最不相似，可以通过欧氏距离函数 $D:S\times S\rightarrow R$ 来衡量，其中 $D(s_1,s_2)$ 表示 $s_1$ 和 $s_2$ 之间的不相似度。

$$F(S)=\frac{1}{N}\sum_{i=1}^{N}f(s_i) \tag{10}$$

### 3.7 终止

重复上述步骤直至达到给定运行时间。GCE3S 返回所有在进化搜索过程中导致自动驾驶车辆发生安全违规的场景。在某种情况下，可能没有找到安全关键场景。此时，GCE3S 将报告为空。许多因素可能导致 GCE3S 无法找到任何安全违规情况。例如，正在测试的自动驾驶系统可能对违反定义的安全约束具有很强的鲁棒性，或者 GCE3S 搜索的迭代数太少。

## 4 仿真实验

本章使用 GCE3S 生成了用于测试工业级自动驾驶系统百度 Apollo 的安全关键场景，主要解决以下研究问题：

RQ1：GCE3S 方法生成自动驾驶安全关键场景的有效性。

RQ2：GCE3S 方法生成自动驾驶安全关键场景的效率。

### 4.1 实验设置

为了评估 GCE3S 方法的有效性和效率，本文在台式计算机 Ubuntu 20.04 系统上进行仿真实验，其配备了 64 GBRAM，1 TB 内存，CPU 处理器为 AMD Ryzen 9 3950x，GPU 为 NVIDIA GeForce RTX 3090，选择本地化的模拟器 LGSVL 2021.3[28] 和百度平台 Apollo 7.0 进行仿真实验。Apollo7.0 配备了多种传感器，如激光雷达、摄像头、GPS 等。在实验中，所有的传感器都处于打开状态，并且选择 LGSVL 地图库中较为复杂的路段地图（San Francisco）来运行 GCE3S 生成安全关键场景。

### 4.2 实验设计

GCE3S 关注的安全要求是测试中的自动驾驶车辆应该避免与场景中其他交通参与者发生碰撞，GCE3S 通过计算 MCED 来估计自动驾驶车辆的碰撞风险，其他潜在的安全因素包括遵守避让行人、红绿灯及车道等交通规则。因此，本文将 GCE3S 与两种基准方法 AV-FUZZER[14] 和 MOSAT[20] 进行比较，以证明 GCE3S 方法在生成安全关键场景方面的有效性和效率。

AV-FUZZER 是一种开源的基于搜索的测试技术，通过利用车辆动力学的领域知识并结合遗传算法来最大化自动驾驶车辆在投影轨迹上的安全违规风险，以暴露自动驾驶系统的安全违规行为。

MOSAT 是一种多目标遗传算法的测试框架，用于生成安全关键场景，测试自动驾驶系统。MOSAT 将驾驶行为分为低级原子基因和高级模式基因两部分，通过组合不同的原子基因和模式基因来创建具有挑战性的场景。在搜索过程中，MOSAT 通过多个目标函数来计算每个场景的适应度，并使用交叉变异等操作不断更新种群。最后，MOSAT 通过在仿真测试环境中执行多个场景，来更全面地评估自动驾驶系统的安全性能。

对于 RQ1，运行 GCE3S 生成测试场景并评估这些场景在自动驾驶系统中的表现，设置碰撞监听器监控自动驾驶车辆的安全违规行为，然后根据 NPC 车辆轨迹和 Apollo 的安全违规行为对生成的安全关键场景进行分类，以此验证 GCE3S 方法生成自动驾驶安全关键场景的有效性。

对于 RQ2，使用基准测试方法 AV-FUZZER 和 MOSAT 来评估 GCE3S，从生成的总测试场景数量、发现的安全关键场景数量、平均多少个测试场景发现一个安全关键场景、安全关键场景随时间推移生成的效率、平均一次实验发现多少个安全关键场景和平均生成一个安全关键场景需要多长时间这 6 个方面来评估 GCE3S 方法生成安全关键场景的效率。

