计算机科学
COMPUTER SCIENCE

融合深度强化学习和图卷积神经网络的类集成测试序列生成方法

王晨源，张艳梅，袁冠

引用本文

王晨源，张艳梅，袁冠. 融合深度强化学习和图卷积神经网络的类集成测试序列生成方法[J]. 计算机科学, 2025, 52(6): 58-65.

WANG Chenyuan, ZHANG Yanmei, YUAN Guan. Class Integration Test Order Generation Approach Fused with Deep Reinforcement Learning andGraph Convolutional Neural Network [J]. Computer Science, 2025, 52(6): 58-65.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240700115

融合深度强化学习和图卷积神经网络的类集成测试序列生成方法

王晨源　张艳梅　袁　冠
中国矿业大学计算机科学与技术学院　江苏 徐州 221116
中国矿业大学矿山数字化教育部工程研究中心　江苏 徐州 221116
(chenyuan_wang@cumt.edu.cn)

摘　要　类集成测试确保软件系统中多个类之间正常交互和协作,合理的类集成测试序列可以降低测试成本。为了降低程序中类集成测试序列的测试成本,国内外研究人员提出了多种类集成测试序列生成方法,但已有的方法生成的类集成测试序列的测试成本过高。针对上述问题,提出一种融合深度强化学习和图卷积神经网络的类集成测试序列生成方法。该方法首先将图卷积神经网络作为深度强化学习中的神经网络部分,并对智能体的网络结构和环境状态等方面进行改进,使环境和智能体可以基于图结构的数据进行交互;然后通过设计强化学习中的动作空间和奖励函数等基本要素,完成类集成测试序列的生成场景;最终实现智能体在不断地学习和尝试中得到最佳的类集成测试序列。实验结果表明,在以总体测试桩复杂度作为度量指标时,该方法能够在一定程度上降低生成类集成测试序列所需的测试桩代价。

关键词:类集成测试序列;深度强化学习;图卷积神经网络;测试桩;测试桩复杂度
中图分类号　TP311

Class Integration Test Order Generation Approach Fused with Deep Reinforcement Learning and Graph Convolutional Neural Network

WANG Chenyuan,ZHANG Yanmei and YUAN Guan
School of Computer Science and Technology,China University of Mining and Technology,Xuzhou,Jiangsu 221116,China
Mine Digitization Engineering Research Center of the Ministry of Education,China University of Mining and Technology,Xuzhou,Jiangsu 221116,China

Abstract　Class integration testing ensures normal interaction and collaboration between multiple classes in the software system and a reasonable class integration test order can reduce testing costs. Therefore,in order to reduce the testing cost of class integration test orders in programs,domestic and foreign researchers have proposed a variety of methods for generating class integration test orders. However,the testing cost of class integration test orders generated by existing methods is too high. To solve this problem,a class integration test order generation approach combining deep reinforcement learning and graph convolutional neural network is proposed. This approach first uses graph convolutional network as the neural network part of deep reinforcement learning,and improves the network structure of the agent and environmental status,so that the environment and the agent can interact based on graph-structured data,and then through design the basic elements such as action space and reward function in reinforcement learning,and complete the generation scenario of the class integration test order. Ultimately,the agent can obtain the best class integration test order through continuous learning and trying. Experimental results show that when the overall stubbing complexity is used as the evaluation metric,this approach can reduce the stubbing cost required to generate class integration test order to a certain extent.

Keywords　Class integration test order,Deep reinforcement learning,Graph convolutional neural network,Test stubs,Stubbing complexity

到稿日期:2024-07-17　返修日期:2024-09-04
基金项目:徐州市科技计划项目(KC22047);徐州市重点研发计划(社会发展)项目(KC23296);国家级大学生创新创业训练计划项目(202010290060Z)
This work was supported by the Xuzhou Science and Technology Project(KC22047),Xuzhou Key R&D Program(KC23296) and National College Students' Innovation and Entrepreneurship Training Program(202010290060Z).
通信作者:张艳梅(ymzhang@cumt.edu.cn)

1 引言

软件测试是软件开发过程中的一个重要阶段，对提高软件的质量和稳定性具有重要意义。其中软件测试包括单元测试、集成测试、系统测试以及回归测试等阶段[1]。在实际的软件系统中，某些类可能比其他类更为重要，它们与其他类的关联程度可能更紧密。如果这些重要的类在测试时出现错误，那么错误传递给其他类的可能性也会更大，从而造成不同的类集成测试序列(Class Integration Test Order，CITO)存在不同的测试成本。Kung 等[2]在 Interviews 系统中对随机生成和最优的类集成测试序列分别构建测试桩时，随机生成的类集成测试序列需要构建 191 个测试桩，花费 152 h；而最优的类集成测试序列仅需构建 8 个测试桩，花费 7 h。因此，如何生成一个合理的类集成测试序列对降低系统的测试成本具有重要意义。其中，为了降低程序中类集成测试序列的测试成本，国内外研究人员提出了多种类集成测试序列的生成方法，主要包括：基于图论的方法、基于搜索的方法以及基于强化学习的方法。

基于图论的方法最初由 Kung 等[2]提出，他们首先将类间关系转化为对象关系图(Object Relation Diagram，ORD)，然后利用逆向拓扑排序获得类集成测试序列。Jiang 等[3]构建了类间传递依赖模型，提高了生成 CITO 的效率。

基于搜索的方法可以分为两种类型，包括基于线性加权的启发式算法和基于帕累托模型的启发式算法。在基于线性加权的启发式算法中，Briand 等[4]提出了基于遗传算法的方法，将耦合度量和遗传算法相结合来计算构造测试桩代价，并通过不断迭代来获得最优的类集成测试序列。Zhang 等[5]提出基于粒子群优化算法的解决方案。在基于帕累托模型的启发式算法中，Cabral 等[6]提出一种基于帕累托模型的蚁群算法，算法中的蚂蚁将趋向于低代价的测试桩，通过迭代逐渐获得最优类集成测试序列。

