DOI:10.19651/j.cnki.emt.2210562

基于改进混合蛙跳算法的测试数据自动生成*

刘会颖 刘紫阳 颜明会
(北华航天工业学院 廊坊 065000)

摘 要: 测试数据的生成是实现软件测试自动化的重要环节。为了提高单元测试中测试数据的生成质量和效率,提出一种基于混合蛙跳算法的测试数据生成算法。该算法通过引入动态阈值来控制个体的移动步长,以平衡算法的全局开发和局部搜索能力,同时改进个体的随机跳动策略,转化为向随机个体学习,增强种群之间的信息交流以提高算法的全局搜索能力。将改进的算法应用到测试数据生成中。实验结果表明,在种群规模不断变化的情况下,改进的混合蛙跳算法相较于标准混合蛙跳算法、布谷鸟搜索算法、粒子群优化算法,其稳定性最强;在测试数据生成的平均迭代次数评价指标上改进的混合蛙跳算法优于对比算法。

关键词: 混合蛙跳算法;测试数据生成;动态阈值;随机跳动策略
中图分类号: TP301.6 **文献标识码:** A **国家标准学科分类代码:** 510.1040

Automatic generation of test data based on improved shuffled frog leaping algorithm

Liu Huiying Liu Ziyang Yan Minghui
(North China Institute of Aerospace Engineering, Langfang 065000, China)

Abstract: The generation of test data is an important part of achieving software test automation. In order to improve the quality and efficiency of test data generation in unit testing, a test data generation algorithm based on shuffled frog leaping algorithm is proposed. The algorithm introduces a dynamic threshold to control the moving step size of individuals, so as to balance the global exploration and local exploitation abilities. At the same time, the worst individual random jump strategy in the standard algorithm is transformed into learning from random individual to enhance the information exchange between populations to improve the algorithm's global search capability. Apply the improved algorithm to test data generation. The experimental results show that the improved shuffled frog leaping algorithm is more stable than the standard shuffled frog leaping algorithm, cuckoo search algorithm and particle swarm optimization algorithm under the condition of changing population size. The improved shuffled frog leaping algorithm is better than the comparison algorithm in the evaluation index of the average number of iterations generated by the test data.

Keywords: shuffled frog leaping algorithm;test data generation;dynamic threshold;random jump strategy

0 引 言

软件测试是软件生命周期中非常重要且复杂的一个阶段,对于评估软件质量和保障软件可靠性具有极其重要的意义[1]。随着软件规模和软件数量的爆发式增长,传统的手工测试难以满足测试的需求,而在这个过程中,自动化测试得到了快速发展[2]。自动化测试可划分为单元测试自动化、功能测试自动化、性能测试自动化,本文重点研究单元测试自动化中的测试数据生成问题。

测试数据生成是软件测试中的一项重要活动,软件测试的准确性及效率主要依赖于测试数据的质量和数量,但现有的自动化测试工具大多仅提供用例的执行驱动,对于测试数据的生成贡献较小。

近年来,应用智能优化算法解决测试数据生成问题受到了越来越多的关注[3]。其核心思想是利用智能优化算法将测试数据的生成问题转化为函数寻优问题,通过计算适应度值来评判测试数据的优劣[4-5]。常用的算法有遗传算法、粒子群优化算法、蚁群算法等。为解决测试数据生成速

收稿日期:2022-07-04
* 基金项目:国家自然科学基金(51875018)、北华航天工业学院青年基金(KY-2021-05)项目资助

度低等问题,Esnashari 等[6]提出一种基于强化学习的遗传算法,虽然达到了提高测试数据生成速度的效果,但该方法未能在本质上对测试数据的正确性以及速度实现平衡。Han 等[7]提出采用粒子群算法并应用不同的适应度函数来评估种群中个体最优位置和全局最优位置将路径覆盖作为测试充分性判断依据,但适应度函数的设计对于算法不能起到很好的引导作用。袁光辉等[8]针对传统遗传算法在测试用例生成中的不足,提出基于混合遗传算法的生成方法,通过调节因子对自适应的交叉和变异算子进行改进,提高算法的局部搜索能力。张馨倬[9]提出了改进遗传蚁群算法,利用遗传算法初始化改进蚁群算法的信息素来提高目标路径的覆盖率。

