智能优化在软件测试中的应用综述

姚香娟[1,2],田甜[3], 党向盈[4], 孙百才[5,1], 巩敦卫[5†]

(1. 中国矿业大学 数学学院, 江苏 徐州 221116;
2. 南京大学 计算机软件新技术国家重点实验室, 南京 221116;
3. 山东建筑大学 计算机科学与技术学院, 济南 250101;
4. 徐州工程学院 信息工程学院(大数据学院), 江苏 徐州 221018;
5. 中国矿业大学 信息与控制工程学院, 江苏 徐州 221116.)

摘 要: 软件测试是软件开发活动中一个关键且耗时的环节,其核心是生成满足特定准则的测试数据.随着软件复杂程度的不断增加,软件测试的难度也越来越高.使用遗传算法等智能优化方法解决复杂软件的测试问题,是近年来软件工程领域的一个研究热点.本文主要针对智能优化在软件测试的应用进行综述.首先,介绍了软件测试的基本原理和方法.然后,介绍了智能优化在不同测试领域的研究进展.接着,对基于不同智能优化方法的软件测试研究进展进行了分析.最后,给出该领域的挑战与展望.

关键词: 智能优化; 优化算法; 软件测试; 测试数据生成; 进化测试

中图分类号: TP273 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2021.1361 **开放科学(资源服务)标识码(OSID):**

Review on the application of intelligent optimization in software testing

YAO Xiangjuan[1,2], *TIAN Tian*[3], *DANG Xiangying*[4], *SUN Baicai*[5,1], *GONG Dunwei*[5†]

(1. School of Mathematics, China University of Mining and Technology, Xuzhou 221116, China; 2. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China; 3. School of Computer Science and Technology, Shandong Jianzhu University, Jinan 250101, China; 4. School of Information Engineering (School of Big Data), Xuzhou University of Technology, Xuzhou 221018, China; 5. School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, Jiangsu 221116, China)

Abstract: Software testing is a critical and time-consuming process during software development, whose key is to generate test data that meet specific criteria. With the increasing complexity of software, software testing is becoming more and more difficult. Recent years, it is a hot topic in software engineering using intelligent optimization, such as genetic algorithms, to test complex software. This paper mainly summarizes the application of intelligent optimization in software testing. Firstly, the basic principles and methods of software testing are introduced. Then, the research progress of intelligent optimization in different testing fields is introduced. Next, the research progress of software testing based on different intelligent optimization methods is analyzed. Finally, the challenges and prospects in this field are given.

Keywords: Intelligent optimization; Optimization algorithm; Software testing; Test data generation; Evolutionary testing

0 引 言

计算机软件是信息产业的重要组成部分,对国民经济、社会发展和国防建设都起着至关重要的作用,其可靠性一直是学术界和产业界关注的焦点[1].提高软件质量的重要途径之一,是在软件投入使用之前进行大量测试,以及时发现其中存在的缺陷甚至错误.软件测试包括很多环节,如生成测试数据、执行被测软件、评价测试结果等.

随着软件规模不断扩大、复杂程度不断增加,软件存在缺陷或者错误的可能性逐步提高.另一方面,人们对软件质量的要求也在不断增强.因此,软件测试所耗费的人力、物力越来越多.如果可以将测试的各个环节自动化,将会大幅度提高软件测试的效率、降低软件测试的成本[2].

基于搜索的软件工程(Search Based Software Engineering,简称SBSE)是软件工程和智能优化相互交叉的新兴研究领域,主要采用遗传算法等智能搜索

收稿日期: 2021-08-04; **录用日期:** 2021-10-27.

基金项目: 国家重点研发计划项目(2018YFB1003802), 国家自然科学基金重点项目(62133015), 中央高校基本科研业务费专项资金(2020ZDPYMS40), 山东省自然科学基金(ZR2020MF084).

†通讯作者. E-mail: dwgong@vip.163.com

技术解决软件工程相关问题[3].该方向自的被提出以来,就引起人们的极大兴趣,并在软件测试数据自动生成、缺陷定位、程序错误自动修复等方面取得丰富的研究成果,有效促进了软件工程的快速发展[4].

尽管智能优化方法可以克服传统测试方法存在的一些问题,但是,该领域仍然存在诸多挑战[5].本文的主要目的是通过综述已有成果,归纳出该领域的主要研究方向和发展趋势.

本文首先介绍了软件测试的基本原理和方法.然后,介绍了智能优化在不同测试领域的研究进展.接着,对基于不同智能优化方法的软件测试研究进展进行了分析.最后,给出该领域的挑战与展望.

1 软件测试的基本原理和方法

软件测试是在软件投入运行前,对软件需求分析、设计规格说明和编码的最终复审,是确保软件质量的关键步骤[6].

1.1 软件测试的定义

1979年,Myers给出软件测试的定义为:软件测试是为了发现错误而针对某个程序或系统的执行过程.1983年,IEEE给出软件测试的定义为:使用人工或自动手段来运行或测试某个软件系统的过程,其目的在于检测其是否满足规定的需求,或弄清被测系统的预期结果与实际结果之间的差别[7].

可以看出,软件测试的核心是检查软件的各项功能特性和非功能特性,从而保证软件的正确性.因此,测试是软件质量保证的关键,也是成功实现软件开发目标的重要保障.

1.2 软件测试的分类

按照软件开发的不同阶段,软件测试可以分为单元测试、集成测试、确认测试和系统测试[8].单元测试主要针对程序的最小组成单元(即模块)进行正确性检验.集成测试把已经通过单元测试的模块进行组装,再对组装模块的体系结构进行测试.确认测试主要检查成品软件是否能完成需求规格说明规定的各种功能、软件配置是否正确等.系统测试则把已经完成确认测试的软件投入实际运行环境,与其它系统组合在一起进行测试.

