关于半监督测试用例生成对仿真模型的益处

Xiao Ling 和 Tim Menzies，*IEEE* 会士

摘要——复杂仿真模型的测试可能成本高昂且耗时。当前探索这一问题的最先进方法均采用全监督学习，即要求所有示例都带有标签。另一方面，GenClu 系统（本文提出）采用半监督方法，即：（a）实际上只需对一小部分信息进行标注（通过仿真）；（b）随后将这些标签传播到其余数据中。当应用于五个信息物理系统的开源仿真模型时，GenClu 的测试生成速度比以往最先进的方法快数个数量级。此外，通过变异测试进行评估发现，GenClu 生成的测试与本文测试的其他方法相比，表现同样出色甚至更优。因此，我们建议采用半监督方法，而非以往的方法（进化搜索和全监督学习）。

为推动开放科学，本研究中使用的所有数据和脚本均可在 https://github.com/ai-se/CPS test generation 获取。_ _

索引词 **s**—基于搜索的软件工程、建模与模型驱动工程、验证与确认 on

arXiv:2305.03714v2 [cs.SE] 1 Dec 2023

1 引言

基于仿真的测试可能非常缓慢且成本高昂。对于复杂的信息物理系统（CPSs）而言尤其如此，因为这类系统将软件与物理组件结合在一起（每次此类仿真所需的时间可能是传统软件仿真的数倍 [1]）。Arrieta 等人警告说，测试复杂系统需要数小时到数天的中央处理器时间 [2],[3], [4]。

传统的仿真模型测试用例生成方法假设所有示例都可以标注。由于生成这些标签可能需要进行仿真，标注过程可能非常缓慢。例如，EPIcuRus 是一种“完全监督”方法，因此其算法需要为每个示例提供标签。其他方法则采用更慢的元启发式算法（例如遗传算法） [5], [2], [1], [6]。

为解决标注问题，我们提出了一种基于无监督聚类的半监督“少标签”方法，称为 GenClu（仅需极少量标签）。如图 1 所示，GenClu 会随机生成测试用例：从已知值的最小值到最大值之间随机选取值。请注意，该过程生成的是有效的测试用例输入，但其测试用例输出是未知的。

接下来，基于流形假设（见第 2.1 节），GenClu 根据候选解的独立属性，将大量候选解分成若干小簇。对于这种聚类，我们使用递归 FASTMAP 聚类算法 [7], [8]。然后，我们只需模拟每个簇中的一个代表候选解，即可找到最佳候选解组。

GenClu 的一个有趣特点是，无需进一步构建另一个模型（例如通过决策树分类），因为对于 GenClu 而言，簇就是模型。

• *X. Ling* 和 *T. Menzies* 就职于美国北卡罗来纳州立大学计算机科学系，地址：罗利，美国。电子邮箱：*lingxiao-hzsz3ban@gmail.com*，*timm@ieee.org*

这意味着从理论上讲，GenClu 的速度远快于元启发式搜索或全监督方法，因为（a）它所需的模拟器运行次数少得多；（b）它无需经过多个建模步骤。本文中的实验证实了这些理论预测：

- 通过变异测试评估，GenClu 的表现不逊于本文测试的任何其他方法，甚至更优（RQ1）。
- GenClu 的运行速度比 OD 和 SAMOTA 系统快几个数量级。实际应用中，这意味着 GenClu 只需几分钟即可终止，而其他方法则需要数小时（RQ2）。
- 我们发现，借助 GenClu 生成的树结构，可以对排名靠前的几个簇中的测试用例进行变异，从而生成性能优于其他簇中新测试用例的测试用例（RQ3）。

基于上述结果，对于仿真模型的测试用例生成，我们建议采用半监督方法，而非进化搜索（如 OD 所采用的方法）或基于代理模型的优化方法（如 SAMOTA 所采用的方法）。

本文其余部分安排如下：第2节介绍本研究所需的重要背景知识以及相关工作。第3节阐述我们提出的 GenClu 框架。第4节简要介绍两种先进算法 OD 和 SAMOTA。第5节介绍案例研究、变异体生成、性能评价标准和统计分析。第6节呈现实验结果，并根据结果总结对各项研究问题的结论。第7节说明本研究可能存在的威胁，第8节总结本研究并提出未来工作方向。

2 背景

2.1 半监督学习

半监督学习器（SSL）旨在“从少量带标签的数据点中找到最佳模型” [9]。例如，主动学习器 是一种 SSL 学习器，它.....

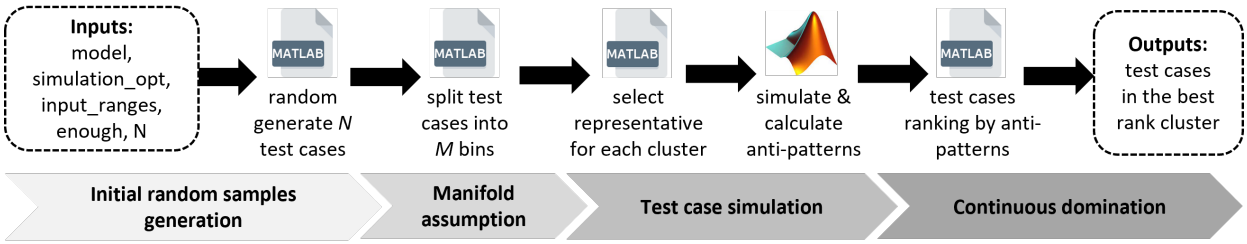


图 1：我们提出的聚类测试用例生成方法 GenClu 的框架。从图中可以看出，该聚类方法简单直接，无需迭代或演化。

(a) 仅标注一组具有代表性的数据点，(b) 基于这些数据构建模型，(c) 利用该模型寻找下一组需要标注的代表性数据点 [10]。半监督学习（SSL）已广泛应用于传统软件工程（SE）任务（例如测试用例优先级排序 [11],查找相关文献 [12],和软件分析 [9]），而这些任务需要标注所有数据点，因此需要较高的预算。

本文探讨基于聚类的半监督学习（SSL）技术能否从随机生成的测试用例集中“筛选”出最相关的测试用例[1]。这是一个合理的问题，因为在仿真模型上进行测试可能非常耗时。例如，高保真系统（如无人机、电梯和自动驾驶汽车）仿真一项任务可能需要数小时至数天 [1], [13], [2]。因此，为所有测试用例（例如 100 至 150 个测试用例）添加标签可能会耗费极长时间。这促使我们寻找一种所需标签更少、因而仿真次数也更少的测试用例生成方法。

SSL 是一种基于流形假设来处理数据挖掘者所探索数据的技术：

- 现实世界中的许多高维数据集实际上位于该高维空间内的低维潜在流形上 [14]。
- 因此，看似使用了许多属性的领域，实际上可以用相对较少的属性来描述，而不会丢失信息。

流形假设对我们探索世界的方式有着重要影响。任何数据集中的选项空间，都是各属性 A的基数 C 的笛卡尔积；即 C^A。根据流形假设，有效属性数为 $A' \subset A$。因此，选项空间实际上是 $C^{A'}$，而当 $|A'| \ll |A|$时，与查看所有属性的所有取值相比，这一空间要小得多，因而更易于探索。

一种理解 SSL 的方式，是将其视为一种智能采样策略，以用最少的观测获取最多的信息。采样是一种统计方法，它从大量初始样本中选取具有信息量的样本来开展实验。Kiaer 等人 [15] 是最早在 1895 年将采样方法用于社会和经济数据的杰出研究者。几种著名的概率抽样技术广泛运用了聚类，包括随机抽样、系统抽样、分层随机抽样、聚类抽样和多阶段抽样 [16]。在此，我们将所有这些采样方法扩展为采用 Nystr¨om 算法（仅基于大型矩阵的一个矩形子矩阵，对该大型矩阵的特征向量进行近似 [8]）。通过对特征向量进行聚类，GenClu 可以找到优质的簇，这些簇能够最大程度地根据数据的“形状”（由特征向量定义）划分数据。有关我们所用 Nystr¨om 近似方法的更多详情，请参见第 3 节的聚类方法。

据我们所知，本研究是首个将基于 Nystr¨om 簇的采样应用于信息物理系统测试任务的研究。

2.2 信息物理系统（CPSs）

信息物理系统嵌入其所处的环境中，至少在我们的案例研究中，这些系统广泛运用了控制理论 [17]。在控制理论中，反馈控制器用于将过程变量的值或状态与期望的设定值进行比较。随后，该控制器将两者之间的差值作为控制信号施加，以使被控对象的过程变量输出达到与设定值相同的值 [18]。因此，许多信息物理系统开发者广泛采用基于仿真的测试方法。由于高保真模拟器通常作为信息物理系统构建的一部分生成，这种方法得到了广泛应用 [19]。

信息物理系统的开发者通常依赖一些仿真开发工具（例如 Simulink）来构建系统 [20]。图 2 展示了一个只有两个层级的 Simulink 模型示例（即汽车巡航控制器） [2],[18]。如图所示，整个系统包含在一个容器中，该容器有 6 个输入端口和 2 个输出端口。在这个容器中，整个系统由许多逻辑运算符、算术运算符、开关模块和关系运算符组合而成，它们通过连线连接起来。此外，子模块的作用类似于常规编程中的“函数”，其中包含用于特定目的的更多操作。而且，子模块拥有自己的输入端口和输出端口，其作用类似于“函数”的输入和输出。

与常规软件非常不同，仿真模型的输入 $x = \{x_1, x_2, \cdots, x_n\}$ 和输出 $y = \{y_1, y_2, \cdots, y_m\}$ 是信号（此处的信号指时间序列向量）。更具体地说，仿真模型会模拟时间域 $T = [0, b]$内一段时间的结果，其中每个时间步长都有对应的输入值和输出值。两个时间戳之间的间隔称为时间步长（ΔT）

1.准确地说，我们的系统从随机生成的一些测试开始，并从中选出少量信息量最大的测试。一种理解方式是“信息凝聚”；也就是说，最初我们对每个测试的取值一无所知，但经过聚类后，我们将相似的测试归为一组，从而在一处汇集足够的测试“凝聚物”，以便有依据地选出哪些测试有用。