随着强化学习[7]的广泛应用，Czibula 等[8]首次提出基于强化学习的解决方法，通过智能体和环境不断交互选出类集成测试序列，并生成所需测试桩个数较低的测试序列。Ding 等[9-10]以测试桩复杂度为衡量标准，通过强化学习中的 Q-Learning 算法，使智能体在环境中逐渐探索出具有最优总体测试桩复杂度的类集成测试序列。此外，Zhang 等[11]首次提出使用深度强化学习的 Advantage Actor-Critic 算法生成类集成测试序列。Li 等[12]通过使用强化学习中的 Sarsa 算法并改进奖励函数等措施，能够以较低的测试桩代价生成类集成测试序列。

在上述研究中，现有基于强化学习的研究方法分别使用了 Q-Learning，Advantage Actor-Critic 和 Sarsa 等算法以解决类集成测试序列生成问题。在使用强化学习的方法来解决类集成测试序列生成问题时，环境状态是智能体学习如何优化其行为的重要依据。现有基于强化学习的研究方法[8-12]所考虑的环境状态仅和智能体所选动作相关，并未引入程序中实际存在的信息，故智能体在计算选择动作时所依据的信息较少，导致得到的类集成测试序列所需的测试成本仍较高。由于程序本身存在复杂的类间依赖关系等信息，这些信息可以构成图结构的数据，因此本文针对此问题考虑引入程序信息，通过使用图卷积神经网络(Graph Convolutional Network，GCN)来帮助智能体计算选择合适的动作。综上所述，本文提出了融合深度强化学习和图卷积神经网络的类集成测试序列生成方法，主要改进工作和特点可归纳为：

1)首次通过结合深度强化学习和图神经网络来解决类集成测试序列生成问题。该方法的目的是引入程序中实际存在的信息，并使用更为优化的强化学习方法[13]以帮助智能体计算选择合适的类。本文将程序中的类节点重要性和类间依赖关系等信息看成图结构的数据，并将其作为环境的初始状态，同时改进智能体的网络结构以及智能体的动作选择策略等方面，使环境状态和程序的类间关系产生关联，最终满足智能体和环境在不断地交互下生成符合要求的具有最小总体测试桩复杂度的类集成测试序列。

2)根据智能体选择的动作考虑更多的环境状态，并设计新的奖励函数。智能体根据环境反馈的奖励值判断当前状态所选动作的好坏，本文针对不同状态下智能体选择不同的动作设计了新的不同的奖励值，以帮助智能体更加准确地判断当前动作的利弊。

2 背景知识

2.1 类间依赖关系

在面向对象编程中，通过程序是否运行将类间依赖关系分为类间静态依赖关系与类间动态依赖关系[14]。本文涉及的主要是类间静态依赖关系，其主要包括继承(Inheritance，In)、聚集(Aggregation，Ag)、关联(Association，As)、组合(Composition，Cp)和使用(Use，Us)[14]。

2.2 类集成测试序列生成问题

类集成测试序列生成问题，即通过某种方法，在有限的时间内确定一组最优的测试顺序，并测试系统中的各个类，同时确保测试桩代价尽可能低。本文通过类集成测试序列状态空间存储每个类的状态和选择顺序，如式(1)所示：

$$S=[C_0,C_1,\cdots,C_{N-1}] \tag{1}$$

其中，S 表示整个类集成测试序列的状态；C_i 表示类 Id 为 i 的状态，$C_i\in\{0,1\}$；N 表示程序所拥有类的个数。当 $C_i=0$ 时，表示类 i 还未被选择；$C_i=1$ 时，表示类 i 已经被智能体选择过。S 的初始状态为$[0,0,\cdots,0]$，表示智能体还未选择任何一个类；智能体和环境不断地交互，直到所有的类都已经被选择时，类集成测试序列达到最终状态，表示为$[1,1,\cdots,1]$，此时智能体选择动作的顺序即为所生成的类集成测试序列。

2.3 类节点重要性

本文类节点重要性的计算借助 Wang 等[15]提出的利用复杂网络理论计算类节点重要性，即通过 HITS 算法[16]，将类的影响力(Impact of Class，IC)作为 HITS 中的权威值，将类的复杂性(Complexcity of Class，CC)作为 HITS 中的枢纽值。其中 IC 代表类的入度，CC 代表类的出度。每个类节点重要性的值主要由式(2)计算得到。

$$C_i=W_{ic}\cdot IC_i+W_{cc}\cdot CC_i \tag{2}$$

其中，C_i 表示类 i 的重要性，IC_i 和 CC_i 分别为类 i 的权威值和枢纽值，W_{ic} 和 W_{cc} 分别为 IC_i 和 CC_i 的权重，W_{ic} 和 W_{cc} 需要满足二者之和为 1。

2.4 测试桩代价度量方法

测试桩代价可以衡量类集成测试顺序的优劣，本文使用总体测试桩复杂度作为衡量 CITO 的标准。

定义 1(测试桩[4]) 用于模拟某个被调用类的方法或属性等其他组件，根据测试桩所模拟类中方法和属性是否完整可分为两种：通用测试桩和特效测试桩。通用测试桩可以模拟出整个类的方法和属性，能够实现模拟类的全部行为；特效测试桩仅实现了模拟类的部分行为，根据测试的需要来模拟某个类的特定行为。

定义 2(类间耦合) 用于表示类之间相互依赖程度的度量，具体包括属性耦合和方法耦合。

定义 3(属性耦合) 类间耦合的一种形式，主要发生在两个独立的类之间。具体来说，当一个类(源类)依赖于另一个类(目标类)的属性时，即源类调用目标类的成员变量，或者将目标类的实例或属性作为自己的方法参数或方法返回值时，称这两个类之间存在属性耦合。