按照测试过程是否需要程序代码,软件测试可以分为白盒测试、黑盒测试和灰盒测试.白盒测试也称结构测试,主要依据程序代码生成测试数据;黑盒测试也称功能测试或数据驱动测试,不需要程序代码,只需要程序应该完成的功能或满足的约束;灰盒测试则是黑盒测试和白盒测试的融合.

1.3 测试数据生成

软件测试的核心是:生成满足特定准则的一组测试数据,并以这些数据为输入运行程序,以发现程序所包含的缺陷或错误.

被测程序的一个输入变量,指的是程序的输入语句或输入参数中出现的变量.如果程序G的输入变量为$x_1, x_2, \cdots, x_k$,那么,称$\mathbf{X} = (x_1, x_2, \cdots, x_k)$为$G$的输入向量.$\mathbf{X}$的一个具体值称为$G$的一个输入.如果$x_i$的取值域为$D_i$,那么$G$的取值域为$D = D_1 \times D_2 \times \cdots \times D_k$,这里,“$\times$”表示空间的笛卡尔积.

测试数据是指被测程序满足特定覆盖要求的某个输入,往往需要附带说明该输入的执行条件和预期结果.为了生成测试数据,首先需要根据测试要求选择合适的测试充分性准则,然后再利用一定的策略生成满足该准则的测试数据.

测试数据生成往往需要耗费巨大的人力、物力和时间,自动求解上述问题将有效减轻测试人员的劳动强度.总体来讲,有4种测试数据自动生成方法,分别为随机法、静态法、动态法和试探法.

随机法通过对被测程序的输入空间随机采样,生成满足特定覆盖要求的测试数据[9].该方法简单易行、开销小,但具有很大的盲目性.静态法仅对程序进行静态的分析和转化,而不涉及程序的实际运行[10].动态法基于被测程序的实际运行生成测试数据,其过程是确定的[11].与动态法不同的是,试探法虽然也基于被测程序的实际运行,但是,生成测试数据的过程往往依赖于某种搜索算法,比如遗传算法、粒子群算法、模拟退火算法等[12].

1.4 基于智能优化的软件测试

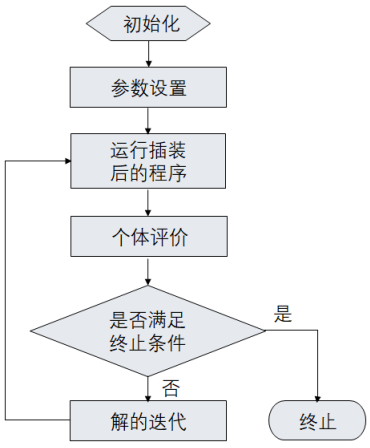

图 1 基于智能优化的软件测试流程

智能优化算法又称启发式算法,是一类具有全局优化性能、通用性强且适合于并行处理的算法[13].使用智能优化方法解决软件测试问题的主要思想是:根

据某种给定的测试准则,定义合适的适应度函数,从而将测试数据生成问题转化为函数优化问题;再利用某种优化方法,得到优化问题的最优解,即为满足要求的测试数据.其主要流程如图1所示.

采用智能优化算法解决测试数据自动生成问题,成为目前国内外软件工程和进化优化界研究的前沿之一,并产生了可喜的研究成果.

1.5 基于智能优化的软件测试研究现状分析

我们以软件测试、结构测试、功能测试、变异测试、并行程序测试、进化、启发式、元启发式、基于搜索、遗传算法、爬山算法、禁忌搜索、模拟退火、粒子群算法、蚁群算法等作为关键字,分别用中文或英文检索了7个学术资源数据库:中国知网、万方、Web of science、IEEE Xplore、ScienceDirect和Springer,ACM library,时间跨度为1995年到2021年.对搜索到的论文进行筛选,共汇总了519篇有关智能优化在软件测试的应用方面的论文.

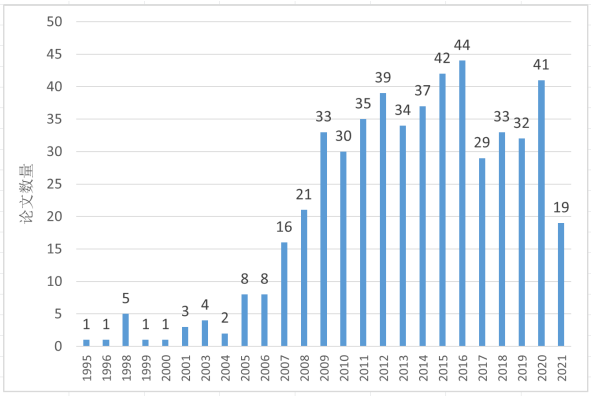

图 2 论文分布情况(按时间)

按照发表时间,论文分布情况如图2所示.我们可以看出,该领域的论文数量近年来居高不下.虽然2017-2019年有所下降,但2020年再次回升.

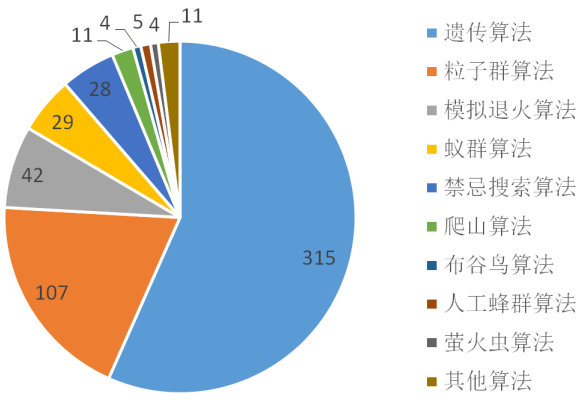

图 3 论文分布情况(按方法)

这些论文使用的智能优化方法主要包括:遗传算法、粒子群算法、模拟退火算法、蚁群算法、搜索禁忌算法、爬山算法、布谷鸟算法、人工蜂群算法和萤火虫算法等,其分布情况如图3所示.从图3可以看出,使用最多的智能优化方法是遗传算法,其次是粒子群算法.模拟退火算法、蚁群算法和禁忌搜索算法的使用也较广.