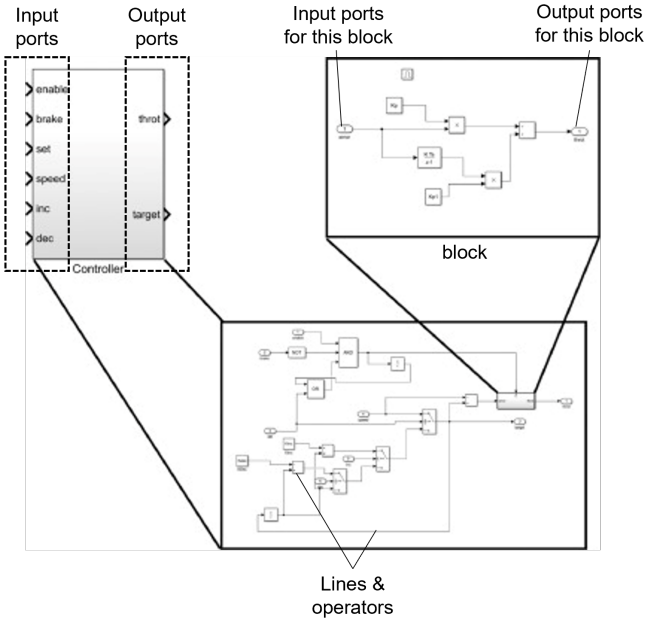
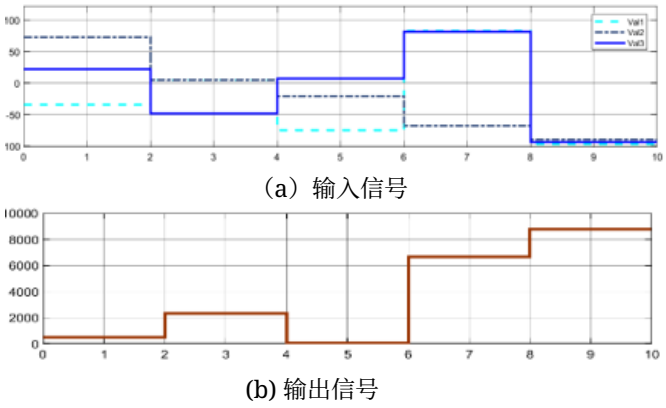


图 2：汽车巡航控制器的 Simulink 模型示例 [2], [18]。



（a）输入信号

(b) 输出信号

图 3：输入信号和输出信号的示例

，通常由开发人员定义。例如，某个仿真模型的仿真时间设置为 10 秒（$T = [0, 10]$），时间戳为 0.001，那么将有 $10/0.001 + 1 = 10001$ 个仿真步。因此，这表示该模型的输入和输出应为长度为 10001 的向量。

2.3 仿真模型测试

过去几十年来，人们广泛研究了仿真模型的测试用例生成。测试用例生成的共同目标是生成高效的 测试套件（即减少测试运行次数），同时提高有效性 [21], [22],[23], [24], [25]。然而，在这两个目标之间找到平衡点始终是一项挑战。

Ling 等人 [18] 讨论了仿真模型测试的特点，以及这些特点如何使其区别于其他类型的软件测试。仿真模型（尤其是信息物理系统）嵌入于其环境之中，因此，仅像传统软件那样探索静态特征（例如代码库）是不够的。测试这些模型需要检验代码库对周围环境的响应。因此，不建议仅通过探索静态特征来测试这些模型。

此外，仿真模型广泛应用过程控制理论，其中反馈控制器用于将过程变量的值或状态与期望的设定值进行比较。此外，运行这些模型有时成本很高，因为许多模型单次运行就需要一到几天 [5], [2], [6]。因此，对于仿真模型而言，收集大量测试用例执行数据或覆盖率信息是一项困难且昂贵的工作。因此，从仿真模型获得的数据（例如输入向量和输出向量）是测试系统的唯一来源。在本研究中，我们仅使用仿真模型中的输入和输出信号，而不使用测试执行历史或覆盖率信息，来生成一个充分的测试套件。

本研究的目标是为仿真模型生成一个充分的测试套件，以尽可能多地检测故障。正如我们稍后将要讨论的，本研究中的“故障”是人为添加的变异体，我们用它们来评估测试用例的性能。在衡量这一性能目标时，我们更倾向于包含较少测试用例的测试套件。由此，我们可以将测试用例生成问题定义如下：

> 给定一个仿真模型 M，并定义一个评估函数 f 来评估测试用例生成方法的性能，我们希望找到一种测试套件生成方法 P_{opt}，使得由 P_{opt} 生成的测试套件 TS_{opt} 具有最高性能。更具体地说，$f(P_{opt}) > f(P)$ 并且$1 \leq |TS_{opt}| \leq |TS|$。

Gaaloul 等人指出了仿真模型中的测试生成挑战，如下所示 [26]:

- 为仿真模型生成的测试用例在大多数情况下应采用信号格式（随时间变化的函数），除非输入是常量。
- 生成的信号必须对其对应的模型有意义，并且应以符合现实的方式生成。这里的“有意义”是指，数值不能过大或过小，以至于模拟模型无法在现实世界中达到。

为应对这两个挑战，Gaaloul 等人 [26]采用了 Matlab 的方法，通过使用（a）插值函数（i_u）、（b）输入域（r_u）和（c）控制点数量（c_u）来生成信号。这里，控制点表示输入信号中信号模式发生变化的特殊点。本研究采用了这种方法，并遵循其工作中的生成模式。具体而言，我们创建了一段 Matlab 代码，将上述三个要求作为输入，并返回 n 个信号，其中 n为输入的数量。对于输入信号 I_m中的每个控制点 c_{u_k}，代码都会在范围 r_{u_m} 内随机生成一个值，并使控制点在时间域 T内均匀分布。之后，利用插值函数生成连接各控制点的信号。Gaaloul 等人指出，Matlab 中包含的插值函数有其他（用户自定义）、线性、分段常数插值和分段三次插值 [26]。在本研究中，除一个模型外，我们所检查的文献中的所有模型均使用分段常数插值，而该模型则

.....

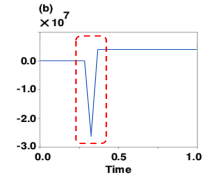
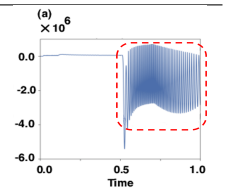
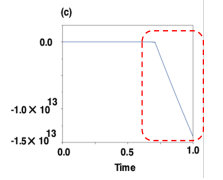
Name	描述	Goal	公式	示例
不连续性	持续时间短 脉冲，位于 the 输出 信号 [27]	Max	$dis(O_i) = \max_{dt=1}^{3}(\max_{j=dt}^{k-dt}(min(lc_j, rc_j)))$ lc_j (左变化率) $= \lvert sig(j \cdot \Delta t) - sig((j-dt) \cdot \Delta t)\rvert / \Delta t$ rc_j (右变化率) $= \lvert sig((j+dt) \cdot \Delta t) - sig(j \cdot \Delta t)\rvert / \Delta t$	
不稳定性	持续时间#+#+#+#+ 快速且 频繁的 振荡中 输出 信号 [27]	Max	$ins(O_i) = \sum_{j=1}^{k} \lvert sig(j \cdot \Delta t) - sig((j-1) \cdot \Delta t)\rvert$	
增长至 无穷大	输出信号 增加或 减少至一个 无穷大 值 [28]	Max	$inf(O_i) = \max_{j=1}^{k} \lvert sig(j \cdot \Delta t)\rvert$	
最小最大值	模型被 检测到的程度 由测试用例 [2]	Max	$minmax(O_i) = \lvert \max_{j=1}^{k}(sig(j \cdot \Delta t)) - \min_{j=1}^{k}(sig(j \cdot \Delta t))\rvert$	

表 1：文献 [2], [28], [27] 中收集的仿真输出信号反模式。在公式中，O_i 表示 i^{th} 输出，k 表示仿真步数总量，Δt 表示仿真中的时间戳。

所有输入均为常量。图3展示了由我们的代码生成的输入信号及其对应的输出信号示例。从图3(a)中可以看出，共有五个控制点，分别表示信号在时间0、2、4、6和8时的取值。所有数值均在输入范围 $\{[-100, 100];\ [-100, 100];\ [-100, 100]\}$ 内随机生成。

2.4 仿真模型中的预言机问题

描述这项工作的一种方式是，我们研究当人们可能不了解系统的所有可能标称行为和非标称行为时，如何在仿真模型中进行测试。这被称为预言机问题，在仿真模型中尤为突出 [1], [2], [6]。我们采用两种技术来解决预言机问题：

- 专门针对仿真模型开发的反模式（见 §2.4.1）；
- 变异测试（参见 §2.4.2）。

2.4.1 仿真模型反模式

仿真输出信号可能呈现一些不良模式，这些模式可以表明仿真过程中的故障行为。这些模式称为反模式，在过去的一些文献中得到了广泛研究 [18], [2], [29],[27], [28]，尤其是在没有人工预言机的情况下。我们也在实验中采用这些反模式来解决预言机问题。

通过查阅以往文献，我们收集了四种与本研究相关且可通过我们的设置获得的反模式。它们是不连续性、不稳定性、增长至无穷大和最小值与最大值之差。注意最小值和最大值之差更多体现的是有效性指标，而非反模式。不过，为简化对这些指标的描述，我们将其纳入反模式示例中。如上所述，过去的文献已证明这些术语能够成功揭示故障行为。我们在表 1 中总结了这些术语：

- **描述**列概述了每种反模式；
- **目标** 列展示了这些反模式最有可能如何从输出信号中揭示故障，
- **公式** 列给出了计算这些反模式的详细数学公式。
- **示例** 列展示了信号格式中的反模式示例。