为了回答上述两个研究问题，在 LGSVL 搭建的模拟环境中对百度 Apollo 系统进行了实验。在多目标进化搜索算

法中，初始种群规模大小决定了搜索空间的广度，太小的初始种群空间可能导致搜索空间不足，难以找到最优解；太大的初始种群则会大幅增加计算成本，降低搜索效率。交叉概率和变异概率决定了新个体产生的频率，交叉概率过高可能导致种群过早收敛到局部最优解，过低可能会减慢算法收敛速度；变异概率过高可能导致种群失去稳定性，难以收敛，过低可能会限制种群的多样性，导致陷入局部最优解。因此，经过多次实验发现，设置初始种群规模为 4，交叉概率为0.3，变异概率为 0.4，最大迭代次数为 100，可以在计算资源和搜索效率之间取得较好的平衡。依据领域专家的观点，在自动驾驶测试技术中，较长的搜索时间预算是不切实际的，因此 AV-FUZZER，MOSAT 和 GCE3S 每次实验运行 12 个小时。为了确保生成场景的真实性和复杂性，本文根据现实交通数据在 AV-FUZZER，MOSAT 和 GCE3S 实验中构建了两个不同的初始场景 S1 和 S2。S1 场景为直行通过十字路口，S2 场景为在十字路口右转。考虑到方法的随机性，S1 和 S2 场景在 AV-FUZZER，MOSAT 和 GCE3S 方法中各重复进行 4 次实验。

### 4.3 结果分析

#### 4.3.1 RQ1：GCE3S 方法生成自动驾驶安全关键场景的有效性

在每次实验执行过程中，GCE3S 平均生成 525.5 个测试场景（最少 429 个，最多 599 个），其中平均生成 218 个安全关键场景（最少 130 个，最多 284 个）。这意味着平均每生成 2.41 个测试场景，其中就有一个测试场景存在安全违规行为（安全关键场景）。根据障碍物的轨迹和 Apollo 系统可能存在的安全缺陷，所有安全关键场景被分为 12 种类型，这些安全违规行为可能会导致碰撞、死锁和交通拥堵。下文将举例说明每种安全关键场景，并分析 Apollo 系统可能存在的缺陷。

1）紧急情况下的软制动：在某些紧急情况下，EGO 刹车太轻，导致与 NPC 相撞。如图 7（a）所示，EGO 跟车行驶时，由于前方 NPC 的突然减速，后方 EGO 来不及反应而继续以当前速度行驶，导致 EGO 追尾 NPC。