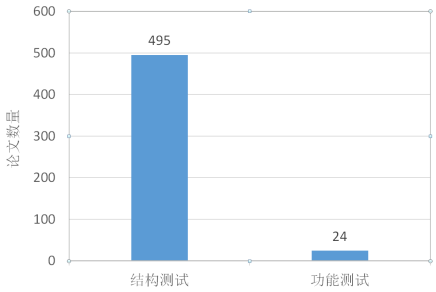

图 4 论文分布情况(按领域)

将论文按结构测试和功能测试两个主要领域进行汇总,论文分布情况如图4所示.从图4可以看出,目前智能优化主要应用于结构覆盖测试,在功能测试领域的应用还很少.

2 智能优化在不同测试领域的研究进展

2.1 智能优化在结构测试的应用研究

结构覆盖测试包含多种测试充分性准则,如语句覆盖、分支覆盖、路径覆盖等[14,15].复杂软件的数据类型繁多,输入空间庞大,软件内部结构复杂,因此,测试数据生成一直是重要且困难的问题之一.Xanthakis等首先将进化算法应用于软件测试数据生成,解决路径覆盖问题[16],开创了利用智能优化方法生成软件测试数据的先河.

为了采用智能优化解决测试数据生成问题,需要先将其转化为一个函数优化问题,以评价个体的适应度.典型的适应度函数通常包括如下两部分:层接近度和分支距离.首先介绍要用到的基本概念.

控制流图是程序控制结构的图形表示,是一种具有如下结构的有向图$G=(N,E,s,e)$,其中,N的元素称作G的节点,对应程序的某一或几条语句;E的元素$e_{ij}=(s_i,s_j)$称为G的边,表示从节点s_i到s_j存在控制流.每个程序的控制流图还包含惟一的入口节点s和出口节点e.

路径P是指一个节点序列$s_1,s_2,\cdots,s_k$,满足从节点s_i到s_{i+1}有边存在,$i=1,2,\cdots,k-1$.

层接近度用于衡量测试数据$\mathbf{X}$穿越的路径与测试目标的偏离程度,记为$appr(\mathbf{X})$.分支距离用于衡量使一个谓词为真(或假)的条件满足程度.对于不同类型的简单分支谓词,分支距离的表达式如表1所列[17].

表 1 简单谓词的分支距离

分支条件	分支距离	
	取真时	取假时
$a \geq b$	0	$\lvert a-b\rvert$
$a > b$	0	$\lvert a-b\rvert+0.1$
$a \leq b$	0	$\lvert a-b\rvert$
$a < b$	0	$\lvert a-b\rvert+0.1$
$a = b$	0	$\lvert a-b\rvert$
$a \neq b$	0	1

一般情况下,个体的适应度函数为层接近度和归一化的分支距离之和,即:

$$fit(\mathbf{X}) = appr(\mathbf{X}) + Narmal(dist(\mathbf{X}))$$

Arcuri[18]在计算分支距离时提出了一种新的归一化函数,从而解决分支覆盖问题.Panichella等[19]将分支覆盖问题看作一个多目标优化问题,并采用多种群遗传算法进行求解.

Bueno和Jino[20],以及Lin和Yeh[21]分别利用遗传算法生成路径覆盖测试数据.但是,上述方法每次只能覆盖一条目标路径.Ahmed和Hermadi[22]将多路径覆盖测试数据生成问题转化为多目标优化问题,使得运行一次遗传算法,就能生成覆盖多条路径的测试数据.但是,当需要覆盖的路径很多时,该模型的求解复杂度会很高.Gong等[23,24]针对多路径覆盖问题,通过路径分组,简化了优化模型,从而降低了测试数据的生成难度.Yao和Gong[25]提出一种基于个体信息分享的遗传算法,用于生成覆盖多条路径的测试数据.巩和张[26]提出一种基于自适应分组的大规模路径覆盖测试数据进化生成方法.

要得到个体的适应度,需要使用测试数据执行程序.为了降低执行程序的代价,姚等[15]提出一种融入神经网络的路径覆盖测试数据进化生成方法,使用神经网络作为代理模型对个体的适应度进行模拟.Sun等[27]构建了针对并行程序的代理模型,估计测试数据的适应度,减少程序实际运行次数.另外,巩等[28]还利用代理模型评估某一流程适应度分量,从而选择估计适应度高的测试数据运行程序.

2.2 智能优化在变异测试方面的应用研究

变异测试最早由Hamlet[29]和DeMillo等[30]提出,基本原理是:首先,采用变异算子对被测程序做微小的合乎语法的变动,例如,将关系运算符“>”替换为“<”,从而产生大量的新程序,每个新程序称为一个变异体;然后,基于相同的测试数据运行原程序和变异体,并比较二者输出的异同.如果不同,就认为测试数据将该变异体杀死.

与传统的结构化测试相比,变异测试具有更强的缺陷检测能力[31].然而,变异测试也存在变异体数量巨大、生成测试数据代价昂贵等问题[32].将智能优化应用于变异测试,有望取得更高的测试效率.

Silva等[33]关于变异测试数据生成方法的统计中,19篇文献中的6篇均采用遗传算法.Pérez和Inmaculada[34]基于遗传算法解决增加测试数据和减少变异体数量之间的优化问题.张功杰等[35]以测试数据集作为决策变量,并采用集合进化遗传算法生成变异测试数据.Souza等[36]基于强变异测试准则,采用爬山法生成变异测试数据.Ayari等[37]提出一种元启发式算法,将蚁群优化方法应用于变异测试,检查测试数据的质量,并生成测试数据集.

2.3 智能优化在并行程序测试的应用研究

并行程序包含多个执行流程,流程之间通过内存共享和消息传递进行交互.由于并行程序的复杂度和出错率高,其测试比起传统软件更为复杂.