请注意，我们所有的输出信号都会有 4 个反模式值。因此，在我们提出的聚类方法中，目标总数为 $4 * n$，其中 n 表示某个仿真模型中的输出数量。我们使用支配 谓词（如下所述）从这 $4 * n$ 个目标中选出更好的候选解。

在多目标优化中，支配谓词 用于评估优化器在多个目标之间进行权衡的效果。传统上，这通过二元支配来实现：如果一个解在所有目标上都不劣于另一个解，并且至少在一个目标上更优，那么前者就优于后者[2]

许多研究指出，对于三个或更多目标，二元支配难以区分候选解 [30], [31], [32]。因此，Zitzler [30] 提出了一种连续支配谓词

2. 换句话说，我们有零个理由偏好另一种解法，至少有一个理由偏好这种解法。

这反映了两个解法 a, b 之间目标值差值 ($\Delta(a, b)$) 的大小。给定 $1 \leq i \leq n$ 个目标以及某个权重值 w_i，对于希望最小化或最大化的目标，该权重为 $(1, -1)$，那么 Zitzler 认为，从解法 a 转到解法 b 将“付出” $\Delta(a, b) = \sum_i e^{w_i * (a_i - b_i)/n}$。随后他得出结论：如果解法 a 的损失最小，那么它就优于解法 b；即 $\Delta(a, b) < \Delta(b, a)$。

2.4.2 仿真模型的变异测试

解决预言机问题的另一种方法，是使用变异测试 来评估不同测试用例生成方法的性能。变异测试是一种基于故障的软件测试技术 [33], [34], [35], [36], [37]，用于检查测试用例揭示某些人为缺陷的能力 [34]。从原始软件中生成的缺陷或故障称为变异体。在传统软件中，变异体是通过有系统地向被测程序注入少量人为故障来创建的 [38]。然而，Simulink模型的变异体生成方式与传统软件大不相同。Simulink模型具有特定的故障模式（我们将在第§5.2节介绍），注入人为故障的操作可能包括：(a) 更改算术运算符，(b) 更改常量值，以及 (c) 更改关系运算符等 [6]。由于存在特定的故障模式，Simulink模型的变异体数量可能不如传统软件那么多（据我们观察，在对开源仿真模型进行变异体预处理后，通常约有10至100个变异体）。即使Simulink模型中的变异体数量较少，要检测出所有变异体仍然是一项挑战（例如，在本研究中，使用小型测试套件进行的重复实验中，只有10%至15%取得了满分变异分数）。在第§5.2节中，我们将详细介绍：(a) Simulink模型中的故障模式，(b) 如何为我们的案例研究手动生成变异体，以及 (c) 如何预处理这些手动生成的变异体。

2.5 其他相关工作

为求完整，本节讨论该领域的其他工作（尽管以下许多内容与我们的问题并无直接关系，即利用半监督学习引导变异测试，以检测仿真模型中的反模式）。

- 许多既有工作探讨了针对不同案例研究或各种需求的测试用例生成。2003年，Zhao等人使用遗传算法和CPS仿真模型，创建了一种测试用例生成方法。他们采用了一种基于其所定义的覆盖率问题的覆盖率度量 [39]。

- 2008年，Gadkari等人使用时序逻辑，并将其转化为形式语言，用于高层需求和测试规范，以生成测试用例 [40]。

- 2013年，Matinnejad等人提出了一种面向连续控制器的基于搜索的测试用例生成器 [41]。

- 2015年，Sinha等人提出了一种框架，利用高层功能需求自动生成测试用例。在他们的研究中，需求以自然语言描述，并通过形式化本体进行识别。在开发过程中，本体会结合控制器和被控对象模型不断完善，从而能够在任意阶段自动生成测试用例 [42]。

- 2016年，Matinneja d et al。存在 d anoth搜索 h - 基于测试用例生成，适用于时间连续仿真案例研究。他们首先依靠工程师定义一系列信号片段，然后选择一些输入来生成测试用例 [19]。

- 2017 年，Arrieta 等人将非支配排序遗传算法 II 应用于三项成本效益指标（即需求覆盖率、测试用例相似度和测试执行时间）。他们提出了一种交叉算子和三种变异算子（分别作用于测试套件级、测试用例级以及两个级别） [5]。

- 同样在 2017 年，Arrieta 等人指出，工业信息物理系统的仿真模型非常复杂，执行这些模型的计算成本也很高。为应对这些挑战，他们提出了一个包含四个目标的适应度函数，并在其案例研究中评估了五种多目标搜索算法。他们发现，NSGA-II 在这项任务中的表现优于其他多目标搜索算法 [13]。

- 2018年，Gonzalez 等人提出在系统及其环境的早期阶段，基于可执行模型开展测试。因此，他们提出了一种基于 SysML 的建模方法和一种高效的 Simulink 协同仿真框架 [3]。

- 同样在2018年，Turlea等人提出了一种离散遗传测试用例生成方法，旨在模型在环测试层面揭示更多需求违例 [43]。

- 2019年，Zhang 等人指出，信息物理系统通常运行在高度不确定的环境中，因此在此类环境下进行测试存在诸多不确定性。为此，他们提出了两种基于不确定性理论和多目标搜索的测试用例生成方法 [44]。

- 2020 年，Menghi 等人指出，对于大量测试输入而言，计算密集型信息物理系统（CPSs）的计算过程非常耗时。因此，他们提出了 ARIsTEO，该方法利用单次仿真结果中的输入和输出构建代理模型，并利用证伪算法在这些代理模型中寻找违反需求的输入 [45]。

另一项重要的相关工作是 EPIcuRus 系统 [26]，它值得用一个独立的小节来介绍。

2.5.1 使用 *EPIcuRus* 进行开环建模与测试

闭环模型会与其自身组件进行交互，而开环建模则探索与其他系统之间的交互。上述所有系统（包括 GenClu）都假定用于测试闭环模型，而 EPIcuRus [26]则用于测试开环系统。EPIcuRus 将决策树分类器（用于探索可能满足需求的潜在假设）与模型检验（用于评估这些假设）相结合。在每次迭代中，决策树数据挖掘器返回的有希望的取值范围将用于生成下一次迭代的测试用例。

“测试”对 EPIcuRus 和 GenClu 来说含义不同。我们“测试”是为了从预言机那里获得一些分数。EPIcuRus“测试”则是为了探索模型内部和外部的假设空间。

尽管如此，EPIcuRus 和 GenClu 在内部共享一些相似的结构。两个系统都会运行一个学习器，然后查询输出。GenClu 使用半监督学习来

递归地将数据聚类，然后探测叶簇。EPIcuRus 使用一个完全监督的树学习器，随后查询其分支，以找出区分不同结果的属性范围（具体而言，EPIcuRus 的测试引擎会返回测试可能失败时所依据的假设）。

原则上，GenClu 可以扩展以实现类似 EPIcuRus 的功能（例如，如果我们运行决策树学习器来找出叶簇之间的差异）。但这属于未来的工作。

3 GenClu——采用聚类方法的测试套件生成方法

本节将介绍我们提出的测试套件生成框架 GenClu。不同于其他基于进化算法的方案以及需要为**所有**候选解标注的完全监督方案，GenClu 只需为**少量**候选解标注，具体方法是：(a) 对未标注示例进行聚类；(b) 使用模拟器为每个簇标注少量示例；(c) 在局部簇周围共享适应度值。

图 1 展示了我们所提出的聚类方法的整体框架。与上一节所述的方法截然不同，GenClu 的聚类方法不是通过多次迭代寻找最优解，而是在一次迭代中完成全部工作：

- 首先，我们按如下方式随机生成测试用例候选解。已知每个属性的最小值和最大值后，我们从该范围内随机选取数值。正如先前文献所指出的 [6], , 信号会在某些设定值处改变其数值。我们在这些设定值处的范围内生成数值，并将这些数值作为聚类算法的特征。请注意，此过程生成的是有效的测试用例输入，但其测试用例输出未知。

- 接下来，在 §2.1 的流形假设下，GenClu 将大量候选解（依据其独立属性）划分为若干小簇。我们使用递归 FASTMAP 聚类算法 [7], [8]进行聚类。

- 然后，我们只需模拟每个簇中的一个代表，就能找出最佳候选解组。我们的算法输出将是最佳组中的测试用例。

有关聚类的详细信息，请参见算法 1：

- 在< стиле id='2'>簇算法中，第 1 行检查当前候选解组的大小是否超过“足够”参数的值。如果大于“足够”，则需要再次对这些候选解进行簇划分。请注意，在我们的研究中，“足够”参数表示簇的大小。

算法 1 簇：用于构建树的聚类方法结构

要求： 测试套件、足够、树、节点
1: **如果大小**(测试套件) > 足够 **那么**
2: 东列表, 西列表 =**split**(测试套件) ▷ 两份项列表
3: 节点.left, 节点.right =东列表, 西列表
4: **簇**（东列表、足够、树、左节点）_ ▷ 递归
5: **簇**（西列表、足够、树、右节点） _ ▷ 递归
6: **结束 if**

- 第 2 行使用**拆分**算法将所有候选解分为两个箱。有关**拆分**的详细信息，请参见算法 2）。
- 第 3—5 行将两个箱放入树结构中，并递归地对新分组调用**簇**算法，直到无法再进行拆分。在我们的实验中，我们只评估最底层节点中的项，因为这些节点无法再进行拆分。

关于**划分**算法 2：

- 代码首先随机选择一个候选解作为枢轴点（第 1 行）。

- 之后，找到距离枢轴点最远的数据点，将其作为东点组代表（第 2 行）。
- 距离东点组代表最远的数据点被标记为西点组代表（第 3 行）。

- 之后，使用余弦定理将测试套件中的每个候选解分配到东点组或西点组（第 5 至 8 行）。此处使用的距离函数是欧几里得距离。

这种方法的一个有趣之处在于，聚类时无需比较所有数据点。因此，该算法无需进行 $O(N^2)$ 次距离计算，而是在完成第 2、3 行的 $O(2N)$ 次计算所需的时间内结束。构建完整的簇树后，我们随机选择一个代表性数据点来计算其目标值。之后，我们使用第 §2.4.1 节中提到的连续支配谓词对所有代表性数据点进行排序。包含排名第一的代表性数据点的簇将是最佳簇。