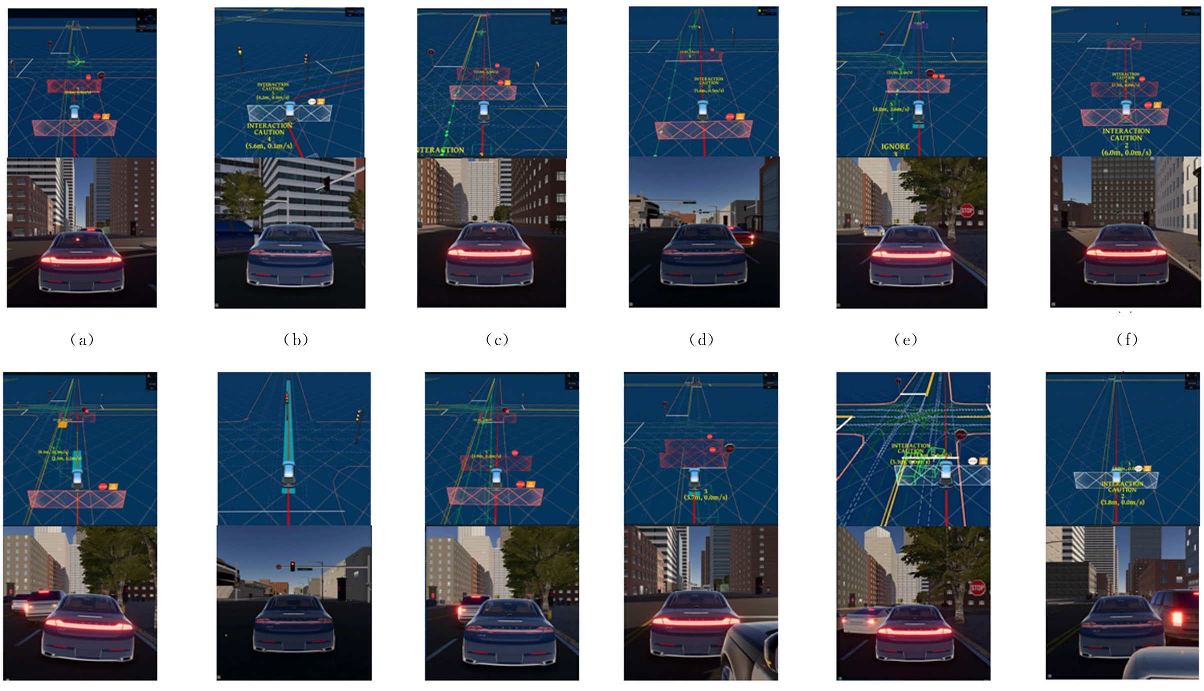



图 7 安全关键场景

Fig. 7 Safety critical scenarios

2）错误规划超车路线：EGO 可能无法超车，从而与相邻的 NPC 相撞。这种情况下，EGO 应该让行，而不是试图超车。如图 7（b）所示，EGO 跟车行驶时，由于 NPC 行驶缓慢，所以 EGO 规划超车路线，但右侧相邻车道有多辆 NPC 正常通行，此时不适合超车，EGO 应该采取让行并继续跟车行驶，以避免发生碰撞。

3）错误规划导致死锁：当前方发生交通事故时，EGO 无法重新规划路线，陷入死锁状态。如图 7（c）所示，当 EGO 正常行驶时，前方 NPC 发生交通事故，而 EGO 无法重新规划路线避让交通障碍，导致 EGO 陷入死锁状态。

4）错误预测 NPC 的运动轨迹：NPC 从相邻车道切入时，EGO 错误地预测了其运动轨迹，并没有让行，导致两车碰撞。如图 7（d）所示，当 EGO 正常巡航行驶时，右侧相邻车道的 NPC 从前方切入，EGO 错误地预测了 NPC 的运动轨迹，无法采取有效的避让行为，导致两车碰撞。

5）停止线处停止运动：当经过由停止标志控制的路口时，EGO 会在白色停止线前停止，并且不会继续行驶。如图 7（e）所示，当 EGO 行驶到交叉路口的白色停止线前时，停车并且不再启动，导致死锁，陷入交通拥堵状态。

6）跟车行驶导致无法完成驾驶任务：当前方 NPC 行驶缓慢时，EGO 即使有足够的距离改变车道，也没有规划变道路线，导致长时间的停车等待，甚至无法完成行驶任务。如

图 7(f)所示，当 EGO 跟车行驶时，前方 NPC 行驶缓慢，甚至出现静止状态，此时 EGO 与前方车辆保持足够的安全距离，但 EGO 没有规划变道路线，陷入了停滞状态且无法完成驾驶任务。

7)无法识别轨迹重叠的 NPC：当两辆 NPC 的运动轨迹在 EGO 感知的重叠区域内时，EGO 无法识别两辆 NPC，也不能正确预测两辆 NPC 的运动轨迹，从而没有让行切入的 NPC，导致两车碰撞。如图 7(g)所示，EGO 左前方有两辆 NPC，其中一辆 NPC 正在试图超车，此时两辆 NPC 的运动轨迹恰好落入 EGO 感知范围内的重叠区域，而 EGO 无法识别两辆 NPC，错误地认为其为一辆 NPC，所以 EGO 继续以当前速度行驶，没有避让切入车辆，导致两车碰撞。

8)错误识别交通信号灯：当经过由交通信号灯控制的交叉路口时，EGO 闯红灯通过交叉路口。如图 7(h)所示，EGO 到达交叉路口时，已经是红灯状态，但 EGO 并没有停车等待，而是在红灯的情况下通过交叉路口。虽然没有发生严重的交通事故，但是 EGO 这种闯红灯行为是导致交通事故的重要安全隐患。