定义 4(方法耦合) 指当一个类(源类)调用另一个类(目标类)中的方法时，这两个类之间所形成的关系。

测试桩复杂度的计算方法如式(3)－式(5)所示：

$$SCplx(x,y)=\sqrt{W_A\cdot\overline{A(x,y)}^2+W_M\cdot\overline{M(x,y)}^2} \tag{3}$$

$$\overline{A(x,y)}=\frac{A(x,y)}{A_{\max}-A_{\min}} \tag{4}$$

$$\overline{M(x,y)}=\frac{M(x,y)}{M_{\max}-M_{\min}} \tag{5}$$

其中，$SCplx(x,y)$指类 x 依赖于类 y 的测试桩复杂度。W_A 和 W_M 分别为属性耦合和方法耦合的权值，取值均在[0,1]之间，二者权重之和为 1。$A(x,y)$与 $M(x,y)$分别代表类 x 依赖于类 y 的属性和方法的个数，$\overline{A(x,y)}$与$\overline{M(x,y)}$分别代表对属性耦合和方法耦合归一化后的值。

$$OCplx(o)=\sum_{x,y\in N}SCplx(x,y) \tag{6}$$

其中，$OCplx$ 指以测试序列 o 进行测试的总体测试桩复杂度，N 表示需要构建的测试桩集合。

3 融合深度强化学习和图卷积神经网络的类集成测试序列生成方法

本文使用深度强化学习和图卷积神经网络[17]来探索 CITO 问题。深度强化学习主要由 2 个交互对象以及 3 项基本要素构成。其中，交互对象包括智能体和环境，基本要素包括环境状态、动作以及奖励。因此，下面将围绕 CITO 问题介绍本文设计的网络结构、环境状态及其变化、动作空间、奖励函数以及算法流程。

3.1 网络结构

在面向对象程序中，每个类承担着不同的职责，使得某些类可能比其他类更为重要，同时考虑程序中存在复杂的类间依赖关系，这些程序中的信息都可作为环境状态来帮助智能体计算选择动作。因此，本文将类节点重要性和类间依赖关系等程序信息作为模型的输入，设计网络结构(见图 1)。

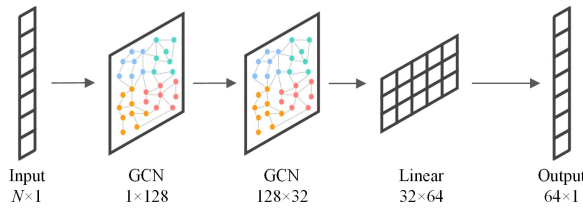


图 1 考虑类节点重要性的网络结构

Fig. 1 Network structure considering class node importance

图 1 是本方法考虑类节点重要性的网络结构，用于处理类集成测试序列生成问题(Network Structure Considering Class Node Importance to Deal with Class Integration Test Order Generation Problem，CITO_INS)。其中，N 表示测试程序中类的个数。本文的每个类节点重要性使用 1 维数据表示，则 N 个类将有 $N\times1$ 维的类节点重要性数据。本文针对 GCN 层数的设计参考了文献[17]，层数过多会造成信息冗余，导致不同节点的特征变得越来越相似，从而损失区分度。数据通过两层的 GCN 后经过一层全连接网络最终输出结果为 $N\times1$ 维，其维度和动作空间的大小 N 保持一致，便于智能体根据贪婪策略选出最大概率的类。

3.2 环境状态及其变化

3.2.1 环境初始状态

智能体的网络结构负责根据当前环境状态计算选择合适的类。为了将程序信息和智能体产生联系以帮助智能体选择动作，本文将程序中的类节点重要性和类间依赖关系构成图结构的数据来作为环境的初始状态，并将这种图数据称为类间关系图。在类间关系图中，类节点重要性作为图的节点特征，用于表示每个类的重要程度；类间依赖关系作为图的有向边，为智能体提供类间依赖关系信息。如图 2 所示，每个节点的数字代表相应类的 ID，其类节点重要性由式(2)计算得到，每个节点的数据维度为 1 维；节点间的有向边通过不同的类间依赖关系表示。此外，为了使智能体在训练前期可以尝试更多的类集成测试序列，本文并未对类间关系图中的有向边加以权重，而是让智能体不断地探索自主学习边的权重。

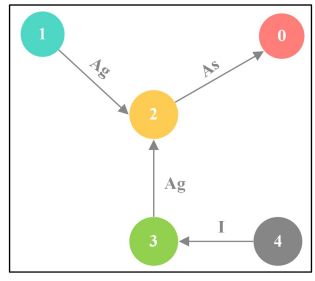


图 2 类间关系图

Fig. 2 Inter-class relationship diagram

3.2.2 环境状态变化

智能体和环境交互时，环境状态的变化使智能体能够及时地调整优化动作选择的策略。本文环境根据智能体选择的动作不断调整环境状态，以帮助智能体选择更为合理的动作。

环境状态的变化描述为：假设当前所选类为 i，环境反馈智能体的奖励值为 R_i，则节点 i 的节点特征变化如式(7)所示：

$$N_i = N_i + (R_i / c) \tag{7}$$

其中，N_i 表示节点 i 的节点特征，c 是奖励系数，是一个常数。$R_i>0$，表示 i 的重要性增大，下一轮 GCN 计算选择其他类时，i 依赖的其他类的值也将增大；$R_i<0$，表示 i 重要性减小，下一轮 GCN 计算选择其他类时，i 依赖其他类的值也将减小，同时避免 i 被重复选择。对于节点间的有向边，其变化描述为：在类间关系图中去除指向 i 的边。