以上研究表明,在测试数据自动生成中应用智能优化算法虽然可以提高测试覆盖率,但是对于数据生成问题的整体优化效果仍有待提高,且现有的研究多是围绕遗传算法,粒子群算法等比较常见的算法,这些算法的改进方法已是屡见不鲜,它们多集中在改进算法本身,不太关联要解决的问题,适应度函数的构造方法仍然呈现单一化[10]。因此本文提出一种基于混合蛙跳算法的改进算法,结合要解决的实际问题,修改适应度函数的构造方法,提高测试数据的生成质量和效率。

混合蛙跳算法根据青蛙个体在石块上觅食时的种群分布变化提出,结合了基于模因进化的模因演化算法和基于群体行为的粒子群算法,是一种全新的启发式群体进化算法。由于其概念简单,调整的参数少,计算速度快,易于实现等优势而受到广大学者的欢迎,同时在处理某些工程领域的问题时取得了显著的成果[11-14]。考虑到该算法独特的优点以及在众多工程应用上的优势,将其应用到测试数据生成中。

从解决测试数据生成问题的角度出发,本文提出一种改进的混合蛙跳算法(improved shuffled frog leaping algorithm,ISFLA),通过在算法中引入动态阈值来控制最差个体的跳动步长,协调算法的全局开发和局部搜索能力;此外,利用一种新的随机搜索策略控制青蛙的随机跳动行为,提高全局搜索能力。最后通过实验证明了改进的混合蛙跳算法在解决测试数据自动生成问题上的有效性及高效性。

1 理论背景

混合蛙跳算法由 Eusuff 和 Lansey 提出,其基本思想来源于青蛙群体的觅食。在混合蛙跳算法中,每个青蛙的位置代表一个可行解,青蛙所在的湿地上有大量的石块,每一代青蛙个体都会被分配到这些石块上,且只有石块上位置最差的青蛙才会跳动[15]。算法主要由以下步骤组成。

1)种群初始化

在湿地中随机生成包含 N 个青蛙的种群,计算种群中每个青蛙个体对应的适应度值 f_i。将 d 维空间的第 i 个青蛙表示为 $X_i=\{x_{i1},x_{i2},\cdots,x_{id}\}$,$i=1,2,3\cdots N$。每个个体仅有两个属性,位置 X_i 和当前位置对应的适应度值 f_i. 其中适应度值用来表示个体所处位置的优劣程度。

2)划分族群

根据实际问题,所有青蛙个体按照位置优劣进行排序,保留当前种群中的最优个体记为 X_g. 将排序后的种群划分成 m 个族群,每个族群中包含 n 只青蛙,满足 $N=m\times n$。

3)局部搜索

步骤(1)向局部最优个体学习。将第 i 个族群中的最优个体和最差个体表示为 X_b 和 X_w。设置 D 为青蛙的跳动步长。族群中最差个体的更新步骤如式(1)和(2)。

$$D=rand\times(X_b-X_w) \tag{1}$$

$$X'_w=X_w+D \tag{2}$$

式(1)中 $rand$ 表示[0,1]之间的随机数。如果个体的新位置 X'_w 的适应度值优于 X_w 对应的适应度值,则 X_w 被 X'_w 所取代,否则族群中的最差个体将向全局最优个体 X_g 学习,进行步骤(2)。

步骤(2)向全局最优个体学习。族群中的最差个体向局部最优个体学习失败后,使用种群中的最优个体 X_g 代替当前族群中的最优个体 X_b,重新对族群中的最差个体进行位置更新。个体位置更新公式如式(3)和(4)。

$$D=rand\times(X_g-X_w) \tag{3}$$

$$X'_w=X_w+D \tag{4}$$

式(3)中 $rand$ 表示 [0,1] 之间的随机数。如果 X'_w 的适应度值优于 X_w 的适应度值,则 X_w 被 X'_w 取代,否则进行步骤(3)。