9)跟车距离过近导致追尾：如图 7(i)所示，EGO 跟随距离较近的 NPC 车辆，NPC 车辆突然减速，EGO 车辆的感知模块没有检测到 NPC 车辆的减速，导致 EGO 与 NPC 发生碰撞。

10)后方快速来车未及时避让：如图 7(j)所示，当 NPC 车辆在车道上缓慢行驶时，EGO 试图超过 NPC 车辆。规划模块计划了超车轨迹。当 EGO 进入相邻车道时，另一 NPC 车辆在同一条车道上从后方加速，这并未被 EGO 的感知模块检测到，导致发生碰撞。

11)旁边车辆突发事故未及时避让：如图 7(k)所示，两辆 NPC 车辆在 EGO 车辆前行驶，当 NPC 车辆发生碰撞时，EGO 无法反应，并撞上了 NPC 车辆。

12)错误预测 NPC 车辆大小：如图 7(l)所示，NPC 车辆正行驶在车道线上，而 EGO 从侧面撞击了它。在这种情况下，传感器模块检测到了 NPC 车辆，但感知模块未能正确地检测到 NPC 车辆的大小，导致碰撞。

综上，GCE3S 能够有效地发现自动驾驶系统存在的 12 种不同类型的安全违规行为，并且这些安全违规行为是导致自动驾驶汽车出现交通安全问题的主要原因。相比之下，AV-FUZZER 仅能检测到 7 种类型的安全违规行为，MOSAT 能检测到 10 种类型的安全违规行为，与 AV-FUZZER 方法较为接近。在对 Apollo 系统的测试中，GCE3S 不仅能够在更短的时间内发现更多类型的安全关键场景，而且其安全违规行为暴露频率也显著高于 AV-FUZZER 和 MOSAT 方法。这一结果充分表明，GCE3S 能够更有效地发现自动驾驶系统的安全违规行为。为了进一步量化分析，计算了 NPC 车辆在不同类型安全关键场景中轨迹之间的欧氏距离，通过 5 次独立的重复实验计算其平均值，结果如表 3 所列。AV-FUZZER 的平均欧氏距离为 51.24 m，MOSAT 的平均欧氏距离为 69.57 m，而 GCE3S 的平均欧氏距离为 78.19 m。这一数据对比表明，GCE3S 生成的安全关键场景在空间分布上更多样化，从而能够发现自动驾驶系统更多种类的安全漏洞。

根据自动驾驶汽车的行驶轨迹和 Apollo 系统可能存在的设计缺陷，将生成的安全关键场景进行分类。在表 3 中，GCE3S 方法生成的安全关键场景类型为 12 种，AV-FUZZER 方法生成的安全关键场景类型为 7 种，MOSAT 方法生成的安全关键场景类型为 10 种，即 GCE3S 方法每生成 145.3 个有效的安全关键场景就能发现一个不同类型的安全关键场景。相比之下，AV-FUZZER 方法发现一种类型的安全关键场景需要 156 个有效的安全关键场景，MOSAT 方法发现一种类型的安全关键场景需要 144.9 个有效的安全关键场景。由此可知，AV-FUZZER 方法生成的不同类型的安全关键场景数量明显少于 GCE3S，表明 AV-FUZZER 方法生成了更多相似类型的安全关键场景，GCE3S 方法发现安全关键场景的类别相较于 AV-FUZZER 提高了 71.4%，相较于 MOSAT 提升了 20%。

表 3 安全关键场景生成有效性对比

Table 3 Comparison of effectiveness of generating safety-critical scenarios

| 场景 | 方法 | 安全违规类型数量 | 平均欧氏距离/m |
|---|---|---|---|
| S1 | AV-FUZZER | 5 | 56.62 |
| | MOSAT | 6 | 75.36 |
| | GCE3S | **8** | **82.45** |
| S2 | AV-FUZZER | 6 | 45.86 |
| | MOSAT | 8 | 63.78 |
| | GCE3S | **10** | **73.93** |
| S1+S2 | AV-FUZZER | 7 | 51.24 |
| | MOSAT | 10 | 69.57 |
| | GCE3S | **12** | **78.19** |