对于该算法，Chen 等人 [46]建议生成 10000 个初始候选解，然后将其剪枝为每个簇中的 $\sqrt{n}$个候选解。我们发现，对于为仿真模型生成测试用例而言，256 个初始随机样本就足以找到较优解（我们在尝试了 10000 个和 256 个初始样本集后得出了这一结论）。在使用 256 个样本时，我们还发现，Chen 等人关于将样本聚类缩减至 $\sqrt{n}$ 的建议不太实用；相比之下，将 **split**（来自算法 1）缩减至“足够数量”更为有效。我们将该数量设为 4、16 和 32，这更符合我们的直觉。（我们选择 4 作为测试套件规模，以检验算法能否有效生成较小的测试套件；同时也检验规模为 16 和 32 时，以确认用于生成测试套件的最佳算法是否能够

算法 2 划分：使用 FASTMAP 技术 [7], [8]，将候选解划分为东区域和西区域，返回东侧项目和西侧项目

Require: 测试套件
1: 枢轴点 =**随机**(测试套件) ▷枢轴点是任意点
2: 东点 = **mostDistance**(测试套件, 枢轴点) ▷ 距离枢轴点最远
3: 西点 =**mostDistance**(测试套件, 东点) ▷ 最远的东点
4: c = **距离**(东点, 西点)
5: **遍历**点于测试套件 **执行**
6: a =**距离**(点, 东点)
7: b =**距离**(点, 西点)
8: 点.d =$(a^2 + c^2 - b^2) / (2c)$
9: **结束循环**
10: 新建测试套件 =**排序**(测试套件.d) ▷按 d 对全部项排序
11: 西侧项目 = 新建测试套件[:0.5***大小**(测试套件)]
12: 西侧项目 = 新建测试套件[0.5***大小**(测试套件):]

较小的测试套件能在测试套件规模增加时完全检测出所有变异体）。

4 基线算法

为了进行比较，我们将 GenClu 系统与代表先前研究主要主题的最先进系统进行了比较：

- Matinnejad等人于2018年的OD进化编程系统 [6]；参见§4.1。
- Haq 等人在 2022年提出的 SAMOTA 基于代理模型的优化系统 [47]；参见 §4.2。

请注意，回顾 §2.5.1，我们不探讨 EPIcuRus，因为该系统中的测试性质与 GenClu、OD 和 SAMOTA 不同。

4.1 OD 测试用例生成

Matinnejad等人于 2019 [6]提出了输出多样性（OD）测试用例生成方法。该方法基于元启发式搜索，以发现仿真输出信号 [6]中的最大多样性。具体而言，OD首先将测试输入中允许的初始信号片段数量 P 设置为1，并将调整参数 σ 设置为0.5。随后，根据输入范围和初始信号片段数量 P生成初始随机测试套件。之后，在特定的超时值内（我们采用与先前工作相同的值，即600秒），OD算法重复执行以下步骤。

- 如果是第一次迭代，则直接跳过此步骤。否则，通过对已记录候选项 应用调整算法和当前调整参数σ来生成新的测试套件。这里的已记录候选项是指从先前迭代中保存的测试用例，而调整参数用于控制调整操作的缩放因子。
- 模拟新的测试套件 中的测试用例，以获取输出信号。
- 计算新测试套件中的测试用例所达到的覆盖率，并将其加入累积覆盖率。这里的覆盖率指 Simulink 中的结构覆盖率，例如“修正条件/判定覆盖”；累积覆盖率则是当前测试用例所达到的结构覆盖率。
- 计算新测试套件的输出多样性目标值 O。如果该目标值高于记录值，则用新测试套件替换已记录的测试套件，并记录最高目标值。
- 检查累积覆盖率是否已在低于%100的值上陷入停滞。如果是，则将信号片段的数量 P 增加 1。
- 如果当前步骤的累积覆盖率高于初始覆盖率，则将调整参数 σ 按比例从 σ-探索 (0.5) 降至 σ-利用 (0.01)。

4.2 SAMOTA

Haq 等人提出的 SAMOTA 算法 2022 [47]基于代理模型优化来优化目标。更具体地说，SAMOTA 主要由两种搜索算法组成，分别称为“全局搜索”和“局部搜索”。在全局搜索中，基于现有的测试用例训练全局代理模型，这些测试用例存储在数据库中。借助这些全局代理模型，全局搜索会为每个尚未覆盖的目标返回最佳测试用例（即适应度分数最高的测试用例） [47]。在全局搜索中探索适应度函数的搜索空间后，局部搜索会进一步利用全局搜索找到的那些有前景的区域。首先，局部搜索使用 HDBSCAN 根据候选解的适应度值对其进行分组。然后针对每个簇，利用该簇中的候选解训练局部代理模型。在这种情况下，局部搜索只需有限数量的代理模型即可探索所有有前景的区域 [47]。只要通过全局搜索和局部搜索为某个尚未覆盖的目标找到更好的测试用例，就会用它替换该目标对应的原始测试用例。当达到指定的迭代次数或超出测试预算时，搜索便会停止。原始 SAMOTA 是为二分类任务设计的（这意味着系统需要输入人工定义的违规判据）。为了在我们的实验中复现 SAMOTA，我们将该违规判据替换为随机生成的候选解的目标值。搜索过程中一旦发现更高的分数，该目标就会被标记为已覆盖。

5 实验设置

5.1 案例研究

对于本研究的案例研究，我们检索了（a）有关仿真模型测试的文献，以及（b）官方 Simulink 网站“MathWorks”。我们的目标是找到满足以下要求的案例研究。

- 模型必须能够在模型在环级别运行（这意味着无需任何硬件或传感器即可进行仿真）。
- 模型必须使用 Simulink 开发，因为我们的所有实验脚本都是用 Matlab 编写的，并且依赖于在 Simulink 中进行的仿真。
- 仿真模型必须使用离散求解器。连续求解器在相同输入下会产生长度不同的输出信号，因此我们无法对其进行变异测试。
- 仿真模型必须包含足够多可供变异的模块。我们将在以下某一节介绍如何对系统进行变异。

经过上述两项标准筛选后，我们找到 5 个可用于本研究的开源案例研究：

- ”Tiny”是一个简单的物理模型 [2]；
- ”Two tanks”模拟两个储罐的流入和流出流量 [48]；

表 2：案例研究摘要。表中汇总了输入和输出的数量（# I/O）、模块数量（# 模块）、原始变异体数量（# OM）、过滤后的变异体数量（# FM）以及变异体筛选百分比（ Filtered %）。

项目	# 输入/输出	# 模块	# OM	# FM	已过滤 %
Tiny	3/1	20	33	11	33%
Two tanks	11/7	449	190	15	8%
CC	6/2	82	35	12	34%
CLC	2/7	101	60	18	30%
CW	15/4	273	88	30	34%

- “CW”模拟四个汽车车窗的电气系统和机械结构 [2];
- ”CC”模拟汽车的巡航控制系统 [5]
- ”CLC”模拟旋转离合器系统。

这些系统总结于表 2。如我们所见，除 Two tanks 案例研究外，变异体筛选比例（预处理后的变异体数量 / 手动生成的变异体数量）均在 30%–35% 左右。Two tanks 中有许多算术运算，因此生成了许多变异体。下一小节 §5.2 将说明我们如何手动创建这些变异体，以及如何在将其应用于研究之前对其进行预处理。

在这五个案例研究中，有两个案例的输入由布尔值和数值组成。如果将布尔值直接用于第 §3 节所述的**拆分** 算法，距离的计算可能会出错。为解决这一问题，我们在**拆分** 算法中将所有布尔值视为 0 到 1 之间的数值。然后，对于每个布尔型输入，我们将常规输入模块替换为开关模块：如果输入值小于 0.5，开关模块的输出为 0；否则，输出为 1。

5.2 变异体生成

上文（§2.4.2）讨论的预言机问题使我们无法通过询问人工来评估模型输出。因此，我们采用了 S2.4.2 中介绍的变异测试方法。回顾一下，变异测试会在正确的模型中植入故障以生成变异体，并检查测试用例是否能够“杀死”这些变异体。当变异模型输出向量中的任一值与原始模型中对应的值不同时，该测试用例就“杀死”了变异体。

在本研究中，我们在文献 [49], [50], [51], [52], [6]中搜索了Simulink故障模式。我们发现了一些与Matinnejad等人在 2018 [6]中记录的变异非常相似的变异。这些Simulink故障模式列于表3中。

我们使用上述模式手动植入故障。植入故障时，我们会考虑对单个组件进行变异的所有可能性。例如，若某个组件是一个算术运算符，它将前两个输入相加，并减去第三个输入（++-），那么我们会考虑所有组合

表 3：文献 [49], [50], [51], [52], 中收集并在 [6] 中列出的Simulink故障模式。

模式	插图
常量值的变化	将常量值 c 更改为某个如果它是数值型，则为其他值。如果它是布尔型，将其取反。
算术运算符的变更运算符	切换 +/- 或将 + 替换为 ×。
关系运算符的变化tor	将 $\geq$ 切换为 $<$ 或 $\leq$，并将 $\leq$ 切换为 $>$或 $\geq$。
逻辑运算符的变化tor	切换**与**、**或**和**异或**；移除**非**或添加**非**。
连接错误	切换以下对象的输入线“切换”模块。
信号数据类型错误	将“double”数据类型改为“single”，或将“fixdt(0,8,3)”数据类型改为“fixdt(0,8,2)”。
初始条件错误和延迟值	更改“Integra-”中的初始值tion”和“Unit Delay”模块。

+ 和 - 的组合（这意味着它可以生成 8 个顺序不同的变异体 +/-），以及将所有这些输入相乘的情况（这意味着该组件可以生成 9 个变异体）。

在为单个案例研究生成所有变异体后，我们希望移除以下变异体：(a) 被所有测试用例杀死的变异体；(b) 任何测试用例都无法检测到的变异体；(c) 可由与其他变异体相同的测试用例检测到的变异体（例如，对于在所有测试用例上具有相同输出的变异体，我们只保留一个，舍弃其余变异体）。