步骤(3)最差个体随机更新。当向族群中的局部最优个体和种群中的全局最优个体学习失败时,在解空间中随机生成一个新个体来代替当前族群中的最差个体。随机个体位置生成如式(5)。

$$X'_w=rand(a,b) \tag{5}$$

式(5)中的 $rand$ 对应一个 d 维的随机向量,a 为向量随机生成的上界,b 为向量随机生成的下界。

4)全局混合

对 m 个族群重复进行局部搜索 $L_{\max}$ 次后,将每个族群中的所有个体进行混合并重新排序,再次进行分组和最差个体位置更新操作,直到满足全局搜索迭代次数 $G_{\max}$ 时算法结束。

以上算法流程如图 1 所示。

2 ISFLA

2.1 个体学习策略改进

标准混合蛙跳算法中个体的学习过程主要是族群中的最差个体以局部最优个体或全局最优个体作为学习对象进行有方向性的跳动。当算法迭代到一定程度时,族群中的局部最优解和全局最优解很难得到更新,使得算法更容易陷入局部最优。为了降低算法陷入局部最优的几率,防止

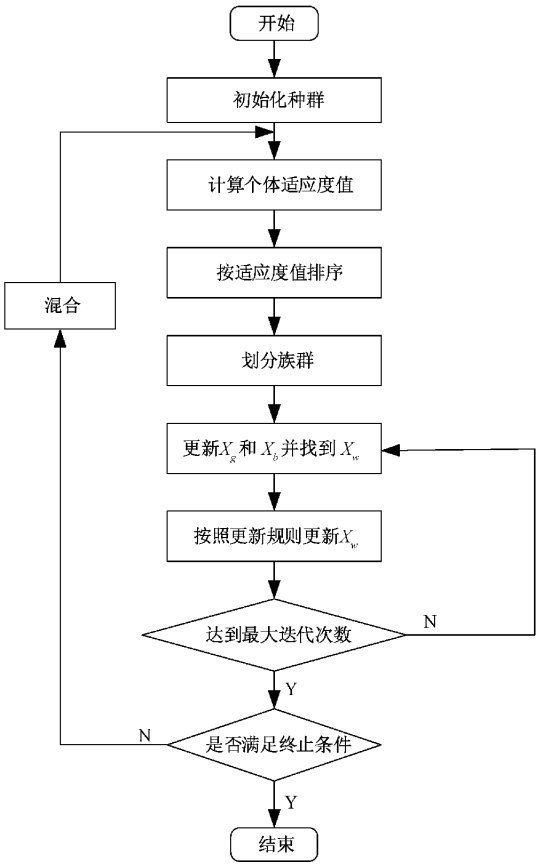

图 1　混合蛙跳算法流程

算法早熟，本文提出动态阈值 A 来调节个体的跳动步长，使最差个体有更大的机会跳出原有的步长范围。与此同时，引入一个用来关联动态阈值与族群局部搜索迭代次数的权重系数 a。动态阈值 A 和权重系数 a 的计算如式(6)和(7)所示。

$$a = 2 \times \sin(\pi/2 \times ((j/L_{\max}) + 1)) \tag{6}$$

$$A = 2 \times a \times rand - a \tag{7}$$

式中：j 为当前族群局部搜索的迭代次数，$L_{\max}$ 表示族群进行局部搜索的最大迭代次数，$rand$ 为 [0,1] 之间的随机数。$a \in [2,0]$ 非线性递减，A 为 $[-a,a]$ 之间的随机数。