3.3 动作空间

在 CITO 问题中，每个回合智能体负责选出类集成测试序列，智能体的动作就是选择某个类。在实际智能体选择动作时，会提前设定一个固定值，当智能体和环境交互次数在固定次数内时，允许智能体选择重复的类，但对于选择重复的类，环境将会返回给智能体一个设定的负奖励值，让智能体学会选择不重复的类。为防止智能体重复选择某个类的次数过多，当智能体和环境交互次数超过固定次数后，智能体必须选择未被选择过的类，直到本回合结束。

3.4 奖励函数

本文基于 Zhang 等[18]提出的用于评估所选类的奖励机制，并在此基础上进行改进，如式(8)－式(10)所示：

$$NR(x) = TR(x) - TC(x) \tag{8}$$

$$TR(x) = \sum_{y \in Y} SCplx(y,x)(y \neq x) \tag{9}$$

$$TC(x) = \sum_{y \in Y} SCplx(x,y)(x \neq y) \tag{10}$$

其中，$NR(x)$指选择类 x 所获得的净收益；集合 Y 指程序中还未被智能体选择过的类的集合；$TR(x)$指在集合 Y 中，集成测试类 y 时需要为 y 构建类 x 的测试桩代价之和；$TC(x)$指在集合 Y 中，集成测试类 x 时需要为 x 构建类 y 的测试桩代价之和。

当本回合结束时，环境需要反馈一个回合结束时的奖励。假设环境目前已存在一个最佳的类集成测试序列 o，总体测试桩复杂度为$OCplx(o)$；当一个新的回合结束时，所获得的类集成测试序列为 o_i，总体测试桩复杂度为 $OCplx(o_i)$。$OCplx(o_i)$可以分为两种情况：$OCplx(o_i)>OCplx(o)$时，奖励值为 $c\times(OCplx(o)-OCplx(o_i))$，即通过奖励值表示当前序列和最佳序列之间总体测试桩复杂度的差距；$OCplx(o_i)\leqslant OCplx(o)$时，奖励值为 MAX，表示智能体找到了一个新的最佳类集成测试序列。

综上所述，CITO 问题中奖励 R 的表达式可以总结为：

$$R=\begin{cases} c\times NR(a_i), & done=false \\ c\times(OCplx(o)-OCplx(o_i)), & \\ \qquad done=true, OCplx(o_i)>OCplx(o) & \\ MAX, \quad done=true, OCplx(o_i)\leqslant OCplx(o) & \\ MIN, \quad done=false, a_i \in o_{i-1} & \end{cases} \tag{11}$$

其中，c 是奖励系数，是一个常数，用于防止奖励值过小；a_i 为 i 时刻智能体选择的类；o 是环境中已有的最佳类集成测试序列；o_i 是 i 时刻智能体获得的类集成测试序列；$done$ 是一个标志，用来表示当前回合是否结束，$done=false$ 表示本回合还未结束，$done=true$ 表示本回合结束。MAX 是实验系统设定的最大奖励值，是一个固定整数，本文的实验系统即借助深度强化学习搭建的类集成测试序列生成系统；MIN 是实验系统设定的固定惩罚负值，当智能体选择重复类时，环境会返回 MIN。

3.5 算法流程

本文借助深度强化学习中的双重深度 Q-网络(Double Deep Q-Network，DDQN)算法[19]和优先经验回放机制[20]完成智能体和环境的搭建，算法流程如算法 1 所示。

算法 1 融合深度强化学习和图卷积神经网络的类集成测试序列生成算法

输入：训练次数 T，重播期 K，步骤计数 q，回合结束标志 done，批量下降的样本数目 m，探索率 ε，折扣因子 γ，当前网络 Q，目标网络 Q′，目标网络参数更新频率 C，指数 α，指数 β，β 增长率 v，sum-tree 叶子节点总数 M

输出：最佳的类集成测试序列 best_order 和最小总体测试桩复杂度 $cost_{min}$

1. 随机初始化当前网络 Q 的参数 w，初始化 Q′的参数 w′=w，初始化 sum-tree 中所有叶子节点的优先级 p(i)=1
2. for episode=1，T do
3. 初始化状态 s_0，done=false，初始化环境 env
4. while not done do
5. 将 ϕ(s)输入当前网络 Q 中，得到 Q 的所有动作对应的 Q 值输出，使用 ε-贪婪策略选择动作 a
6. 在状态 s 下执行当前动作 a，得到新的状态 s′，奖励 r，done
7. 将样本{ϕ(s)，a，r，ϕ(s′)，done}存入 sum-tree
8. s′=s
9. if q>K then
10. 更新 β，β=min(1.0，β+v)
11. 从 sum-tree 中采样 m 个样本{ϕ(s_i)，a_i，r_i，ϕ(s_i')，done}，i=0，1，2，…，m−1
12. 每个样本的采样概率基于 $P(i)=\frac{p_i^{\alpha}}{\sum_t p_t^{\alpha}}$
13. 计算每个样本的重要性采样权重
14. 计算当前目标 Q 值 y_i：

$$y_i=\begin{cases} r_i, & done=true \\ r_i+\gamma Q'(\phi(s_i'), \arg\max_{a'} Q(\phi(s_i'),a',w),w'), & done=false \end{cases}$$

15. 计算损失
16. 反向传播，更新当前网络 Q 的参数
17. 计算样本误差，更新 sum-tree 中节点优先级
18. end if
19. if q%C==0 then
20. 更新目标网络 Q′参数，w′=w
21. end if
22. if done then
23. 记录当前总体测试桩复杂度 cost，类集成测试序列 order
24. if cost<$cost_{min}$ then
25. $cost_{min}$=cost
26. best_order=order
27. end if
29. break
30. end if
31. end while
32. end for