Letko[38]使用基于搜索的技术控制噪声发生器,以检测并发缺陷.Bhattacharya等[39]利用爬山法和模拟退火方法优化共享变量访问顺序,以期暴露更多缺陷.Qi和Zhou[40]在拆分代码时,使用启发式策略生成细粒度同步序列,检测并发故障.Chin等[41]针对部分可疑代码,搜索导致错误的流程交互,使用循环启发式策略避免状态爆炸.此外,Shousha等[42]以并行程序UML图为基础,用遗传算法来检测死锁.

上述工作侧重于检测并发故障.智能优化也用于解决并行程序的结构覆盖问题[43].Hsu和Chung[44]使用启发式方法选择满足节点或分支覆盖标准的并发任务路径集.对于多路径覆盖问题,Gong等[45]利用与流程相关的变量改进路径相似度度量,从而实现路径的合理分组.Gong等[46]和Sun等[47]基于一定的评价方法,选择可行通信顺序来覆盖目标语句和路径.Tian等[48]将通信顺序作为决策变量的一部分,利用遗传算法同时生成测试数据和通信顺序.Gong等[49]利用反馈定向遗传算法生成路径覆盖测试数据.

此外,人们还将智能优化用于并行程序的变异测试问题.Silva[50]针对并行程序设计了与个体编码相符合的遗传操作.Cao等[51]在进化优化方法中融入符号执行,在产生满足弱变异测试准则测试数据的前提下,使用进化优化方法生成满足强变异测试准则的测试数据.

可以看出,与串行程序相比,智能优化在并行程

序的研究成果相对较少.主要原因在于,并发、交互和不确定等特性使得并行程序的测试难度增大;另外,并行程序的实现途径多,通常涉及具体的并发环境和技术,导致对测试研究人员的背景知识要求较高.充分利用智能优化的优势,更有效的解决并行程序的测试问题是非常必要的.

2.4 基于智能优化的软件测试工具

虽然智能优化算法在软件测试领域得到广泛应用,但是,特别有效的测试工具还很少.

Pargas等[52]基于遗传算法,开发了一款名为TGen的工具,用于生成满足分支覆盖、路径覆盖等准则的测试数据. Lakhotia等[53]开发了面向C语言的测试工具:AUSTIN. 该工具支持三种搜索算法:随机搜索、爬山算法和结合了符号执行的爬山算法.EvoSuite则是Fraser 和Arcuri[54]针对Java语言设计的,所产生的测试用例符合Junit 标准.此外,还有ET-S[55]、eToc[56]、EVACON[57]等.虽然人们热衷于开发多种测试工具,但是,仍然缺乏非常适用且开放的实用工具.

从以上研究可以看出,智能优化为有效解决软件测试问题提供了多种途径,从而为产生可信软件奠定了基础.但是,随着科学技术的不断更新,软件测试面临新的挑战. 首先,软件更新的频率、新软件发布的速度都在不断提高.此外,人工智能、机器学习、互联网等新兴技术不断发展,软件产品的形式也随之发生变化.这就为软件测试带来新的挑战.如何利用智能优化解决新的测试问题,将成为新的研究课题.

3 基于不同智能优化方法的软件测试研究进展

3.1 基于遗传算法的测试

遗传算法(Genetic Algorithm,GA)最早是由美国的Holland于20世纪70年代提出,该算法是模拟达尔文生物进化论的自然选择和遗传学机理而设计的.遗传算法是智能优化方法中应用最早、最广泛的一种算法.

适应度函数是使用遗传算法生成测试数据的关键.Miller和Spoone[58]首次设计了针对分支覆盖的目标函数.Tracey等[59]结合期望覆盖分支的控制依赖节点数与进化个体实际执行的控制依赖节点数来设计目标函数.Wegener 等[60]引入层接近度来评价测试数据,大大提高了进化测试的性能.Lin等[21]设计了一种新的目标函数,并基于海明距离判定是否生成覆盖目标路径的测试数据.Nirpal和Kale[61]基于新的目标函数使用遗传算法生成满足路径覆盖准则的测试用例.

遗传算子是遗传算法的核心,很大程度上决定了算法的搜索能力和求解效率.姬峰[62]通过交叉算子和变异算子的自适应变化,保证了种群具有较强的进化能力和多样性,解决了算法早熟问题.戚荣志等[63]为实现种群的多样性,每隔一定进化代数选择一些个体在种群间移动.Yao和Gong[25]通过个体信息共享来提高搜索到最优解的效率.Michael等[2]利用遗传算法解决条件覆盖问题时,使用两种不同的遗传算法代替局部搜索来生成满足覆盖准则的测试数据集.

基于数据流的结构覆盖测试需要覆盖所有的定义-使用对[64].Girgis[65]首先运用遗传算法生成满足所有使用覆盖准则的测试数据.Ghiduk等[66]发现在某些情况下Girgis 的方法无法判定测试数据的优劣,为此,设计了一种新的多目标函数,根据测试数据与数据流需求中的定义和使用关系来评估测试数据的质量.针对面向对象程序的数据流结构覆盖测试,Vivanti等[67]使用遗传算法高效生成满足需求的测试数据.此外,Denaro 等[68]也使用遗传算法来生成测试数据集,以满足面向对象系统的数据流覆盖测试.

3.2 基于粒子群算法的测试

粒子群算法(Particle Swarm Optimization,PSO)是由Kennedy和Eberhart在1995年提出的一种基于群体智能的元启发式搜索算法.

Agarwal和Srivastava[69]使用离散粒子群算法生成满足分支覆盖准则的测试数据.针对粒子群算法易陷入局部最优等问题,史等[70]设计了新的目标函数,并使用约简自适应粒子群算法自动生成测试数据.Mao[71]使用粒子群算法高效生成满足分支覆盖准则的测试数据集,并验证了和进化算法相比,粒子群算法在分支覆盖率和求解速度方面更具优势.针对粒子群算法算法搜索精度低等问题,Jiang等[72]提出了一种约简自适应粒子群算法测试数据自动生成方法,明显提高算法的收敛速度.