我们使用 200 个随机测试用例对这些变异体进行了变异测试，并按照上述三条规则过滤掉变异体。原始变异体数量、过滤后的变异体数量以及过滤百分比显示在表 2 的最后三列中。尽管有些案例研究中的变异体数量较少，但我们发现很难检测到所有变异体，而这正是测试用例生成中的一个难题。

5.3 性能评价标准

为了评估每种测试用例选择技术的性能，我们采用变异分数这一指标。变异分数是所有变异体中被检测到的变异体数量所占的百分比（例如，如果在 10 个变异体中检测到 8 个变异体，则变异分数为 $8/10 = 0.8$）。变异分数越大，生成的测试用例就越好。

5.4 统计分析

为确保结果具有普遍性，我们对每种方法重复执行 20 次。为了比较不同测试用例生成方法在这 20 次重复中的表现，我们采用 Scott-Knott分析，根据测试用例生成方法的变异分数中位数对其进行排序。

具体而言，Scott-Knott 会递归地将候选解列表（c）划分为两个子列表（c_1 和 c_2），使划分前后的期望均值最大化 [53], [54], [55]:

$$E(\Delta) = \frac{\text{len}(c_1) * |\overline{c_1} - \overline{c}| + \text{len}(c_2) * |\overline{c_2} - \overline{c}|}{\text{len}(c)} \quad (1)$$

随后使用 Cliff's delta方法进行假设检验，以检查两个子列表之间的差异是否显著。delta 值为 $Delta = (\#(x > y) - \#(x < y))/(\text{len}(c_1) * \text{len}(c_2))$，其中 $\forall x \in c_1$ 和 $\forall y \in c_2$。具体来说，Cliff's delta 用于估计子列表 c_1 中的一个值大于子列表 c_2 中的一个值的概率，减去反向概率 [56]。如果 delta 不属于“小”效应（$Delta >= 0.147$） [57]，则两个子列表之间的差异显著。

我们选择 Scott-Knott 的原因是：(a) 它完全是非参数的；(b) 它能减少统计分析中潜在错误的数量，因为在对 $O(N^2)$ 进行分析时，最多只需要进行 $O(log2(N))$ 次统计检验。其他研究者也支持使用该检验 [58]，因为它克服了其他多重比较统计检验（例如 Friedman检验 [59]）的一项常见局限：这类检验会将处理分配到多个组中（使实验人员难以区分均值实际所属的组 [60]）。

项目	算法	测试套件规模 - 4		测试套件规模 - 16		测试套件规模 - 32		wins
		MS中位数	IQR	MS中位数	IQR	MS中位数	IQR	
Tiny	随机	0.73	0.55	0.82	0.09	0.91	0.09	1
	OD	0.82	0.18	1.00	0.09	0.91	0.09	**3**
	SAMOTA	0.73	0.18	-	-	-	-	0
	GenClu	0.82	0.09	0.91	0.09	0.91	0.09	**3**
CLC	随机	0.94	0.05	1.00	0.00	1.00	0.00	**3**
	OD	0.78	0.05	0.67	0.00	0.67	0.00	0
	SAMOTA	-	-	0.94	0.11	-	-	0
	GenClu	0.89	0.11	1.00	0.06	1.00	0.00	**3**
CC	随机	0.92	0.25	1.00	0.00	1.00	0.00	2
	OD	0.96	0.12	1.00	0.00	1.00	0.00	2
	SAMOTA	1.00	0.00	-	-	-	-	**1**
	GenClu	1.00	0.08	1.00	0.00	1.00	0.00	**3**
CW	随机	0.97	0.07	1.00	0.00	1.00	0.00	**3**
	OD	0.83	0.16	0.76	0.05	0.82	0.01	0
	SAMOTA	-	-	0.97	0.07	-	-	0
	GenClu	0.97	0.07	1.00	0.00	1.00	0.00	**3**
Twotanks	随机	0.87	0.26	1.00	0.07	1.00	0.00	**3**
	SAMOTA	-	-	1.00	0.13	-	-	**1**
	GenClu	0.80	0.40	1.00	0.27	1.00	0.00	**3**

表4：5 个案例研究的变异分数。比较采用 3 种不同的测试套件规模：4、16 和 32。该方法在小规模测试套件（即 4）中表现出色，意味着该方法生成的测试用例有效且高效。我们还检查了测试套件规模为 16 和 32 时的表现，以探究不同方法的性能收敛速率。浅灰色单元格标示了基于统计检验得出的最佳方法，最后一列汇总了该方法在统计检验中排名第

6 结果

在本节中，我们将探讨三个研究问题（RQ），并根据实验结果回答这些问题。

RQ1：在开源仿真模型上生成小型测试套件时，哪种方法效果最佳？为回答RQ1，我们分别运行（a）随机方法、（b）OD、（c）SAMOTA和（d）我们提出的聚类方法GenClu，每种方法重复20次，每次使用不同的随机种子。随后，我们应用第5.4节所述的Scott-Knott分析。如果两种算法的变异分数在Scott-Knott分析中的排名不同，我们就认为它们存在显著差异。

表4展示了不同算法的性能。请注意，SAMOTA仅在Tiny和CC的测试套件规模为4时，以及在CLC、CW和Twotanks的测试套件规模为16时有性能得分。原因是SAMOTA是一种基于代理模型的方法，它使用为每个未覆盖目标生成的代理模型来优化各个目标。因此，SAMOTA生成的测试用例数量主要取决于目标的数量。如第2节所述，我们实验中的目标数量为 4*输出信号数量。因此，在本实验中，SAMOTA的重复运行在不同案例研究中可以生成数量各异的测试用例。此外，由于Twotanks中的所有输入都是常量信号，OD算法不适用于Twotanks案例研究。因此，OD的调整算法不适用于该案例研究。

如上所述，为了确保分析完整，我们针对测试套件规模为 4、16 和 32 的情况重复了分析。尽管 4 是较小的测试套件规模，但在我们的案例研究中，GenClu 在这一规模下仍能在所有案例研究中取得最高的变异分数。对于 16 和 32，我们没有发现明显差异，因为大多数方法在所有研究中的变异分数都达到了 1.00。

从表4可以看出，当测试套件规模为4时（表4左侧部分），GenClu 在全部5个案例研究中的表现均最佳；随机方法仅在3个案例研究中胜出，而 OD 仅在1个案例研究中胜出。测试套件规模为4时，Tiny 和 CC 案例研究中也提供了 SAMOTA；与 GenClu 相比，SAMOTA 在 Tiny 中表现较差，在 CC 中表现相近。具体而言，测试套件中只有4个测试用例时，GenClu 能够检测出 CC 案例研究中的所有故障，而其他三种方法大多会漏掉少量故障。在另外四个案例研究中，尽管 GenClu 未能达到100%的变异分数，但其表现仍优于其他方法。

当测试套件规模增加到 16 时（表 4 中间部分），可以看到，与规模为 4 时相比，除 OD算法 外，所有方法的性能相近或有所提升。为了解释这一点，我们检查了仿真的日志信息，发现由于测试套件规模大幅增加，在给定的超时参数下，OD 的迭代次数远少于测试套件规模为 4 时的情况（例如，规模为 4 时迭代 30+ 次，而规模为 16 时仅迭代 8 到 10 次）。这表明 OD 在迭代次数较少时找到的信息量较低的候选解，因此测试套件规模增加到 16 时，其性能有所下降。在测试套件规模为 16 时，SAMOTA 仅在 Twotanks 案例研究中检测出所有变异体，并且在 CLC 和 CW 中的性能低于 GenClu。查看测试套件规模为 16 时的结果可见，与其他所有方法相比，GenClu 再次在所有案例研究中取得最高性能。

尽管我们发现，当测试套件规模为 16 时，大多数方法的性能已经趋于收敛至 100%，但我们仍然记录了测试套件规模为 32 时的结果。这是因为 OD 中每次迭代的传统样本量为 30，而 32 是最接近 30 且能够被 256 整除的数值。正如预期的那样，除 OD 外，其他方法要么在测试套件规模为 16 时已经收敛，因此表现相近；要么在测试套件规模为 16 时尚未收敛，因此性能有所提升。

综上所述，在测试套件规模为 4、16 或 32 的所有案例

测试套件规模	算法	Tiny		CLC		CC		CW		Twotanks	
		运行时间	较慢	运行时间	更慢	运行时间	较慢	运行时间	较慢	运行时间	较慢
4	随机	151s	1.0	293s	1.0	458s	1.0	2340s	1.0	950s	1.0
	OD	12861s	85.2	8323s	28.4	11495s	25.1	15381s	6.6	-	-
	SAMOTA	15500秒	102.6	-	-	63848秒	139.4	-	-	-	-
	GenClu	317s	2.1	481s	1.6	881s	1.9	3000秒	1.3	1418秒	1.5
16	随机	608s	1.0	1207秒	1.0	2363 秒	1.1	9024秒	1.0	3613秒	1.0
	OD	16659秒	27.4	18242s	14.4	21536s	10.4	35525s	3.9	-	-
	SAMOTA	-	-	432530s	358.4	-	-	91789s	10.2	117110秒	33.1
	GenClu	669s	1.1	1283秒	1.1	2062秒	1.0	10704秒	1.2	3538秒	1.0
32	随机	3061秒	1.0	4415秒	1.0	9041秒	1.0	32954秒	1.0	12845秒	1.0
	OD	20446秒	6.7	24253秒	5.5	119131秒	14.8	174183秒	5.3	-	-
	SAMOTA	-	-	-	-	-	-	-	-	-	-
	GenClu	4598s	1.5	5126s	1.2	8509s	1.1	43591s	1.3	13121s	1.0