在标准混合蛙跳算法中，根据第 2 节中的式(1)和(3)可知，族群中最差个体的跳跃步长始终在蛙体当前位置和全局最优或局部最优位置之间，使得算法的搜索空间受到了一定的约束，因此本文引入动态阈值 A 来调节跳跃步长。当 $|A|>1$ 时，青蛙将扩大跳跃范围，使其搜索空间更广泛，有更大的可能去寻找潜在的优秀解；当 $|A| \leqslant 1$ 时，青蛙将缩小跳跃范围，完成局部精细搜索。在迭代初期，A 的值较大，保证了算法的全局搜索能力，随着迭代的增加，A 的值不断的减小，提高了算法的局部搜索能力。对于动态阈值 A 的添加在很大的程度上平衡了算法的全局开发与局部搜索能力。步长的更新公式如式(8)和(9)。

$$D_1 = rand \times (A \times X_b - X_w) \tag{8}$$

$$D_2 = rand \times (A \times X_g - X_w) \tag{9}$$

式中：D_1 表示改进算法的最差个体向局部最优个体学习的步长，D_2 表示最差个体向种群中的最优个体学习的步长，$rand$ 为 [0,1] 之间的随机数。

2.2　个体随机搜索策略改进

在标准混合蛙跳算法中，当族群中的最差个体向族群中的最优个体和种群中的最优个体学习失败时，最差个体将进行随机跳动，同时利用随机跳动得到的个体来替代当前族群中最差个体。这一策略虽然在一定程度上保证了算法的随机性但不利于算法的收敛，因此，本文提出一种新的随机更新策略。

从现有的种群中随机选择一个个体，将这个随机个体作为当前族群中最差个体的学习目标，更新最差个体。这样不仅增强了种群个体之间的信息交流还提高了全局搜索能力。更新公式如式(10)和(11)。

$$D = rand \times (X_{rand} - X_w) \tag{10}$$

$$X'_w = D + X_w \tag{11}$$

式中：X_{rand} 为种群中一个随机个体，$rand$ 表示[0,1]之间的随机数。

3　测试数据生成模型

3.1　适应度函数

利用启发式搜索算法生成测试数据的过程中，适应度函数起着十分重要的作用。一般情况下，适应度函数反应数据的优劣。本文将适应度函数的返回值作为评判一条测试数据是否覆盖了目标路径的标准。

测试数据生成问题中，适应度函数的设计通常采用分支距离法[9]来实现。分支距离对测试数据接近目标覆盖分支的程度做出评价，通过分支谓词实现值计算。程序中分支谓词的一般表示形式为：$E_1\ op\ E_2$，其中 E_1 和 E_2 为基本的算数表达式，op 为相应的关系运算符{<，≤，>，≥，==，!=}。一个判定分支节点的覆盖情况计算如表 1 所示。

表 1　分支距离计算

表达式	真	假
a==b	abs(a-b)	a==b? k:0
a!=b	a!=b? k:0	a!=b? abs(a-b):0
a<b	a<b? 0:a-b+k	a<b? a-b+k:0
a≤b	a≤b? 0:a-b	a≤b? a-b:0
a>b	a>b? 0:a-b+k	a>b? a-b+k:0
a≥b	a≥b? 0:a-b+k	a≥b? a-b+k:0
a‖b	min[fit(a),fit(b)]	fit(a)+fit(b)
a&&b	fit(a)+fit(b)	Max[fit(a),fit(b)]

传统的分支函数构造法需要对每一条路径分别设计距离计算公式，而当程序分支较多时，这种设计步骤相当繁琐

且效率低下。因此,本文在分支函数构造方法的基础上设计一种基于数组存储的方式来构造及保存适应度值。

程序中的每一个分支节点对应数组中的一个元素,并将数组中每个元素定义为两种状态,一种是该分支未执行过,赋值为'0';另一种是执行过,将通过分支距离法计算得到的值作为当前数组对应元素的值。这种设计方法消除了分支距离法每条路径都要单独设计的弊端,更具有实用性。