针对数据流测试,Singla等[73]提出了一种混合粒子群优化算法,自动生成满足数据流覆盖准则的测试数据.Varshney和Mehrotra[74]基于优势关系和分支距离设计目标函数,并提出了一种混合粒子群优化算法,根据程序数据流依赖性生成测试数据.Ghiduk[75]基于蚁群算法生成包含所有定义使用对的一组最优路径,并使用蚁群算法生成覆盖最优路径组的测试数据.Rauf[76]开发了名为DFG的数据流生成器,利用蚁群算法得到满足所有定义使用覆盖准则的测试数据集.

尽管粒子群算法在某些方面优于遗传算法,但

是,应用于复杂软件,群智能优化算法也存在高测试成本等问题.

3.3 基于模拟退火算法的测试

模拟退火算法来源于固体退火原理,最早的思想是由Metropolis 等人于1953年提出的.

针对分支覆盖测试,Tracey等[77]概述了基于模拟退火算法的测试数据通用生成框架.针对路径覆盖测试,Zhang和Wang[78]提出一种自适应遗传模拟退火算法,并根据Kale[61]的目标函数,高效生成覆盖每一目标路径的测试数据.Man等[79]运用模拟退火等算法逐一生成覆盖每一分支的测试数据.针对变异测试,Wang等[80]提出了一种基于改进模拟退火算法的软件测试方法,保证了测试数据的生成效率.

3.4 基于蚁群算法的测试

蚁群系统(Ant System或Ant Colony System)是由由Dorigo于1992年在他的博士论文中提出的,其灵感来源于蚂蚁在寻找食物过程中发现路径的行为.

针对路径覆盖测试,Li和Lam[81]把总分支距离作为目标函数,提出了一种基于改进蚁群算法和路径覆盖准则的测试数据生成模型.为提高测试数据集的质量,Mao等[82] 提出了基于蚁群算法的测试数据生成框架,生成覆盖所有分支的测试数据集.Biswas等[83]首先使用蚁群算法得到一组最优路径,并继续使用该算法在输入域内生成覆盖路径的测试数据.此后,Mao等[84]提出了离散版蚁群算法,并用于生成满足分支覆盖准则的测试数据,而且指出在测试数据的质量和稳定性方面,离散版本蚁群算法优于模拟退火和遗传算法,可与粒子群算法相媲美.

3.5 基于其它类型智能优化算法的测试

Harman等[85]分析与每一分支相关的变量,并使用爬山算法重点搜索相关变量对应的输入域,从而显著提高测试数据的生成效率. 为了降低强变异测试的代价,Souza[36] 使用爬山法自动生成杀死变异体的测试数据.针对面向对象软件的分支覆盖测试,Arcuri和Yao[86]提出了一种减少面向对象软件输入域的方法,并使用爬山算法高效生成满足覆盖准则的测试数据.Díaz等[87]使用禁忌搜索算法生成满足分支覆盖准则的测试数据集.为进一步提高分支覆盖测试效率,Mao[88]设计了一种新的目标函数,并使用和声搜索算法高效生成覆盖每一分支的测试数据.Eugenia等[89]应用禁忌搜索算法求解分支覆盖问题,研究了参数对算法性能的影响.Srivastava等[90]综合使用布谷鸟算法和禁忌搜索算法来生成测试数据.

通过上述成果可以看出,多种智能优化方法都在软件测试中得以广泛应用,其中,应用最多的是遗传算法、粒子群算法、模拟退火算法.目前,新的智能优化方法层出不穷.如何针对软件测试问题的特点,设计更合适的智能优化方法,进一步提高软件测试的效率和质量,是值得研究的方向.

4 挑战与展望

软件测试是软件开发活动中的一个重要环节,能够极大地提升软件产品的可靠性.智能优化大大提高了软件测试的效率与质量.然而,随着软件行业的不断发展,基于智能优化的软件测试技术,依然面临较多新的问题与挑战.

4.1 大数据和智能优化相结合的自动测试技术

随着物联网、云计算、大数据等新技术的蓬勃发展,软件的内涵不断扩展,从而对软件测试技术提出更多的挑战.如何自动化地获取大量特殊场景的数据,并对数据进行分析和处理,是测试的一个重要环节.因此,大数据环境下的基于搜索的软件工程将成为未来的发展趋势之一.

4.2 基于智能优化的功能测试

智能优化在结构测试方面已经取得很多研究成果.但是,如何使用智能优化解决功能测试问题,成果相对偏少.如前所述,在检索到的519文献中,仅有24篇是关于功能测试的.当前,嵌入式产品新替代硬件不断推出,芯片、器件、单板等硬件以及操作系统、数据库中间件等软件形态多元化.在无法获取源码的场景下,如何利用智能优化进行快速有效的测试,是软件工程面临的新问题.

4.3 基于智能优化的并行软件测试

当前,软件高度并行化,如何充分利用智能优化方法具有的并行性和并行计算资源,进一步提高测试数据生成效率,以及为并行程序生成测试数据,是值得关注和研究的问题.针对并行程序的基于智能优化算法的测试数据生成在未来必将得到显著发展.

5 结 论

将智能优化应用于软件测试,产生了很多可喜的研究成果.本文总结归纳了该领域的研究进展,分析了研究现状,也对存在的挑战及未来可能的一些研究方向进行了探讨.

随着软件规模和复杂程度的不断提高,基于智能优化的软件测试方法仍然存在很多难题,需要不断解决.随着机器学习、大数据等新技术的发展,该方向的研究必然会呈现新的趋势.

参考文献(References)

[1] 李东, 宫云战. 软件测试方法综述[J]. 装甲兵工程学院学报, 2003, 17(2): 9-13.
(Li D, Gong Y. A summary about software testing methods[J]. Journal of Armored Force Engineering Institute, 2003, 17(2): 9-13.)

[2] Michael C C, McGraw G, Schatz M A. Generating software test data by evolution[J]. IEEE Transactions on Software Engineering, 2001, 27(12): 1085-1110.