表5：5个案例研究的运行时间比较。浅灰色单元格 提出的 GenClu 方法的运行时间比较。如果某种方法的运行时间接近基线方法“随机”，我们就认为它效率更高。由此可见，GenClu 的运行时间仅为“随机”方法的1至2倍（即最坏情况下慢2.1倍），而所有其他方法的运行时间都比“随机”方法慢数倍，甚至多达十倍。

研究中，GenClu 始终保持最高排名。测试套件规模为 32 时，随机方法的表现并不差；但在测试套件规模分别为 16 和 4 时，其表现分别在 1 个和 2 个案例研究中落后。由于在特定超时设置下可用的迭代运行次数较少，OD 在 CLC 和 CW 案例研究中测试套件规模为 4 时表现不佳，而在测试套件规模为 16 和 32 时表现更差。尽管我们建议随着测试套件规模增大而提高 OD 的超时时限，但这会使 OD 的总体运行时间大幅延长，并超出测试中的时间预算。总体而言，我们对 RQ1 的回答是：

在**所有**项目中，都能发现 GenClu 始终处于最佳排名。更具体地说，当测试套件规模为 4 时，在全部 5 个案例研究中，GenClu 是唯一一个获得最高变异分数的方法。当测试套件规模增加到 16 时，GenClu 能够检测出 Tiny 中 91% 的故障，以及其余 4 个案例研究中的全部故障。当测试套件规模为 32 时，GenClu 和随机方法在所有案例研究中的性能趋于一致，而 OD 在少数案例研究中仍然落后。总而言之，在不同测试套件规模下，GenClu 都是最有效的方法。

RQ2：哪种方法在开源仿真模型上生成小型测试套件时效率最高？ 为了回答 RQ2，我们记录了每种方法的总运行时间（算法运行时间加仿真运行时间再加变异测试时间）。为确保比较足够公平，我们在同一台 64 位 Windows 10 计算机上运行了这三种算法，该计算机配备 4.2 GHz 8 核 Intel Core i7 处理器和16GB的内存。

这些运行时间见表5。此外，对于每个案例研究，我们总结了算法比最快算法慢多少倍。正如我们观察到的，在大多数情况下，基线随机方法是最快的，因为它不需要获取信息。因此，“慢于”列中的所有倍数基本上都以随机方法的运行时间为基准。

如表5所示，GenClu 的性能优于随机方法（正如我们在 RQ1 中所展示的），且所需时间仅为随机方法的 1 至 2 倍。此外，与当前最先进的方法相比，GenClu 的运行速度比 OD 快 4 至 41 倍，比 SAMOTA 快 40 至 300 倍。正如 GenClu 运行速度远快于其他三种方法：

- OD 通过多轮迭代调整输入信号，并在每轮迭代中计算输出多样性。每一代都需要进行多次仿真。
- SAMOTA 为每个尚未覆盖的目标生成代理模型，并在多轮迭代中根据代理模型优化各个目标。因此，随着目标数量增加，SAMOTA 的运行时间将呈指数级增长。
- 另一方面，GenClu 无需仿真即可将随机生成的大型初始样本划分为不同的簇，并且每组只需进行一次仿真就能找到最佳簇（由于我们使用余弦规则划分候选解，同一簇中的候选解彼此接近）。在这种情况下，GenClu 只需较少的仿真次数就能找到最优解。

总之，我们对 RQ2 的回答如下：

在**所有**项目中，GenClu 的运行速度比所有其他最先进的方法都快数倍（例如，比 OD 快 4–41 倍，比 SAMOTA 快 40–300 倍）。更重要的是，GenClu 仅比无需信息增益的基线随机方法慢 1–2 倍。因此，我们可以得出结论：GenClu 是一种用于仿真模型的高效测试用例生成方法。

RQ3：未来的测试能否受益于 GenClu 生成的树结构？ 为回答 RQ3，我们首先对 GenClu 生成的树结构进行实证验证，然后实现了差分进化算法中使用的变异算子，以对测试用例进行变异。

首先，我们需要验证 GenClu 排序后的簇是否具有信息价值。具体而言，排名较高的簇更有可能具有较高的变异分数，而排名较低的簇更有可能具有较低的变异分数。为此，我们首先运行常规的 GenClu 执行流程，该流程会生成树结构，并将样本划分为 64 个簇。随后，常规的 GenClu 流程会根据从这些簇中随机抽取的代表样本的反模式值，对这 64 个簇进行排序。为了验证这种排序系统是否具有信息价值，我们接着执行一次暴力仿真。

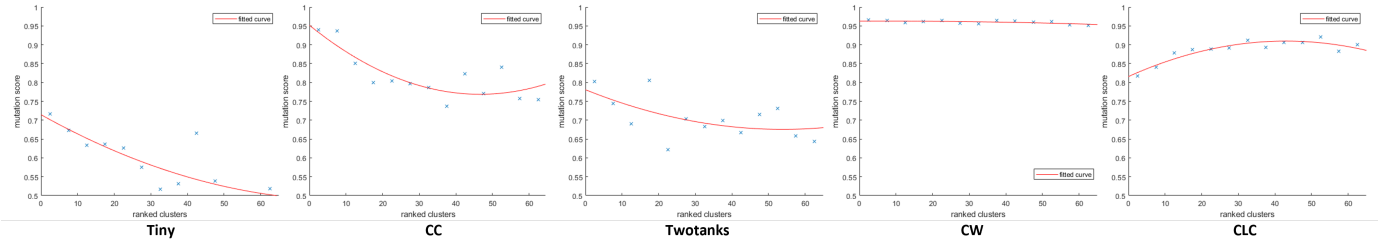


图 4：散点图展示了 GenClu 排序后的所有簇的暴力变异分数。沿 x 轴从左到右排列的散点表示从最高排名到最低排名，y 轴表示变异分数。为避免散点过于密集，我们绘制每五个簇的平均变异分数，并根据这些散点绘制最佳拟合曲线。

该方法会为每个簇模拟所有候选解，并在图4的散点图中展示各簇的变异分数。请注意，x 轴显示的是按正常 GenClu 执行结果排序的簇（靠近原点的簇具有较高的反模式值，而靠近 x 轴末端的簇具有较低的反模式值）。y 轴显示通过模拟相应簇中的所有测试用例得到的平均变异分数。为了展示变异分数从优质簇到劣质簇的变化趋势，我们每五个簇取一次平均值，并根据这些平均值生成最佳拟合曲线。如图4所示，Tiny 和 CC 中拟合曲线的斜率非常大，这表明 GenClu 排序得到的簇包含了丰富的信息。尽管 CW 和 Twotanks 中拟合曲线的斜率低于 Tiny 和 CC 中的斜率，但仍能表明前几个簇的表现优于其他簇。唯一的例外是 CLC，其拟合曲线的斜率与其他案例研究相反。这表明，在 5 个案例研究中，有 4 个案例研究生成并排序的簇包含丰富的信息，能够引导算法生成最佳测试用例。

在验证了 GenClu 生成的排序簇具有信息价值后，我们接着探讨 GenClu 生成的簇结构能否使后续测试受益。EPIcuRus 的结论（如上所述）是，某些输入数据范围比其他范围更具信息价值 [26]。受其研究启发，我们的实证验证结果表明，GenClu 排名前几位的簇能够探索输入空间中信息量最大的区域。然而，面临的挑战是，GenClu 并不会生成信息量最大的范围，而只是将数据划分为不同的簇，其中每个簇内唯一的信息就是该簇中的数据点。为应对这一挑战，我们采用差分进化算法中的变异操作。差分进化是一种通过迭代改进候选解来优化问题的方法。它巧妙地利用个体之间的差异，通过简单快速的线性算子在每次迭代中对候选解进行变异，从而找到全局最优解 [61]。在 GenClu 结构中，我们采用差分进化算法中的变异算子，对排名靠前的几个簇中的候选解进行变异，以生成信息量更大的测试用例。更具体地说，差分进化算法

案例研究	原始分数	改进	改进后分数
Tiny	0.599	+0.242	0.841
CC	0.808	+0.072	0.880
CW	0.958	+0.017	0.975
Twotanks	0.744	+0.051	0.795
CLC	0.881	-0.026	0.855

表 6：从原始随机生成的测试用例到变异测试用例的平均改进。

按照以下公式对数据点进行变异：

$$p_{new,i} = \begin{cases} a_i + F * (b_i - c_i) & \text{if } r_i < CR \text{ or } i = R \\ p_{old,i} & \text{O.W.} \end{cases} \quad (2)$$

其中

- a， b， c是种群中的三个不同候选解。
- r_i 是一个均匀随机数 $\in (0, 1]$。
- R是一个随机索引 $\in \{1, 2, \cdots, n\}$。
- F 和 CR 是变异算子的参数。我们选择 $F = 0.9$ 进行大规模变异，并选择 $CR = 1.0$以确保候选解中的每个元素都发生变异。

表6展示了从所有原始测试用例到变异测试用例的变异分数平均改进情况。如我们所见，四个案例研究在变异测试方面的分数都有所提升。正如预期，Tiny和CC的改进幅度较大，因为图4中的最佳拟合曲线斜率更大。CW和Twotanks的改进幅度较小，原因相同。

总之，我们对 RQ3 的回答如下。

GenClu 生成的簇结构具有信息性，其中最优的几个簇能够反映输入空间中更多的缺陷区域。此外，通过利用差分进化算法中的变异算子，我们可以对这些簇进行变异，生成变异分数更高的新测试用例，从而有利于后续测试。

7 有效性威胁

在本节中，我们将讨论 Feldt等人 [62]提出的有效性威胁问题。

构念效度 主要体现在算法的参数设置上。不同的参数可能会导致截然不同的结果。例如，我们对 EPIcuRus 的重复实验使用了 30 次迭代来生成最佳

输入范围。在不同的案例研究中，这个数值很重要，可能需要适当调整。此外，在我们提出的聚类方法中，我们随机生成 256 个初始样本集，并将每个箱中的样本聚类为 4、16 和 32 个。在不同的评估标准或案例研究中，需要根据研究的要求和目的调整这些数值。使用不同的参数可能会导致基于这些参数得出的观察结果有所不同。未来工作中，我们应考虑进行超参数优化调优 [63], [64] ，以减轻这一威胁。