3.2 算法描述

本文中的算法以生成满足单路径覆盖准则的最优测试数据为目的,同时以达到迭代次数上限或找到最优测试数据为算法搜索的终止条件。改进的算法伪代码如图2所示。

Algorithm 1 改进的混合蛙跳算法

相关参数设置: N: 种群大小, P: 种群, d: 变量维数, L:局部搜索最大迭代次数, G: 种群迭代次数

输出: 最优测试数据

```
function ISFLA(Array, left, middle, right)
    for i = 0 → N do
        P = Initialization()
    end for
    P = P.Sort()
    Xg = P[0] //全局最优解
    for k = 0 → G do
        Group(P)
        for i = 0 → m do
            for j = 0 → L do
                Xb, Xw
                X = Learning(Xb, Xw)
                if F(X) < F(Xw) then
                    Xw = X
                else
                    X = Learning(Xg, Xw)
                    if F(X) < F(Xw) then
                        Xw = X
                    else
                        Xw = Learning(Xrand, Xw)
                    end if
                end if
            end for
        end for
        P = reSort()
        Xg = P[0] //每一次迭代更新全局最优解
    end for
    return result
end function
```

图2 ISFLA伪代码

4 实验分析

4.1 实验参数设置

选取5个基准程序来验证ISFLA的性能,并将实验结果与标准混合蛙跳算法,布谷鸟搜索算法和粒子群优化算法进行比较。5个测试程序的详细信息如表2所示。

第1个程序待覆盖的目标路径为输入的year同时满足对4和100取余等于0。第2个程序待覆盖的目标路径为age=22的判定;第3个程序要覆盖的目标路径为是否为等边三角形的判断;第4个程序要覆盖的目标路径为满足不胖不瘦条件的身高体重,最后一个验证程序要覆盖的目标路径为找到符合条件的个位数。

实验中4种算法的相关参数设置如表3所示。

考虑到初始化种群规模对于算法性能的影响,将种群规模依次设为30、50、70和100。为保证实验的公平性,除去算法本身所特有的一些参数外,所有共用的参数都设为相同的值。本文将算法找到最优解时的迭代次数的均值作为比较算法性能优劣的标准。每个程序均进行50次寻优操作,记录这50次寻优中最优解出现的次数以及出现的代数。若该次没有找到最优解,迭代次数按照式(12)记录。

表 2　测试程序相关信息

相关信息	判断闰年	允许查看范围	三角形分类	标准体重判断	找到数据的各位数
参数个数	1	2	3	2	1
取值范围（从左至右）	[1,10 000]	[−100,500] [0,100]	[0,60] [0,60] [0,60]	[0,200] [0,200]	[−99 999,99 999]
程序描述	判断闰年	根据年龄判断查找范围	判断三角形的形状	找出满足不胖不瘦的体重	找出满足条件的个位数

表 3　算法相关参数设置

参数	混合蛙跳算法	布谷鸟搜索算法	粒子群优化算法	ISFLA
种群大小	30,50,70,100	30,50,70,100	30,50,70,100	30,50,70,100
迭代次数	100	100	100	100
被发现的可能性	——	0.5	——	——
速度最大值	——	——	5	——
速度最小值	——	——	−5	——
寻优次数	50	50	50	50

$$Iter = N \times I + N \tag{12}$$

其中，N 为表 3 中的种群大小，I 为表 3 中的迭代次数。

4.2　实验分析

通过 5 个基准程序分别比较在标准混合蛙跳算法、布谷鸟搜索算法、粒子群优化算法及改进的混合蛙跳算法中测试数据的生成情况。

对于不同的测试程序，分别记录 4 种算法在 50 次寻优操作中，种群规模分别为 30、50、70、100 时找到能够覆盖目标路径的测试数据时的平均迭代次数。其中判断闰年的测试程序中将年份限制在[0,10 000]；允许查看范围的测试程序中编号取值为[−100,500]，年龄输入范围为[0,100]；判断三角形类型中三条边的输入范围为[0,60]，且三角形的三条边是在排序后进行输入的。标准体重判断中将身高体重限制在 0 到 200 之间；求整型数据的各位数字时，设计数据的输入范围为[−99 999,99 999]。设置适当的输入范围，降低因阈值的选择对算法搜索效率造成的影响。