[3] Harman M, Jia Y, Zhang Y. Achievements, open problems and challenges for search based software testing[C]. Proceedings of International Conference on Software Testing, Verification and Validation. Graz, 2015: 1-12.

[4] Harman M, Mcminn P. A theoretical and empirical study of search-based testing: local, global, and hybrid search[J]. IEEE Transactions on Software Engineering, 2010, 36(2): 226-247.

[5] Bertolino A. Software testing research: achievements, challenges, dreams[C]. Proceedings of Future of Software Engineering. Minneapolis MN, 2007: 85-103.

[6] 蔡开元, 董昭, 刘克. 关于软件可靠性测试的若干问题[J]. 工程数学学报, 2008, 25(6): 967-978.
(Cai K, Dong Z, Liu K. On Several Issues in Software Reliability Testing[J]. Chinese Journal of Engineering Mathematics, 2008, 25 (6): 967-978.)

[7] 曾文. 软件测试基础教程[M]. 中国水利水电出版社, 2016.
(Zeng W. Basic course of software testing[M]. China Water & Power Press, 2016.)

[8] 徐仁佐. 软件可靠性工程[M]. 北京: 清华大学出版社, 2007.
(Xu R. Software reliability engineering[M]. Beijing: Tsinghua University Press, 2007)

[9] Chen T Y, Kuo F C, Merkel R G, et al. Adaptive random testing: the ART of test case diversity[J]. The Journal of Systems and Software, 2010, 83(1): 60-66.

[10] 张德平, 聂长海, 徐宝文. 划分测试中测试用例最优分配问题研究[J]. 南京大学学报(自然科学版), 2005, 41(5): 553-561.
(Zhang D, Nie C, Xu B. Optimal allocation of test case considering test-resource in partition testing[J]. Journal of Nanjing University (Natural Science), 2005, 41 (5): 553-561.)

[11] Callagher M, Narasimhan V L. ADTEST: A test data generation suite for Ada software systems[J]. IEEE Transaction on Software Engineering, 1997, 23(8): 473-484.

[12] Sofokleous A A, Andreou A S. Automatic, evolutionary test data generation for dynamic software testing[J]. The Journal of System and Software, 2008, 81(11): 1883-1898.

[13] 巩敦卫, 孙晓燕. 基于模式定理的遗传算法交叉和变异概率上限[J]. 控制与决策, 2004, 19(005): 554-556, 581.
(Gong D, Sun X. Upper limits of crossover and mutation rate of genetic algorithm based on schema theorem [J]. Control and Decision, 2004, 19 (005): 554-556, 581.)

[14] Yao X, Gong D, Zhang G. Constrained multi-objective test data generation based on set evolution[J]. IET Software, 2015, 9(4):103-108.

[15] 姚香娟, 巩敦卫, 李彬. 融入神经网络的路径覆盖测试数据进化生成[J]. 软件学报, 2016, 27(4): 828-838.
(Yao X, Gong D, Li B. Evolutional test data generation for path coverage by integrating neural network[J]. Journal of Software, 2016, 27 (4): 828-838.)

[16] Xanthakis S, Ellis C, Skourlas C, et al. Application of genetic algorithms to software testing [C]. Proceedings of International Conference on Software Engineering and Its Applications. IEEE, 1992: 625-636.

[17] Korel B. Automated software test data generation[J]. IEEE Transaction on Software Engineering, 1990, 16(8): 870-879.

[18] Arcuri A. It does matter how you normalise the branch distance in search based software testing[C]. Proceedings of International Conference on Software Testing, Verification and Validation. Paris, 2010: 205-214.

[19] Panichella A, Kifetew F M, Tonella P. Reformulating branch coverage as a many-objective optimization problem[C]. Proceedings of IEEE International Conference on Software Testing. Graz, 2015: 1-10.

[20] Bueno P M S, Jino M. Automatic test data generation for program paths using genetic algorithms[J]. Software Engineering and Knowledge Engineering, 2002, 12(06): 691-709.

[21] Lin J C, Yeh P L. Automatic test data generation for path testing using GAs[J]. Information Sciences, 2001, 131(1): 47-64.

[22] Ahmed M A, Hermadi I. GA-based multiple paths test data generator[J]. Computers and Operations Research, 2008, 35(10): 3107-3124.

[23] Gong D, Zhang W, Yao X. Evolutionary generation of test data for many paths coverage based on grouping[J]. Journal of Systems and Software, 2011, 84(12): 2222-2233.

[24] Gong D, Tian T, Yao X. Grouping target paths for evolutionary generation of test data in parallel[J]. Journal of Systems and Software, 2012, 85(11): 2531-2540.

[25] Yao X, Gong D. Genetic algorithm-based test data generation for multiple paths via individual sharing[J]. Computational Intelligence and Neuroscience, 2014, 29 (1): 1-13.

[26] 巩敦卫, 张婉秋. 基于自适应分组的大规模路径覆盖测试数据进化生成[J]. 控制与决策,2011, 026(007): 979-983.
(Gong D, Zhang W. Evolutionary generation of test data for many path coverage based on adaptive grouping[J]. Control and Decision, 2011, 026 (007): 979-983.)

[27] Sun B, Gong D, Tian T, Yao X. Integrating an ensemble

surrogate model's estimation into test data generation[J]. IEEE Transactions on Software Engineering, 2020(99), DOI: 10.1109/TSE.2020.3019406

[28] Gong D, Sun B, Yao X, Tian T. Test data generation for path coverage of MPI programs using SAEO. ACM Transactions on Software Engineering and Methodology, 2021, 30(2): 1-37.

[29] Hamlet R G. Testing programs with the aid of a compiler[J]. IEEE transactions on software engineering, 1977, 3(4): 279-290.

[30] DeMillo R A, Lipton R J, Sayward F G. Hints on test data selection: Help for the practicing programmer[J]. Computer, 1978, 11(4): 34-41.