值得注意的是，AV-FUZZER 所揭示的 7 种安全违规行为和 MOSAT 发现的 10 种安全违规行为均被 GCE3S 披露，这进一步证明了 GCE3S 在揭示不同类型的安全关键场景方面的高效性，并且相较于 AV-FUZZER 和 MOSAT，GCE3S 发现的不同类型安全关键场景之间的差异也更大。

#### 4.3.2 RQ2：GCE3S 方法生成自动驾驶安全关键场景的效率

本小节在相同的场景路段下运行 AV-FUZZER 和 MOSAT，AV-FUZZER 和 MOSAT，分别生成并运行了 3000 多个测试场景，通过对比 AV-FUZZER 和 MOSAT 方法，从以下 6 个方面来讨论 GCE3S 方法生成安全关键场景的效率。

1)生成的总测试场景数。

2)发现安全关键场景的数量。

3)安全关键场景随时间推移生成的效率。

4)平均发现一个安全关键场景所需测试场景的数量。

5)平均一次实验发现安全关键场景的数量。

6)平均生成一个安全关键场景所需时间。

在仿真模拟实验中，每次实验生成安全关键场景的数量如图 8 所示。在 S1 场景的 4 次实验中，GCE3S 生成的安全关键场景数量分别为 260，264，283，204；AV-FUZZER 生成的安全关键场景数量分别为 118，130，166，145；MOSAT 生成的安全关键场景数量分别为 178，198，217，189。在 S2 场景

的 4 次实验中，GCE3S 生成的安全关键场景数量分别为 172，130，219，211；AV-FUZZER 生成的安全关键场景数量分别为 118，118，152，145；MOSAT 生成的安全关键场景数量分别为 137，149，1 996，185。由此可见，在相同的时间内，GCE3S 生成的安全关键场景数量比 MOSAT 和 AV-FUZZER 都更多，其较 AV-FUZZER 多 59.7%，相较于 MOSAT 多 20.4%。

图 9 给出了 GCE3S 方法、MOSAT 方法和 AV-FUZZER 方法基于 S1 和 S2 初始场景下发现的安全关键场景数量随时间推移的增长情况。从基于 S1 和 S2 初始场景下的 8 次对比实验可以看出，GCE3S 方法发现的安全关键场景均多于 AV-FUZZER，只有在 S2-2 实验中，MOSAT 方法发现的安全关键场景数量多于 GCE3S 方法，其余实验发现的安全关键场景数量均少于 GCE3S 方法。GCE3S 在前期搜索安全关键场景的增长速率稍领先于 AV-FUZZER 和 MOSAT，但随着时间的推移，GCE3S 方法生成安全关键场景的效率略高于 MOSAT 方法，明显领先于 AV-FUZZER 方法。