结论效度主要与算法的随机变化以及获取真实故障的途径有关。为降低这一威胁的影响，我们将所有实验重复运行 20 次，以确保结果具有普遍性，并使用 Scott-Knott 统计分析评估这 20 次重复实验的性能。此外，由于我们研究中的所有案例研究都来自开源项目，这些软件中没有公开的真实故障。为应对这一威胁，我们更全面地讨论了以往文献中出现的故障模式，并将其整理成表格（见第 §5.2 节）。随后，我们生成了覆盖该表中所有故障模式的变异体。在这一步中，我们确保生成的变异体尽可能接近真实故障，并使用这些变异体来更好地应对因无法获取真实故障而产生的威胁。

内部效度关注的是处理方式是否正确导致了相应结果。为降低这一威胁的影响，针对不同算法的所有实验均在相同的案例研究上运行。此外，我们在相同规模的测试套件上比较了三种方法。这里还需要指出另一个威胁：我们对效率的结论是基于实验中所采用案例研究的执行情况得出的。由于本研究中的系统可能不需要很长的仿真时间，因此运行时间可能因案例研究而异。然而，我们的结论充分讨论了各算法所需仿真次数方面的效率。因此，从理论上讲，将我们的算法应用于更复杂的仿真模型时，可以减少仿真工作量。

外部效度涉及本研究在其他问题中的应用。我们使用五个开源仿真模型来验证本研究的结论。这可能构成一项威胁：五个模型无法代表仿真模型的整体情况。然而，我们从（a）以往文献和（b）Simulink 官方网站中搜集这些案例研究，并筛选出不适用于本研究的案例（关于案例研究选择的详细信息，请参见第 §5.1 节）。此外，仿真模型难以收集，因此以往大多数研究仅使用了 2–4 个模型（而且其中大多数是私有模型）。因此，我们利用现有资源，在本研究中尽可能妥善地应对这一威胁。另一项潜在威胁是将本研究中的算法应用于其他类型的研究。我们复现的算法 EPIcuRus 和提出的聚类算法都需要针对不同的目标函数进行修改。未来，我们计划将这些技术应用于更多仿真模型，以减轻这一威胁。此外，系统覆盖率、系统需求和环境不确定性等信息对研究人员而言可能无法获取，但对于正在开发 SUT 的实践者而言却可能是可获取的。因此，这些信息的可获取性可能构成本研究的一项威胁。

8 结论与未来工作

在本研究中，我们（a）采用输出信号上的反模式值这一理念，以发现仿真模型中潜在的故障；（b）探索将半监督方法应用于测试用例生成。“流形假设”——即高维数据可以近似映射到较低维流形上，而不会造成太多信号 [65],损失——是我们所提方法取得成功的关键因素。

我们在 5 个使用 Simulink 设计的仿真模型上评估了 GenClu，并将其与以下方法进行了比较：(a) 基线随机方法；(b) 改进后的最新方法 EPIcuRus，该方法将新的测试用例生成算法纳入其功能；(c) 另一种最新方法 OD测试用例生成；以及 (d) 面向 DNN 赋能系统的最新在线测试框架 SAMOTA。通过重复运行实验 20 次，我们发现，基于半监督方法的方法GenClu 在变异测试中能够以显著更高的性能生成测试用例（在全部 5 项案例研究中均胜出，并通过统计检验）。综合结果来看，在预言机条件较少的情况下，GenClu 是一种有效且高效的方法，可生成用于检测故障的测试套件。

未来工作中，我们计划：(a) 从其他资源收集更多仿真模型，并在其他编程语言中复现本研究；(b) 发现更多可能揭示仿真模型中故障的潜在反模式；(c) 通过在分割算法中加入一些启发式方法来增强 GenClu；(d) 将文献 [46]中的二进制分割算法应用于当前的连续分割算法，以更好地处理二进制输入。

对于探索这一领域或软件工程其他领域的研究人员，我们有以下建议：我们并非总是需要探索状态空间的所有部分。有时（如本文所示），稍作几次（$\sqrt{n}$）探查就能达到同样的效果，而且速度快得多。

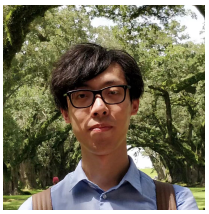
参考文献

[1] A. Arrieta、S. Wang、G. Sagardui 和 L. Etxeberria，“面向基于仿真的验证，信息物理系统产品线的搜索式测试用例选择”，载于第*20*届国际系统与软件产品线会议论文集，2016年，第297–306页。[2] A. Arrieta、S. Wang、U. Markiegi、A. Arruabarrena、L. Etxeberria 和 G. Sagardui，“面向基于仿真的测试的帕累托高效多目标黑盒测试用例选择”，信息与软件技术，第114卷，第137–154页，2019年。[3] C. A. Gonz´alez、M. Varmazyar、S. Nejati、L. C. Briand 和 Y. Isasi，“实现信息物理系统的模型测试”，载于第*21*届*ACM/IEEE*国际模型驱动工程语言与系统会议论文集，2018年，第176–186页。[4] G. Sagardui、J. Agirre、U. Markiegi、A. Arrieta、C. F. Nicol´as 和 J. M. Mart´ın，“Multiplex：一种用于电梯验证的协同仿真架构”，载于*2017*年*IEEE*电子、控制、测量、信号及其在机电一体化中的应用国际研讨会（*ECMSM*）。IEEE，2017年，第1–6页。[5] A. Arrieta、S. Wang、U. Markiegi、G. Sagardui 和 L. Etxeberria，“信息物理系统的搜索式测试用例生成”，载于*2017*年*IEEE*进化计算大会（*CEC*）。IEEE，2017年，第688–697页。[6] R. Matinnejad、S. Nejati、L. C. Briand 和 T. Bruckmann，“具有动态行为的Simulink模型的测试生成与测试优先级排序”，*IEEE*软件工程汇刊，第45卷，第9期，第919–944页，2018年。[7] C. Faloutsos 和 K.-I. Lin，“Fastmap：一种用于传统及多媒体数据集索引、数据挖掘和可视化的快速算法”，载于*1995*年*ACM SIGMOD*数据管理国际会议论文集，1995年，第163–174页。

[8] J. Platt，“Fastmap、metricmap 和 landmark mds 均为 nystr ¨ om 算法”，载于《人工智能与统计国际研讨会》。PMLR，2005年，第261–268页。[9] H. Tu 和 T. Menzies，“Frugal：为软件分析解锁半监督学习”，载于《2021年第36届IEEE/ACM自动化软件工程国际会议（ASE）》。IEEE，2021年，第394–406页。[10] E. Kocaguneli、T. Menzies、J. Keung、D. Cok 和 R. Madachy，“主动学习与工作量估算：发现软件工作量估算数据中的核心内容”，《IEEE软件工程汇刊》，第39卷，第8期，第1040–1053页，2012年。[11] Z. Yu、F. Fahid、T. Menzies、G. Rothermel、K. Patrick 和 S. Cherian，“Terminator：更好的自动化UI测试用例优先级排序”，载于《2019年第27届ACM欧洲软件工程会议与软件工程基础研讨会联合会议论文集》，2019年，第883–894页。[12] Z. Yu、N. A. Kraft 和 T. Menzies，“为加快文献综述寻找更好的主动学习器”，《Empirical Software Engineering》，第23卷，第3161–3186页，2018年。[13] A. Arrieta、S. Wang、U. Markiegi、G. Sagardui 和 L. Etxeberria，“采用多目标搜索，增强工业信息物理系统测试中的反应式测试用例生成与优先级排序”，《IEEE Transactions on Industrial Informatics》，第14卷，第3期，第1055–1066页，2017年。[14] L. Cayton，“流形学习算法”，加州大学圣迭戈分校技术报告，第12卷，第1-17期，第1页，2005年。[15] A. N. Kiaer，“关于代表性计数的观察与经验”，《国际统计学会公报》，第9卷，第2期，第176–183页，1895年。[16] H. Taherdoost，“研究方法中的采样方法；如何为研究选择采样技术”，《如何为研究选择采样技术》（2016年4月10日），2016年。[17] K. J. ˚Astr¨om 和 R. M. Murray，《反馈系统：献给科学家与工程师的导论》。普林斯顿大学出版社，2021年。[18] X. Ling 和 T. Menzies，“使用doless（最小二乘近似支配法）加速多目标仿真测试”，arXiv预印本 arXiv:2112.01598，2021年。[19] R. Matinnejad、S. Nejati、L. C. Briand 和 T. Bruckmann，“针对时间连续Simulink模型的自动化测试套件生成”，载于《第38届软件工程国际会议论文集》，2016年，第595–606页。[20] S. A. Chowdhury、S. Mohian、S. Mehra、S. Gawsane、T. T. Johnson 和 C. Csallner，“使用slforge在商业信息物理系统开发工具链中自动查找缺陷”，载于《第40届软件工程国际会议论文集》，2018年，第981–992页。[21] W. E. Wong、J. R. Horgan、A. P. Mathur 和 A. Pasquini，“测试集规模最小化与故障检测效果：一项航天应用案例研究”，载于《第二十一届年度国际计算机软件与应用会议（COMPSAC’97）论文集》。IEEE，1997年，第522–528页。[22] W. E. Wong、J. R. Horgan、S. London 和 A. P. Mathur，“测试集最小化对故障检测效果的影响”，《Software: Practice and Experience》，第28卷，第4期，第347–369页，1998年。[23] B. S. Ahmed，“一种用于配置感知结构测试的故障检测与组合优化技术测试用例最小化方法”，《Engineering Science and Technology, an International Journal》，第19卷，第2期，第737–753页，2016年。[24] D. Di Nardo、N. Alshahwan、L. Briand 和 Y. Labiche，“ 基于覆盖率的回归测试用例选择、最小化与优先级排序：一项工业系统案例研究”，《Software Testing, Verification and Reliability》，第25卷，第4期，第371–396页，2015年。[25] S. Yoo 和 M. Harman，“ 回归测试最小化、选择与优先级排序：综述 ”，《Software Testing, Verification and Reliability》，第22卷，第2期，第67–120页，2012年。[26] K. Gaaloul、C. Menghi、S. Nejati、L. C. Briand 和 D. Wolfe，“使用机器学习挖掘软件组件的假设”，载于《第28届ACM欧洲软件工程会议与软件工程基础研讨会联合会议论文集》，2020年，第159–171页。[27] R. Matinnejad、S. Nejati 和 L. C. Briand，“混合Simulink/Stateflow控制器的自动化测试：工业案例研究”，载于《2017年第11届软件工程基础联合会议论文集》，2017年，第938–943页。[28] R. Matinnejad、S. Nejati、L. C. Briand 和 T. Bruckmann，“适用于混合离散-连续Stateflow控制器”，载于《2015年第10届软件工程基础联合会议论文集》，2015年，页码84–95。[29] S. Wang、S. Ali 和 A. Gotlieb，“使用基于权重的遗传算法最小化软件产品线中的测试套件”，载于《第15届遗传与进化计算年度会议论文集》，2013年，页码1493–1500。[30] E. Zitzler 和 S. K¨unzli，“多目标搜索中的基于指标的选择”，载于《计算机科学讲义》。施普林格·柏林·海德堡，2004年，页码832–842。 [在线]。网址：https://doi.org/ 10 1007 978 3 540 30217 9 84 。 /