[31] Nishtha J, Bharti S, Shweta R. Systematic literature review on search based mutation testing[J]. E-Informatica Software Engineering Journal, 2017, 11(1): 59-76.

[32] Wang B, Xiong Y, Shi Y. Faster mutation analysis via equivalence modulo states [C]. Proceedings of International Conference on Software Testing and Analysis. Santa Barbara CA, 2017: 295-306.

[33] Silva R A, Rocio S D S S D, Sergio L D S P. A systematic review on search based mutation testing[J]. Information & Software Technology, 2017, 81(81): 19-35.

[34] Delgado-Pérez P, Inmaculada M B. Search-based mutant selection for efficient test suite improvement: evaluation and results[J]. Information & Software Technology, 2018: 130-143.

[35] 张功杰, 巩敦卫, 姚香娟. 基于变异分析和集合进化的测试用例生成方法[J]. 计算机学报, 2015, 38(11): 2318-2331.
(Zhang G, Gong D, Yao X. Test case generation method based on mutation analysis and set evolution[J]. Chinese Journal of Computers, 2015, 38(11): 2318-2331.)

[36] Souza F C M, Papadakis M, Traon Y L, Delamaro M E. Strong mutation-based test data generation using hill climbing [C]. Proceedings of International Conference on Search Based Software Testing. Austin, TX, 2016: 45-54.

[37] Ayari K, Bouktif S, Antoniol G. Automatic mutation test input data generation via ant colony [C]. Proceedings of International Conference on Genetic and Evolutionary Computation Conference. London, 2007: 1074-1081.

[38] Letko Z. Sophisticated testing of concurrent programs[C]. Proceedings of International Symposium on Search Based Software Engineering. Benevento, 2010: 36-39.

[39] Bhattacharya N, El-Mahi O, Duclos E, et al. Optimizing threads schedule alignments to expose the interference bug pattern[C]. Proceedings of International Symposium on Search Based Software Engineering. Berlin, Heidelberg, 2012: 90-104.

[40] Qi X, Zhou H. A splitting strategy for testing concurrent programs[C]. Proceedings of International Conference on Software Analysis, Evolution and Reengineering (SANER). Guangzhou, 2019: 388-398.

[41] Chin K C, Tsay R S, Wu H I. A heuristic region-based concurrency bug testing approach[C]. Proceedings of International Conference on on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking. Exeter, 2020: 1126-1135.

[42] Shousha M, Briand L C, Labiche Y. A UML/SPT model analysis methodology for concurrent systems based on genetic algorithms[C]. Proceedings of International Conference on Model Driven Engineering Languages and Systems. Berlin, Heidelberg, 2008: 475–489.

[43] Tian T, Gong D. Test data generation for path coverage of message-passing parallel programs based on co-evolutionary genetic algorithms[J]. Automated Software Engineering, 2016, 23(3): 469-500.

[44] Hsu S Y, Chung C G. A heuristic approach to path selection problem in concurrent program testing[C]. Proceedings of The Third Workshop on Future Trends of Distributed Computing Systems. Taipei, 1992: 86-92.

[45] Gong D, Tian T, Wang J, et al. A novel method of grouping target paths for parallel programs[J]. Parallel Computing, 2020, 97: 102665.

[46] Gong D, Zhang C, Tian T, et al. Reducing scheduling sequences of message-passing parallel programs[J]. Information & Software Technology, 2016, 80: 217-230.

[47] Sun B, Wang J, Gong D, et al. Scheduling sequence selection for generating test data to cover paths of mPI programs[J]. Information & Software Technology, 2019, 114(OCT.): 190-203.

[48] Tian T, Gong D, Kuo F C, et al. Genetic algorithm based test data generation for MPI parallel programs with blocking communication[J]. Journal of Systems and Software, 2019, 155: 130-144.

[49] Gong D, Pan F, Tian T, et al. A feedback-directed method of evolutionary test data generation for parallel programs[J]. Information & Software Technology, 2020, 124: 106-318.

[50] Silva R A. Search based software testing for the generation of synchronization sequences for mutation testing of concurrent programs[D]. São Paulo: Universidade de São Paulo, 2018.

[51] Cao L, Zheng W, Hu D, Bai H. Concurrent program semantic mutation testing based on abstract memory model[C]. Proceedings of International Conference on Information and Automation. Lijiang, 2015: 1200-1205.

[52] Pargas R P, Harrold M J, Peck R R. Test-data generation using genetic algorithms[J]. Software testing, verification and reliability, 1999, 9(4): 263-282.

[53] Lakhotia K, Harman M, Gross H. AUSTIN: an open source tool for search based software testing of C programs[J]. Information & Software Technology, 2013, 55(1): 112-125.

[54] Fraser G, Arcuri A. EvoSuite: automatic test suite

generation for object-oriented software[C]. Proceedings of ACM SIGSOFT Symposium and the European Conference Foundations of Software Engineering(FSE), Szeged, Hungary, 2011: 416-419.

[55] Wegener J, Baresel A, Sthamer H. Evolutionary test environment for automatic structural testing[J]. Information & Software Technology, 2001, 43(14): 841-854.

[56] Tonella P. Evolutionary testing of classes[C]. Proceedings of ACM Sigsoft International Symposium on Software Testing and Analysis. Boston, USA, 2004: 119-128.

[57] Inkumsah K, Xie T. Evacon: a framework for integrating evolutionary and concolic testing for object-oriented programs[C]. Proceedings of IEEE/ACM International Conference on Automated Software Engineering. Atlanta, Georgia, USA, 2007: 425-428.

[58] Miller W, Spooner D L. Automatic generation of floating-point test data[J]. IEEE Transactions on Software Engineering, 1976 (3): 223-226.

[59] Tracey N, Clark J, McDermid J, et al. A search-based automated test-data generation framework for safety-critical systems[J]. University of York, 2000: 174-213.

[60] Wegener J, Baresel A, Sthamer H. Evolutionary test environment for automatic structural testing[J]. Information & software technology, 2001, 43(14): 841-854.