4 实验

4.1 实验对象

实验对象的详细信息如表 1 所列，表 1 中的第 2—5 列分别表示选取程序中类的数量、依赖关系的个数、环路个数和代码总行数。本文和文献[3,9-10]中的实验对象保持一致，其中 SPM，ATM，ANT，BCEL 和 DNS 来自文献[4]elevator，daisy，email_spl 和 notepad_spl 是 SIR[1)] 上的开源程序。

表 1 实验对象信息

Table 1 Experimental subject information

程序	类数量	依赖关系	环路个数	代码行数
elevator	12	27	23	934
SPM	19	72	1178	1198
ATM	21	67	30	1390
daisy	23	36	4	1148
ANT	25	83	654	4093
email_spl	39	63	38	2276
BCEL	45	294	416091	3033
DNS	61	276	16	6710
notepad_spl	65	142	227	2419

4.2 参数设计

实验前需要为本算法设置相关参数，本文的参数设置如表 2 所列。

表 2 模型训练参数

Table 2 Model training parameters

学习参数	值	学习参数	值
训练次数 T	200000	重播期 K	200
学习率 lr	0.005	折扣因子 γ	0.9
探索率 ε	动态 ε	批量下降的样本数目 m	256
指数 α	0.6	目标网络参数更新频率 C	200
指数 β	0.4	sum-tree 叶子节点总数 M	100000
β 增长率 v	0.001	选择重复类的惩罚值 MIN	−500
奖励系数 c	100	更好结果的奖励值 MAX	10000

其中，T 表示智能体需要完成的训练轮数；lr 是优化器的参数，用于随机梯度下降；ε 是智能体 ε-贪婪策略的参数，即以 ε 的概率随机选择动作，以 $1-\varepsilon$ 的概率由智能体计算选择动作；α，β 和 v 是优先经验回放机制的相关参数；重播期 K 是每隔多少交互次数时，智能体开始从 sum-tree 中抽取样本进行当前网络 Q 的参数更新，所抽取的样本个数为 m；γ 表示当前状态受未来奖励的影响比例；C 指智能体在训练多少次时将当前网络 Q 的参数复制给目标网络 Q'；M 是 sum-tree 叶子节点的总个数，表示最多可以保存样本数据的总数量；MIN，MAX 和 c 是奖励函数中式(11)的相关参数。

4.3 实验结果及分析

4.3.1 总体测试桩复杂度对比

本小节实验主要对比所提方法和其他方法的效果。本文对表 1 中的程序展开实验，实验结果以类集成测试序列的总体测试桩复杂度为指标进行比较。CITO_INS 中的智能体采用 ε-贪婪策略选择动作，因此每次的实验结果都有一定的随机性。为了解决此问题，本文对每次实验均重复运行 10 次，将 10 次的结果取平均值作为实验的最终结果并和其他方法相比较。

通过实验得到了 CITO_INS 的实验结果。为了更好地展示本方法在 CITO 问题上的效果，本文将实验结果和其他方法相比较，包括基于图论的方法、基于搜索的方法以及基于强化学习的方法。其中，基于图论的方法包括 Le Traon 等[21]的方法、Tai 和 Daniels[22]的方法，以及 Briand 等[23]的方法。基于搜索的方法包括 Briand 等[4]提出的遗传算法 GA、Wang 等[24]提出的随机交互算法 RIA 以及 Zhang 等[5]提出的粒子群算法 PSO。基于强化学习的方法包括 Ding 等[9]提出的强化学习算法 CITO_RL。其他方法的实验数据来自文献[9]。

各方法在不同程序上的总体测试桩复杂度和特效测试桩个数如表 3 所列。表中每一行展示了不同的程序通过使用不同的方法得到的总体测试桩复杂度，每一行的最小值均用粗体表示。另外，基于图论的方法中并未对 elevator，daisy，email_spl 以及 notepad_spl 程序进行实验，因此表 3 中没有列出对应数据。

表 3 总体测试桩复杂度对比

Table 3 Comparison of overall stubbing complexity

程序	图论			搜索			强化学习	
	Le Traon	Tai	Briand	GA	RIA	PSO	CITO_RL	CITO_INS
elevator	—	—	—	2.03	2.04	2.87	1.72	**1.70**
SPM	8.4	8.08	5.82	3.5	3.48	3.02	5.26	**2.40**
ATM	3.37	2.99	2.7	3.09	2.43	2.59	2.81	**2.05**
daisy	—	—	—	0.58	0.32	0.92	**0.22**	0.33
ANT	3.72	3.87	3.31	2.13	2.23	2.32	**2.08**	3.58
email_spl	—	—	—	0.74	0.66	1.02	0.59	**0.55**
BCEL	8.23	8.68	**5.81**	9.7	8.71	8.61	8.58	19.23
DNS	5.02	4.63	**1.51**	5.51	4.33	5.81	3.7	5.05
notepad_spl	—	—	—	**1.96**	1.92	4.68	4.67	3.09

通过观察表 3 中 CITO_INS 一列，可以发现在 9 个程序中，有 4 个程序的实验结果能够达到最优，分别是 elevator，SPM，ATM 和 email_spl 程序。对于 daisy，ANT，BCEL，DNS 和 notepad_spl 程序，实验结果按从小到大排序，该方法分别位于第 3，6，8，6 和 3 名。

对于 daisy 程序，CITO_INS 和 CITO_RL 相比仅差 0.11，和第 2 名的 RIA 相比仅差 0.01，说明 CITO_INS 在 daisy 程序上是有效的。但在 daisy 程序中，ID 为 1 和 2 的类依赖其

1) http://sir.unl.edu

他类的有向边过多，CITO_INS 的智能体在和环境交互过程中无法较好地处理类 1 和类 2 被选择的顺序，从而使得 CITO_INS 所获得的总体测试桩复杂度较高。

对于 ANT 程序，CITO_INS 位于第 6 名。通过实验发现，ANT 程序上的环路个数较多，导致智能体在选择测试桩复杂度较高的动作时，环境反馈的奖励值较大。智能体根据环境反馈的奖励，在后续选择动作时将优先选择测试桩复杂度较高的动作，从而导致 CITO_INS 生成的类集成测试序列的总体测试桩复杂度较高。