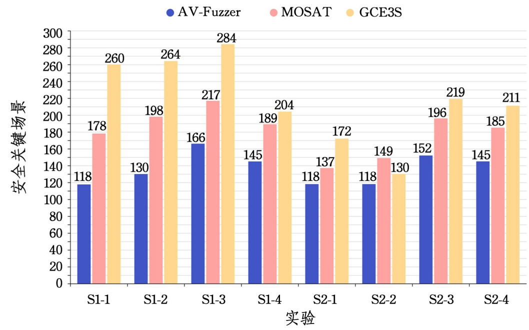
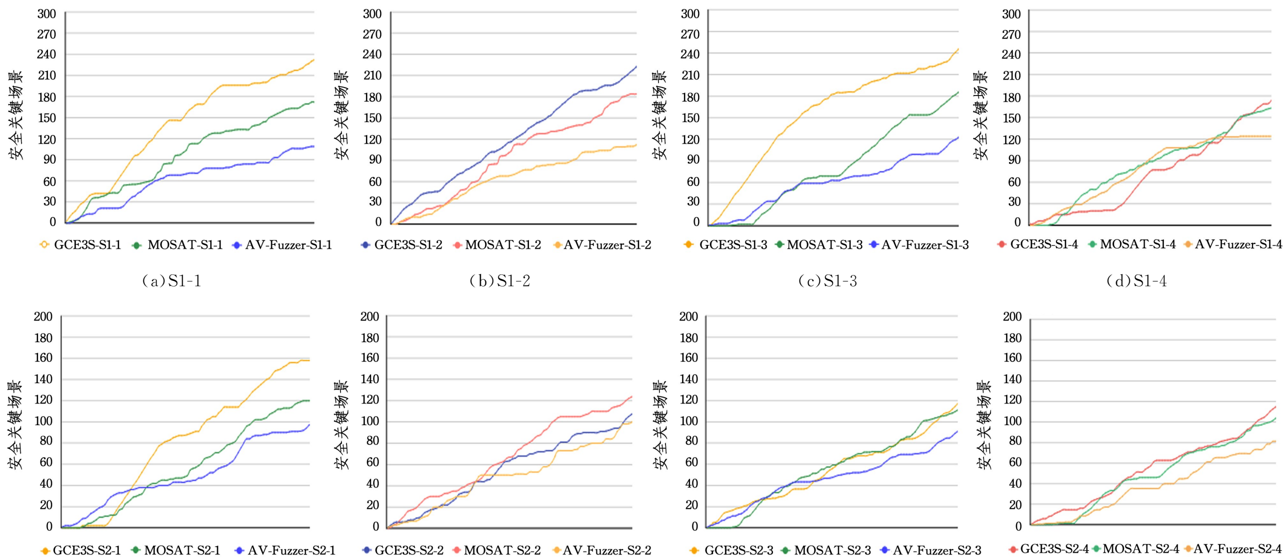



图 8 安全关键场景生成数量对比

Fig. 8 Comparison of the number of safety critical scenarios generated




(a) S1-1 (b) S1-2 (c) S1-3 (d) S1-4

(e) S2-1 (f) S2-2 (g) S2-3 (h) S2-4

图 9 安全关键场景生成进度对比

Fig. 9 Comparison of progress in generating safety critical scenarios

如表 4 所列，在相同的总运行实验中，GCE3S 在 S1 和 S2 场景的 8 次实验中，一共生成并执行了 4 204 个测试场景，而 AV-FUZZER 一共生成并执行了 3 840 个测试场景，MOSAT 一共生成并执行了 4 160 个测试场景。这表明相较于 AV-FUZZER 和 MOSAT 方法，GCE3S 能够更高效地生成测试场景，并运行测试场景，得到测试结果。

表 4 安全关键场景生成效率对比

Table 4 Comparison of the efficiency of generating safety-critical scenarios

| 场景 | 方法 | 总测试场景数 | 总安全关键场景数 | 平均所需测试场景成本 | 平均生成数量 | 平均生成时间成本/s |
|---|---|---|---|---|---|---|
| S1 | AV-FUZZER | 1884 | 559 | 3.37 | 139.750 | 309.1 |
| | MOSAT | 2126 | 782 | 2.72 | 195.500 | 221.0 |
| | GCE3S | **2248** | **1012** | **2.22** | **252.750** | 170.9 |
| S2 | AV-FUZZER | 1956 | 533 | 3.67 | 133.250 | 324.2 |
| | MOSAT | 2 034 | 667 | 3.05 | 166.750 | 259.1 |
| | GCE3S | **2179** | **732** | **2.98** | **183.000** | 236.1 |
| S1+S2 | AV-FUZZER | 3840 | 1092 | 3.52 | 136.500 | 316.5 |
| | MOSAT | 4160 | 1449 | 2.87 | 181.125 | 238.5 |
| | GCE3S | **4204** | **1744** | **2.41** | **217.875** | **198.3** |

具体地，GCE3S 在 S1 场景中每生成 2.22 个测试场景发现一个安全关键场景，平均生成的安全关键场景数量为 252.75 个，平均生成一个安全关键场景需要 170.9 s；AV-FUZZER 在 S1 场景中每生成 3.37 个测试场景发现一个安全关键场景，平均生成的安全关键场景数量为 139.75 个，平均生成一个安全关键场景需要 309.1 s；MOSAT 在 S1 场景中每