种群大小为 30 时，算法在 5 个基准程序上的对比的实验结果如表 4 所示。从表中可知，与标准的混合蛙跳算法，布谷鸟搜索算法及粒子群优化算法相比，改进的算法在测试数据生成的平均迭代次数最少，表明了在当前的种群规模下，改进的混合蛙跳算法的查找性能要优于其它算法。当测试基准程序为判断闰年时，4 种算法对应的平均迭代次数波动性较小，这也说明了当基准程序的结构及流程的复杂程度较低时，算法的选取对于测试数据的生成影响较小。

表 4　ISFLA 与其他算法在 5 个程序上的比较

种群大小：30				
算法	平均迭代次数			
	混合蛙跳算法	布谷鸟搜索算法	粒子群优化算法	ISFLA
判断闰年	1.02	1.0	1.0	1.04
根据年龄判断查找范围	16.92	55.52	86.82	**10.12**
判断三角形的形状	46.12	69.06	88.34	**13.04**
找出满足不胖不瘦的身高和体重	19.14	17.2	86.46	**7.1**
找出满足条件的个位数	83.64	97.58	96.68	**82.38**

种群大小为 50 时，算法在 5 个基准程序上的对比的实验结果如表 5 所示。从表中可知，与混合蛙跳算法，布谷鸟搜索算法及粒子群优化算法相比，ISFLA 在测试数据生成的平均迭代次数最小。与表 4 相比，表 5 中每个算法对应的基准程序得到的平均迭代次数更小，这说明了当种群规模增加时，算法有更大的可能性找到最优解。这种情况在混合蛙跳算法，布谷鸟搜索算法及 ISFLA 中更为明显。

表5 ISFLA与其他算法在5个程序上的比较

种群大小:50				
算法	平均迭代次数			
	混合蛙跳算法	布谷鸟搜索算法	粒子群优化算法	ISFLA
判断闰年	1.0	1.02	1.0	1.0
根据年龄判断查找范围	9.2	52.8	72.74	**6.26**
判断三角形的形状	31.34	55.26	92.12	**8.98**
找出满足不胖不瘦的身高和体重	9.78	8.22	40.12	**3.74**
找出满足条件的个位数	83.7	94.2	92.62	**62.56**

种群大小为70时,算法在5个基准程序上的对比的实验结果如表6所示。表中的数据更加验证了当种群规模较大时,算法有更大的可能性找到最优解之一结论。

表6 ISFLA与其他算法在5个程序上的比较

种群大小:70				
算法	平均迭代次数			
	混合蛙跳算法	布谷鸟搜索算法	粒子群优化算法	ISFLA
判断闰年	1.0	1.0	1.42	1.0
根据年龄判断查找范围	7.76	62.26	82.28	**5.48**
判断三角形的形状	27.0	71.36	80.36	**5.66**
找出满足不胖不瘦的身高和体重	9.24	8.38	39.9	**3.06**
找出满足条件的个位数	77.66	84.46	96.14	**53.18**

种群大小为100时,算法在5个基准程序上的对比的实验结果如表7所示。从表中可知,与混合蛙跳算法,布谷鸟搜索算法及粒子群优化算法相比,ISFLA在测试数据生成的平均迭代次数最小,表明了在当前的种群规模下,改进的混合蛙跳算法的查找性能要优于其它算法。与其他表中的数据相比,ISFLA对应的平均迭代次数更小,说明了合适的种群规模的设定对于算法的进化起到正向推动作用。

表7 ISFLA与其他算法在5个程序上的比较

种群大小:100				
算法	平均迭代次数			
	混合蛙跳算法	布谷鸟搜索算法	粒子群优化算法	ISFLA
判断闰年	1.0	1.0	1.0	1.0
根据年龄判断查找范围	4.2	63.16	62.94	**3.3**
判断三角形的形状	15.36	73.9	70.86	**4.66**
找出满足不胖不瘦的身高和体重	4.54	3.94	31.64	**2.4**
找出满足条件的个位数	82.66	83.62	98.02	**42.5**