对于 BCEL 程序，CITO_INS 位于第 8 名。观察实验结果并结合表 1 可以发现，BECL 程序中的类间依赖关系和环路个数过多，导致智能体通过网络计算选择类时无法完整地考虑所有的类间依赖关系，从而无法正确地选择合适的类；同时智能体在训练过程中每次抽取经验的数量和 BECL 程序中类的数量相比，二者数量的比值较小，导致智能体无法在训练过程中完整地学习所有环境状态下的动作，使得 CITO_INS 生成的类集成测试序列的总体测试桩复杂度过高。

对于 DNS 程序，CITO_INS 位于第 6 名。观察实验结果并结合表 1 可以发现，DNS 程序中的类的数量和类间依赖关系过多，当智能体选择测试桩复杂度较高的动作时，环境反馈的奖励值较大，智能体在训练过程中会优先选择此动作，导致 CITO_INS 生成的类集成测试序列的顺序较为固定，智能体无法尝试其他的测试序列，从而导致智能体无法找出总体测试桩复杂度更低的类集成测试序列。

对于 notepad_spl 程序，CITO_INS 位于第 3 名，和 CITO_RL 相比高出 1.58，说明在基于强化学习的方法中 CITO_INS 能更好地处理 notepad_spl 程序。观察实验结果并结合表 1 可以发现，notepad_spl 中的依赖关系和环路个数较多，且 notepad_spl 程序中每个类的类节点重要性数值差距较小，智能体在计算选择类时无法准确判断在某个环境状态下哪个类才是最佳的类，从而导致智能体生成的类集成测试序列并不能保证是最佳的类集成测试序列。

CITO_INS 的总体测试桩复杂度箱型图如图 3 所示。从图 3 中可以发现，CITO_INS 在 elevator，SPM，ATM，daisy，email_spl 和 notepad_spl 程序上的 10 次结果中比较稳定，ANT，DNS 和 BCEL 程序次之。

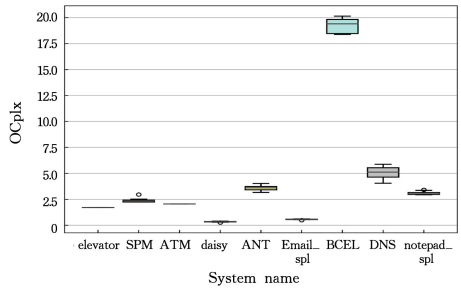


图 3 CITO_INS 的总体测试桩复杂度箱型图

Fig. 3 Box diagram of overall stubbing complexity for CITO_INS

图 4 展示了各方法在不同程序上的实验结果对比。其中，Briand 方法在 2 个程序中的实验结果最优，RIA 在 notepad_spl 程序中的实验结果最优，CITO_RL 在 daisy 和 ANT 程序中的实验结果最优，而 CITO_INS 在 9 个程序中有 4 个程序的实验结果能够达到最优。

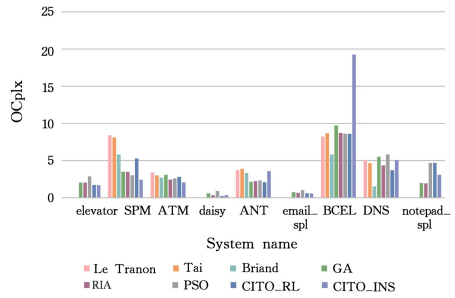


图 4 总体测试桩复杂度对比

Fig. 4 Comparison of overall stubbing complexity

通过对 CITO_INS 的实验结果进行分析可知，以总体测试桩复杂度作为主要指标时，CITO_INS 在一些类间依赖关系和环路个数较少的程序中的总体测试桩复杂度能够达到最优，但在类间依赖关系和环路个数较多的程序中却效果不佳。考虑 CITO_INS 的网络结构使用了图卷积神经网络，可能是网络结构中使用图卷积神经网络的层数过少、维度较小等原因导致智能体在面对类间依赖关系和环路个数较多的程序时无法全局处理程序的类间依赖关系等信息，从而计算错误选择了不合理的类，使得 CITO_INS 生成的类集成测试序列的总体测试桩复杂度较高。另外一个原因可以归结为当处理类数量、类间依赖关系和环路个数较多的程序时，奖励函数的设计并不能合理地反馈智能体，有时还会出现当前动作的测试桩复杂度过高，但奖励值依然是一个较大正数的情况，使得智能体无法生成合理的类集成测试序列。