生成 2.72 个测试场景发现一个安全关键场景，平均生成的安全关键场景数量为 195.5 个，平均生成一个安全关键场景需要 221.0 s。GCE3S 在 S2 场景中每生成 2.98 个测试场景发现一个安全关键场景，平均生成的安全关键场景为 183 个，平均生成一个安全关键场景需要 236.1 s；AV-FUZZER 在 S2 场景中每生成 3.67 个测试场景发现一个安全关键场景，平均生成的安全违规数量为 133.25 个，平均生成一个安全关键场景需要 324.2 s；MOSAT 在 S2 场景中每生成 3.05 个测试场景发现一个安全关键场景，平均生成的安全关键场景数量为 166.75 个，平均生成一个安全关键场景需要 259.1 s。因此，GCE3S 方法能够提高有效安全关键场景生成的效率，相较于 AV-FUZZER 方法的安全关键场景的平均时间成本减少了 37.3%，相较于 MOSAT 方法的安全关键场景生成的平均时间成本减少了 16.9%。

**结束语** 本文提出了一种基于多目标进化搜索驱动的测试方法 GCE3S，通过场景的基因表达方式构建多样性的测试场景，以揭露自动驾驶系统的安全违规行为。具体来说，GCE3S 基于测试场景的基因建模构建更加细致化的场景测试自动驾驶系统，并通过多个目标函数引导进化搜索的方向，从而生成多样性的安全关键场景。基于 LGSVL 和 Apollo 的仿真实验结果表明，在相同的实验时间内，GCE3S 生成的安全关键场景数量相较于最好的基准 MOSAT 方法提升了 20.4%，生成的安全关键场景多样性相较于最好的基准 MOSAT 提升了 20%。

未来的工作目标是扩展 GCE3S 方法：1）将 GCE3S 方法扩展到其他自动驾驶系统和模拟器中，如 Autoware，carla；2）通过基因建模的方式构建更加复杂的测试场景，如添加天气、行人等障碍物。

## 参 考 文 献


[1] Ministry of Industry and Information Technology，National Standards Commission. Notice of the two departments on the issuance of the Guidelines for the Construction of National Telematics Industry Standard System（Intelligent Networked Vehicles）（Version 2023）[EB/OL]. [2024-07-09]. https://www.gov.cn/zhengce/zhengceku/202307/content_6894735.htm.

[2] TESLA-Tesla China. Autopilot [EB/OL]. [2024-07-09]. https://www.tesla.cn/autopilot?isappinstalled=0.

[3] WAYMO. Waymo Driver [EB/OL]. [2024-07-09]. https://waymo.com/.

[4] HUAWEI. Qian Kun ADS 3.0 [EB/OL]. [2024-07-09]. https://auto.huawei.com/cn/.

[5] BAIDU. Apollo[EB/OL]. [2024-05-19]. https://apollo.baidu.com.

[6] JIANG Z M，DANG S B，LI H Y，et al. A review of research progress on scenario testing of self-driving cars[J]. Automotive Technology，2022(8)：10-22.

[7] ZHAO X M National Key R&D Programme(2021YFB2501200) Team，ZHAO X M. Research progress of autonomous driving test and evaluation technology[J]. Journal of Transportation Engineering，2023，23：10-77.

[8] ZAMPETTI F，KAPUR R，PENTA M D，et al. An empirical characterization of software bugs in open-source Cyber-Physical Systems[J]. Journal of Systems and Software，2022，192：111425.

[9] CHEN J Q，SHU X X，LAN F C，et al. Construction of autonomous driving test scenarios with typical hazardous accident characteristics[J]. Journal of South China University of Technology (Natural Science Edition)，2021，49(5)：1-8.

[10] GAMBI A，HUYNH T，FRASER G. Generating effective test cases for self-driving cars from police reports[C]//Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2019：257-267.

[11] ZHANG X，CAI Y. Building Critical Testing Scenarios for Autonomous Driving from Real Accidents[C]//Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. 2023：462-474.

[12] LEILABADI S H，SCHMIDT S. In-depth analysis of autonomous vehicle collisions in california[C]//International Conference on Intelligent Transportation Systems. 2019：889-893.