综上所述,改进的混合蛙跳算法在一定程度上提高了算法的收敛速度,避免了因陷入局部最优而导致算法搜索停滞不前,能够在更快的时间内找到覆盖某条目标路径的最优测试数据。同时设计合适的种群规模将更有助于算法寻找到最优解。

5 结 论

本文通过将混合蛙跳算法用于解决测试数据生成问题,提出了ISFLA算法,通过引入动态阈值平衡算法的全局开发和局部搜索能力,改进原有的随机更新策略,增强种群之间的信息交流。实验结果表明该算法在测试数据的生成上有着不错的寻找最优解的能力,且收敛性较强。这说明ISFLA对于测试数据自动生成具有一定的工程应用价值,是一种有效的测试数据生成方法,可以为自动化测试提供新思路。

参考文献

[1] HE Z T, LIU C, YAN H H. Software testing evolution process model and growth of software testing

quality[J]. Science China (Information Sciences), 2015, 58(3): 196-201.
[2] EDGAR S M, RAQUEL M, TAMAYO P. A review of reality of software test automation [J]. Computacion y Sistemas, 2019, 23(1): 169-183.
[3] 薛猛，姜淑娟，王荣存. 基于智能优化算法的测试数据生成综述[J]. 计算机工程与应用，2018，54(17)：16-23.
[4] ZHU E, YAO C, MA Z, et al. Study of an improved genetic algorithm for multiple paths automatic software test case generation[J]. Springer, Cham, 2017,DOI:10.1007/978-3-319-61824-1_44.
[5] XING Y, GONG Y, ZHOU X, et al. A hybrid backtracking algorithm for automatic test data generation[J]. Tehnicki Vjesnik, 2017, 24(3): 761-768.
[6] ESNASHARI M, DAMIA A H. Automation of software test data generation using genetic algorithm and reinforcement learning[J]. Expert Systems with Applications, 2021 (5): 115446, DOI: 10.1016/j.eswa.2021.115446.
[7] HAN X, LEI H, WANG Y S. Multiple paths test data generation based on particle swarm optimisation[J]. Iet Software, 2017, 11(2): 41-47.
[8] 袁光辉，刘兆春. 基于混合遗传算法在测试用例生成中的研究[J]. 黑龙江工业学院学报(综合版)，2019，19(10)：33-38.
[9] 张馨俸. 基于人工智能的测试用例自动生成研究及应用[D]. 青岛:山东科技大学，2020.
[10] HUANG H, LIU F, ZHUO X, et al. Differential evolution based on self-adaptive fitness function for automated test case generation [J]. IEEE Computational Intelligence Magazine, 2017, DOI: 10.1109/MCI.2017.2670462.
[11] 阳同光，蒋新华，付强. 混合蛙跳脊波神经网络观测器电机故障诊断研究[J]. 仪器仪表学报，2013，34(1)：193-199.
[12] 戴月明，张明明，王艳. 协同进化混合蛙跳算法[J]. 计算机工程与科学，2018，40(1)：139-147.
[13] 郭小燕，刘学录，王联国. 基于混合蛙跳算法的土地利用格局优化[J]. 农业工程学报，2015，31(24)：281-288,63.
[14] 鲁建厦，翟文倩，李嘉丰，等. 基于改进混合蛙跳算法的多约束车辆路径优化[J]. 浙江大学学报(工学版)，2021，55(2)：259-270.
[15] 崔文华，刘晓冰，王伟，等. 混合蛙跳算法研究综述[J]. 控制与决策，2012，27(4)：481-486,493.

作者简介

刘会颖，硕士研究生，主要研究方向为软件测试。
E-mail:1115194813@qq.com

刘紫阳(通信作者)，硕士，讲师，主要研究方向为深度学习与智能优化算法。
E-mail: 2210903484@qq.com