4.3.2 训练时间对比

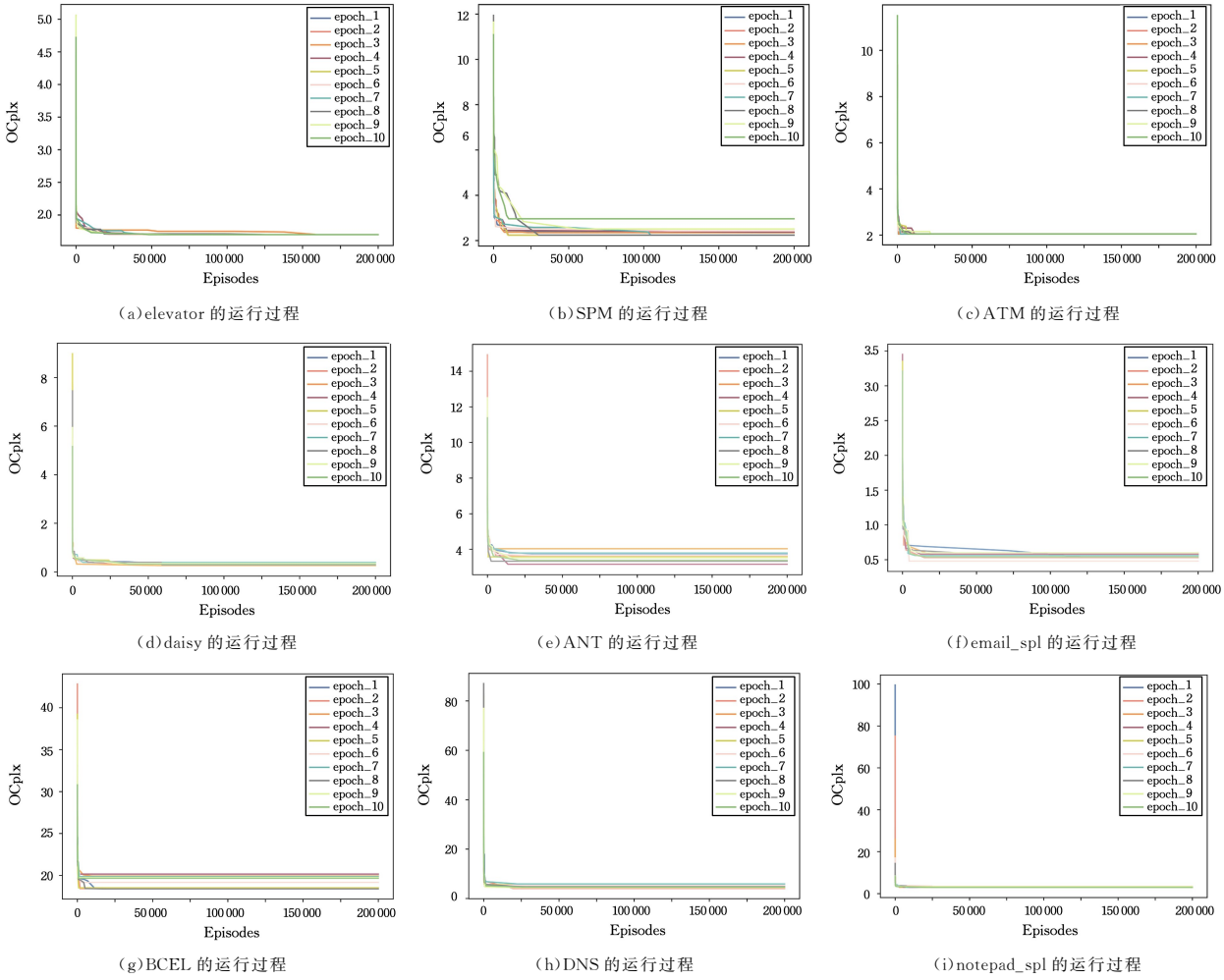
由于本文方法涉及深度学习，同时为了保证智能体可以完整地学习每个程序中的信息，因此本文所设计每轮的训练次数为 20 万次，当每一轮训练结束时，智能体会进行上万次的网络更新以保证实验效果，故本文方法所耗费时间在 10^8 ms 以上。其他方法通过固定的算法进行迭代得到实验结果，每轮训练时长保持在 $10^3 \sim 10^5$ ms 区间。对训练时间进行分析可以发现，本文方法在时间上并没有优势，但本文方法可以在大多数程序中获得总体测试桩复杂度较低的类集成测试序列。

4.3.3 收敛速度

本文方法尽管在训练时间上和其他方法相比不占优势，但我们可以从训练次数上对总体测试桩复杂度进行研究，通过减少训练次数来减少本文方法的训练时间。在某一次程序运行过程中，随着训练次数的增加，本文方法得到的总体测试桩复杂度也会逐渐平稳达到收敛。本文研究程序每次运行时总体测试桩复杂度随训练次数增加的变化，通过找到合适的训练次数，使得本文方法在较少的训练次数内即可获得总体测试桩复杂度较低的类集成测试序列。

图 5 展示了程序运行过程中总体测试桩复杂度随训练次数的变化。对图 5 进行分析可以发现，CITO_INS 在大多数程序上进行训练时最小总体测试桩复杂度达到收敛时的训练次数基本都保持在 2.5 万次或 5 万次内，仅有 2 次运行时的训练次数需要超过 10 万次时最小总体测试

桩复杂度才能达到收敛。这说明对大多数的程序进行训练时，可以将每次运行的训练次数缩短至 5 万次，这样既保证了实验效果，又可以缩短训练时长并降低计算资源的成本。因此，通过研究总体测试桩复杂度在运行过程中的收敛速度，可以在本文方法的基础上找到一个固定且数值较小的训练次数，在此训练次数内既能够保证训练所生成的类集成测试序列的总体测试桩复杂度较低，又可以降低计算资源和时间的成本。



(a) elevator 的运行过程　(b) SPM 的运行过程　(c) ATM 的运行过程

(d) daisy 的运行过程　(e) ANT 的运行过程　(f) email_spl 的运行过程

(g) BCEL 的运行过程　(h) DNS 的运行过程　(i) notepad_spl 的运行过程

图 5　程序的运行过程

Fig. 5　Running process of program

结束语　本文利用深度强化学习和图卷积神经网络等技术提出了融合深度强化学习和图卷积神经网络的类集成测试序列生成方法。本文基于深度强化学习，使用了 DDQN 算法搭建类集成测试序列生成场景，通过图卷积神经网络作为智能体的网络完成计算并选择动作，同时环境返回的状态由图结构的数据组成。完成算法设计后对选取的 9 个实验对象进行实验并将得到的实验结果和其他方法进行对比，发现在 9 个程序中有 4 个程序的实验结果可以达到最优；同时，对剩余程序的实验结果进行了对比分析。实验结果表明，与现有方法相比，在以总体测试桩复杂度为评价指标时，本文方法的结果更优。

本文方法尽管能够在一定程度上降低生成类集成测试序列所需的测试桩代价，但是其时间开销过大，且该方法在处理类间依赖关系和环路个数较多的程序时，无法获得总体测试桩代价较低的类集成测试序列。因此，针对本文方法，可以从时间开销和算法自身等方面进行改进，如通过探索改进算法模型和网络目标值的更新策略等，使算法在较短的时间内即可获得总体测试桩复杂度尽可能低的类集成测试序列，同时面对类间依赖关系和环路个数较多的程序时，也能够在规定的时间内获得总体测试桩复杂度较低的类集成测试序列。

参 考 文 献

[1] BERNARD H. Fundamentals of Software Testing[M]. John Wiley and Sons Inc, 2024.