[13] ARCAINI P，ZHANG X Y，ISHIKAWA F. Less is More：Simplification of Test Scenarios for Autonomous Driving System Testing[C]//15th IEEE Conference on Software Testing，Verification and Validation，ICST 2022. IEEE，2022：279-290.

[14] LI G，LI Y，JHA S，et al. AV-FUZZER：Finding Safety Violations in Autonomous Driving Systems[C]//31st IEEE International Symposium on Software Reliability Engineering，ISSRE 2020. 2020：25-36.

[15] FENG S，FENG Y H，YU C，et al. Testing Scenario Library Generation for Connected and Automated Vehicles，Part I：Methodology[J]. IEEE Transactionson Intelligent Transportation Systems，2021，22(3)：1573-1582.

[16] FENG S，FENG Y，SUN H W，et al. Testing Scenario Library Generation for Connected and Automated Vehicles，Part II：Case Studies[J]. IEEE Transaction on Intelligent Transportation Systems，2020，22(9)：5635-5647.

[17] KIM S，LIU M，RHEE J J，et al. DriveFuzz：Discovering Autonomous Driving Bugs through Driving Quality-Guided Fuzzing[C]//Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security，CCS 2022. 2022：1753-1767.

[18] LUO Y，ZHANG X Y，ARCAINI P，et al. Targeting Requirements Violations of Autonomous Driving Systems by Dynamic Evolutionary Search[C]//36th IEEE/ACM International Conference on Automated Software Engineering，ASE 2021. 2021：279-291.

[19] ZHANG X，ZHAO W，SUN Y，et al. Testing Automated Driving Systems by Breaking Many Laws Efficiently[C]//Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis，ISSTA 2023. 2023：942-953.

[20] TIAN H X，JIANG Y，WU G Q，et al. MOSAT：finding safety violations of autonomous driving systems using multi-objectiveg-

enetic algorithm[C]// Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022. 2022: 94-106.

[21] HUAI Y Q, ALMANEE S, CHEN Y T Y, et al. scenoRITA: Generating Diverse, Fully Mutable, Test Scenarios for Autonomous Vehicle Planning[J]. IEEE Transactions Software Engineering, 2023, 49(10): 4656-4676.

[22] Ministry of Industry and Information Technology. Public Notice for Submission and Approval of Recommended National Standardsfor Automated Vehicle Driving Classification [EB/OL]. [2024-07-09]. https://www.miit.gov.cn/zwgk/wjgs/art/2020/art_9a7eb2afbd5c411e88b5bbfc7012d7b1.html.

[23] Ministry of Industry and Information Technology. Automotive driving automation classification [EB/OL]. [2024-07-09]. https://std.samr.gov.cn/gb/search/gbDetailed?id=CA6C0E542CB4C983E05397BE0A0AED11.

[24] Ministry of Industry and Information Technology, Ministry of Public Security, Ministry of Housing and Urban-Rural Development, et al. Circular of four ministries on the pilot work of access and on-road passage of intelligent networked vehicles [EB/OL]. [2024-07-09]. https://www.gov.cn/zhengce/zhengceku/202311/content_6915788.htm.

[25] CCTV. China carries out a pilot project on access and on-road passage of intelligent networked vehicles [EB/OL]. [2024-07-09]. https://www.gov.cn/yaowen/shipin/202406/content_6955492.htm.

[26] DAI J R, LI Z R, ZHANG W Q, et al. Simulation fuzzy testing for unmanned systems: current status, challenges and perspectives [J]. Computer Research and Development, 2023, 60(7): 1433-1447.

[27] ZHU X L, WANG H C, YOU H M, et al. Review of autonomous driving intelligent system testing research[J]. Journal of Software, 2021, 32(7): 2056-2077.

[28] LG. SVL[EB/OL]. [2024-07-13]. https://svlsimulator.com/.

**SUN Lele**, born in 1999, postgraduate, is a member of CCF (No. V0414G). His main research interests include software testing and automous driving testing.

**HUANG Song**, born in 1970, Ph.D, professor, Ph. D supervisor, is a distinguished member of CCF (No. 29597S). His main research interests include software engineering, software security, software testing and quality assessment.

（责任编辑：何杨）