[31] - - - - A. S. Sayyad 、 T. Menzies 和 H. Ammar，¯“ 基于搜索的软件工程中用户偏好的价值：软件产品线案例研究”，载于《2013年第35届国际软件工程大会（ICSE）》。IEEE，2013年，页码492–501。[32] T. Wagner、N. Beume 和 B. Naujoks，“多目标优化中的帕累托法、聚合法和基于指标的方法”，载于《进化多准则优化》，S. Obayashi、K. Deb、C. Poloni、T. Hiroyasu 和 T. Murata 编。柏林，海德堡：施普林格·柏林·海德堡，2007年，页码742–756。[33] Y. Jia 和 M. Harman，“变异测试的发展分析与综述”，《IEEE软件工程汇刊》，第37卷，第5期，页码649–678，2010年。[34] M. Papadakis、M. Kintis、J. Zhang、Y. Jia、Y. Le Traon 和 M. Har- man，“变异测试进展：分析与综述”，载于《计算机进展》。Elsevier，2019年，第112卷，页码275–378。[35] A. V. Pizzoleto、F. C. Ferrari、J. Offutt、L. Fernandes 和 M. Ribeiro，“降低变异测试成本的技术与度量的系统性文献综述”，《系统与软件杂志》，第157卷，文章编号110388，2019年。[36] D. Schuler 和 A. Zeller，“Javalanche：面向 Java 的高效变异测试”，载于《第7届欧洲软件工程会议与 ACM SIGSOFT 软件工程基础研讨会联合会议论文集》，2009年，页码297–298。[37] A. J. Offutt、J. Pan、K. Tewary 和 T. Zhang，“ 数据流测试与变异测试的实验评估 ”，《软件：实践与经验》，第26卷，第2期，页码165–176，1996年。[38] R. Just、D. Jalali、L. Inozemtseva、M. D. Ernst、R. Holmes 和 G. Fraser，“变异体能否作为软件测试中真实故障的有效替代品？ ”，载于《第22届 ACM SIGSOFT 软件工程基础国际研讨会论文集》，2014年，页码654–665。[39] Q. Zhao、B. H. Krogh 和 P. Hubbard，“为嵌入式控制系统生成测试输入”，《IEEE控制系统杂志》，第23卷，第4期，页码49–57，2003年。[40] M. F. Bashir 和 S. H. K. Banuri，“自动化的基于模型的软件测试数据生成系统”，载于《2008年第4届新兴技术国际会议》。IEEE，2008年，页码275–279。[41] R. Matinnejad、S. Nejati、L. Briand、T. Bruckmann 和 C. Poull，“使用搜索对连续控制器进行自动化模型在环测试”，载于《基于搜索的软件工程国际研讨会》。Springer，2013年，页码141–157。[42] R. Sinha、C. Pang、G. S. Mart´ınez、J. Kuronen 和 V. Vyatkin，“面向工业信息物理系统的需求辅助自动测试用例生成”，载于《2015年第20届复杂计算机系统工程国际会议（ICECCS）》。IEEE，2015年，页码198–201。[43] A. Turlea，“面向信息物理系统的基于搜索的模型在环测试”，载于《2018年IEEE第16届嵌入式与普适计算国际会议（EUC）》。IEEE，2018年，页码22–28。[44] M. Zhang、S. Ali 和 T. Yue，“信息物理系统的不确定性感知测试用例生成与最小化”，《系统与软件杂志》，第153卷，页码1–21，2019年。[45] C. Menghi、S. Nejati、L. Briand 和 Y. I. Parache，“计算密集型信息物理模型的近似细化测试：一种基于系统辨识的方法”，载于《2020年IEEE/ACM第42届国际软件工程大会（ICSE）》。IEEE，2020年，页码372–384。[46] J. Chen、V. Nair、R. Krishna 和 T. Menzies，“将‘采样’作为基于搜索的软件工程的基线优化器”，《IEEE软件工程汇刊》，第45卷，614，2018 6，597第期，页码– 。[47] F. U. Haq、D. Shin 和 L. Briand，“使用代理辅助多目标优化对深度神经网络赋能系统进行高效在线测试”，载于《第44届国际软件工程会议论文集》，2022年，页码811–822。[48] C. Menghi、S. Nejati、K. Gaaloul 和 L. C. Briand，“为 Simulink模型生成自动化在线测试预言机，采用连续…… …

“连续且不确定的行为”，载于《2019年第27届ACM欧洲软件工程会议与软件工程基础研讨会联合会议论文集》，2019年，第27–38页。[49] N. T. Binh 等，“Simulink模型的变异算子”，载于《2012年第四届知识与系统工程国际会议》。IEEE，2012年，第54–59页。[50] A. Brillout、N. He、M. Mazzucchi、D. Kroening、M. Purandare、P. R¨u mmer 和 G. Weissenbacher，“基于变异的Simulink模型测试用例生成”，载于《组件与对象形式化方法国际研讨会》。Springer，2009年，第208–227页。[51] Y. F. Yin、Y. B. Zhou 和 Y. R. Wang，“Simulink模型变异算子的研究与改进”，载于《应用力学与材料》，第687卷。Trans Tech Publ，2014年，第1389–1393页。[52] Y. Zhan 和 J. A. Clark，“基于搜索的Simulink模型变异测试”，载于《第7届遗传与进化计算年度会议论文集》，2005年，第1061–1068页。[53] H. Tu、Z. Yu 和 T. Menzies，“使用Emblem实现更好的数据标注（及其对缺陷预测的影响）”，TSE，2020年。[54] T. Xia、R. Krishna、J. Chen、G. Mathew、X. Shen 和 T. Men-zies，“工作量估算的超参数优化”，arXiv预印本 arXiv:1805.00336，2018年。[55] H. Tu、G. Papadimitriou、M. Kiran、C. Wang、A. Mandal、E. Deel-man 和 T. Menzies，“异常数据传输的工作流挖掘”，载于《2021年IEEE/ACM第18届国际软件存储库挖掘会议（MSR）》，2021年，第1–12页。[56] G. Macbeth、E. Razumiejczyk 和 R. D. Ledesma，“Cliff’s delta计算器：用于两组观测值的非参数效应量程序”，《Universitas Psychologica》，第10卷，第2期，第545–555页，2011年。[57] M. R. Hess 和 J. D. Kromrey，“效应量的稳健置信区间：非正态性与异方差条件下 Cohen’s d 和 Cliff’s delta 的比较研究”，载于美国教育研究协会年会。Citeseer，2004年，第1–30页。[58] C. E. Gates 和 J. D. Bilbro，“用于均值分离的聚类分析方法示例1”，《农学杂志》，第70卷，第3期，第462–465页，1978年。 [在线]。可访问：https://acsess.onlinelibrary.wiley.com/doi/abs/10.2134/agronj1978.00021962007000030024x [59] M. Friedman，“使用秩来避免方差分析中隐含的正态性假设”，《美国统计学会杂志》，第32卷，第200期，第675–701页，1937年。[60] S. G. Carmer 和 W. M. Walker，“农学研究中处理均值的成对多重比较”，《农学研究中处理均值的成对多重比较》，春季刊，1985年。[61] V. Feoktistov，《差分进化》。Springer，2006年。[62] R. Feldt 和 A. Magazinius，“实证软件工程研究中的效度威胁——初步调查”。载于 SEKE，2010年，第374–379页。[63] H. Tu 和 V. Nair，“调优虽好，但没有哪个调优器是最好的”，FSE SWAN，2018年。[64] H. Tu、G. Papadimitriou、M. Kiran、C. Wang、A. Mandal、E. Deel-man 和 T. Menzies，“异常数据传输的工作流挖掘”，载于《2021年IEEE/ACM第18届国际软件存储库挖掘会议（MSR）》。IEEE，2021年，第1–12页。[65] O. Chapelle、B. Sch¨olkopf 和 A. Zien 编，《半监督学习》。麻省理工学院出版社，2006年。 [在线]。可访问：http://dblp.uni-trier.de/db/books/collections/CSZ2006.html



Xiao Ling 是北卡罗来纳州立大学计算机科学专业的一名四年级博士生。他的研究兴趣包括面向软件工程的自动化软件测试和机器学习。



TimMenzies（IEEE会士，博士，新南威尔士大学，1995年）是美国北卡罗来纳州立大学计算机科学教授，主要教授软件工程、自动化软件工程和编程语言。他的研究兴趣包括软件工程（SE）、数据挖掘、人工智能、基于搜索的软件工程以及开放获取科学。更多信息，请访问 http://timm.fyi。