[2] KUNG D, GAO J, PEI H, et al. A Test Strategy for Object-oriented Programs[C]// Proceedings of Computer Software and Applications Conference. 1995: 239-244.

[3] JIANG S J, ZHANG M, ZHANG Y M, et al. An Integration Test Order Strategy to Consider Control Coupling[J]. IEEE Transactions on Software Engineering, 2019, 47(7): 1350-1367.

[4] BRIAND L C, FENG J, LABICHE Y. Using Genetic Algorithms and Coupling Measures to Devise Optimal Integration Test Orders[C]// Proceedings of the International Conference on Software Engineering and Knowledge Engineering. 2002: 43-50.

[5] ZHANG Y M, JIANG S J, CHEN R Y, et al. Class Integration

Testing Order Determination Method Based on Particle Swarm Optimization Algorithm[J]. Chinese Journal of Computers, 2018,41(4):931-945.

[6] CABRAL R D V,POZO A,VERGILIO S R. A Pareto Ant Colony Algorithm Applied to the Class Integration and Test Order Problem[C]// Proceedings of the IFIP WG 6.1 International Conference on Testing Software and Systems. 2010:16-29.

[7] GAO Y Z,NIE Y M. Survey of Multi-agent Deep Reinforcement Learning Based on Value Function Factorization[J]. Computer Science,2024,51(S1):34-42.

[8] CZIBULA G,CZIBULA I G,MARIAN Z. An Effective Approach for Determining the Class Integration Test Order Using Reinforcement Learning[J]. Applied Soft Computing, 2018, 65(6):517-530.

[9] DING Y R,ZHANG Y M,JIANG S J,et al. Research on Generation Method of Class Integration Test Order Based on Reinforcement Learning[J]. Journal of Software,2022,33(5):1674-1698.

[10] DING Y R,ZHANG Y M,YUAN G,et al. Integration Test Order Generation Based on Reinforcement Learning Considering Class Importance[J]. The Journal of Systems and Software, 2023,205:111823.

[11] ZHANG Y H,ZHANG Y M,ZHANG Z C,et al. Generation Method of Class Integration Test Order Based on Deep Reinforcement Learning[J]. Acta Electronica Sinica, 2023, 51(2): 455-466.

[12] LI Y,ZHANG Y M,DING Y R,et al. A Class Integration Test Order Generation Approach Based on Sarsa Algorithm[J]. Automated Software Engineering. 2024,31(7):1-33.

[13] WANG Y,CHEN Z B,WU Z R,et al. Review of Reinforcement Learning for Combinatorial Optimization Problem[J]. Journal of Frontiers of Computer Science and Technology, 2022, 16(2): 261-279.

[14] ZHANG Y M,JIANG S J,ZHANG M,et al. Survey of Class Test Order Generation Techniques for Integration Test[J]. Chinese Journal of Computers,2018,41(3):670-694.

[15] WANG Y, YU H, ZHU Z L. A Class Integration Test Order Method Based on the Node Importance of Software[J]. Journal of Computer Research and Development,2016,53(3):517-530.

[16] KLEINBERG J. Authoritative Sources in a Hyperlinked Environment[J]. ACM,1999,46(5):604-632.

[17] KIPF T N,WELLING M. Semi-Supervised Classification with Graph Convolutional Networks[C]// Proceedings of the 5th International Conference on Learning Representations. 2017:1-14.

[18] ZHANG M,JIANG S,ZHANG Y,et al. A Multi-level Feedback Approach for the Class Integration and Test Order Problem[J]. The Journal of Systems and Software,2017,133(1):54-67.

[19] HASSELT H V,GUEZ A,SILVER D. Deep Reinforcement Learning with Double Q-learning[C]// Proceedings of the 30th AAAI Conference on Artificial Intelligence. 2016:2094-2100.

[20] CHAUL T,QUAN J,ANTONOGLOU I,et al. Prioritized Experience Replay[C]// Proceedings of the 4th International Conference on Learning Representations. 2016:1-21.

[21] LE TRAON Y,JERON T,JEZEQUEL J M,et al. Efficient Object-oriented Integration and Regression Testing[J]. IEEE Transactions on Reliability,2000,49(1):12-25.

[22] TAI K C,DANIELS F J. Test Order for Inter-class Integration Testing of Object-Oriented Software[C]// Proceedings of the 21st Annual International Computer Software and Applications Conference. 1997:602-607.

[23] BRIAND L C,LABICHE Y,WANG Y. An Investigation of Graph-based Class Integration Test Order Strategies[J]. IEEE Transactions on Software Engineering,2003,29(7):594-607.

[24] WANG Z S,LI B X,WANG L L,et al. Using Coupling Measure Technique and Random Iterative Algorithm for Inter-Class Integration Test Order Problem[C]// Proceedings of the 34th Annual IEEE Computer Software and Applications Conference Workshops. 2010:329-334.

WANG Chenyuan, born in 2001. His main research interests include intelligent software engineering and so on.

ZHANG Yanmei, born in 1982, Ph.D, associate professor, is a member of CCF (No. 27031M). Her main research interests include intelligent software engineering and machine learning.

(责任编辑:何杨)