[61] Nirpal P B, Kale K V. Comparison of software test data for automatic path coverage using genetic algorithm[J]. International Journal of Computer Science & Engineering Technology, 2012: 42-48.

[62] 姬峰. 基于改进遗传算法的软件测试自动化机制研究[J]. 信息技术, 2019, 10: 88-93.
(Ji F. Research on software testing automation mechanism based on improved genetic algorithms[J]. Information Technology, 2019, 10: 88-93.)

[63] 戚荣志, 王志坚, 黄宜华, 等. 基于Spark 的并行化组合测试用例集生成方法[J]. 计算机学报, 2018, 6: 1064-1079.
(Qi R, Wang Z, Huang Y, et al. Generating combinational test suite with spark based parallel approach[J]. Chinese Journal of Computers, 2018, 6: 1064-1079.)

[64] Frankl P G, Weyuker E J. An applicable family of data flow testing criteria[J]. IEEE Transactions on Software Engineering, 1988, 14(10): 1483-1498.

[65] Girgis M R. Automatic test data generation for data flow testing using a genetic algorithm[J]. Journal of Universal Computer Science, 2005, 11(6): 898-915.

[66] Ghiduk A S, Harrold M J, Girgis M R. Using genetic algorithms to aid test-data generation for data-flow coverage[C]. Proceedings of Asia-Pacific Software Engineering Conference. Nagoya, 2007: 41-48.

[67] Vivanti M, Mis A, Gorla A, et al. Search-based data-flow test generation[C]. Proceedings of International Symposium on Software Reliability Engineering. Pasadena CA, 2013: 370-379.

[68] Denaro G, Margara A, Pezze M, et al. Dynamic data flow testing of object oriented systems[C]. Proceedings of IEEE International Conference on Software Engineering. Florence, 2015, 1: 947-958.

[69] Agarwal K, Srivastava G. Towards software test data generation using discrete quantum particle swarm optimization[C]. Proceedings of the 3rd India Software Engineering Conference. Mysore, 2010: 65-68.

[70] 史娇娇, 姜淑娟, 韩寒, 等. 自适应粒子群优化算法及其在测试数据生成中的应用研究[J]. 电子学报, 2013, 41(8): 1555-1559.
(Shi J, Jiang S, Han H, et al. Adaptive particle swarm optimization algorithm and its application in test data generation [J]. Acta electronica Sinica, 2013, 41 (8): 1555-1559.)

[71] Mao C. Generating test data for software structural testing based on particle swarm optimization[J]. Arabian Journal for Science and Engineering, 2014, 39(6): 4593-4607.

[72] Jiang S, Shi J, Zhang Y, et al. Automatic test data generation based on reduced adaptive particle swarm optimization algorithm[J]. Neurocomputing, 2015, 158: 109-116.

[73] Singla S, Kumar D, Rai H M, et al. A hybrid PSO approach to automate test data generation for data flow coverage with dominance concepts[J]. International journal of advanced science and technology, 2011, 37: 15-26.

[74] Varshney S, Mehrotra M. A hybrid particle swarm optimization and differential evolution based test data generation algorithm for data-flow coverage using neighbourhood search strategy[J]. Informatica, 2018, 42(3): 417-438.

[75] Ghiduk A S. A new software data-flow testing approach via ant colony algorithms[J]. Universal Journal of Computer science and engineering Technology, 2010, 1(1): 64-72.

[76] Rauf A. Data flow testing of UML state machine using ant colony algorithm (ACO)[J]. International Journal of Computer Science and Network Security, 2017, 17(10): 40-44.

[77] Tracey N, Clark J, Mander K. Automated program flaw finding using simulated annealing[C]. Proceedings of the International Symposium on Software Testing and Analysis. Clearwater Beach Florida, 1998: 73-81.

[78] Zhang B, Wang C. Automatic generation of test data for path testing by adaptive genetic simulated annealing algorithm[C]. Proceedings of International Conference on Computer Science and Automation Engineering. Shanghai, 2011, 2: 38-42.

[79] Mann M, Sangwan O P, Tomar P, et al. Automatic goal-oriented test data generation using a genetic algorithm and simulated annealing[C]. Proceedings of

International Conference-Cloud System and Big Data Engineering (Confluence). Noida, 2016: 83-87.

[80] Wang K, Wang Y, Zhang L. Software testing method based on improved simulated annealing algorithm[C]. Proceedings of International Conference on Reliability, Maintainability and Safety. Guangzhou, 2014: 418-421.

[81] Li H, Lam C P. Software Test Data Generation using Ant Colony Optimization[C]. Proceedings of International conference on computational intelligence. Istanbul, 2004: 1-4.

[82] Mao C, Yu X, Chen J, et al. Generating test data for structural testing based on ant colony optimization[C]. Proceedings of International Conference on Quality Software. Xi'an, 2012: 98-101.

[83] Biswas S, Kaiser M S, Mamun S A. Applying ant colony optimization in software testing to generate prioritized optimal path and test data[C]. Proceedings of International Conference on Electrical Engineering and Information Communication Technology. Dhaka, 2015: 1-6.

[84] Mao C, Xiao L, Yu X, et al. Adapting ant colony optimization to generate test data for software structural testing[J]. Swarm and Evolutionary Computation, 2015, 20: 23-36.

[85] Harman M, Hassoun Y, Lakhotia K, et al. The impact of input domain reduction on search-based test data generation[C]. Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. Dubrovnik, 2007: 155-164.

[86] Arcuri A, Yao X. Search based software testing of object-oriented containers[J]. Information Sciences, 2008, 178(15): 3075-3095.

[87] Díaz E, Tuya J, Blanco R, et al. A tabu search algorithm for structural software testing[J]. Computers & Operations Research, 2008, 35(10): 3052-3072.

[88] Mao C. Harmony search-based test data generation for branch coverage in software structural testing[J]. Neural Computing and Applications, 2014, 25(1): 199-216.

[89] Eugenia D, Javier T, Raquel B. Automated software testing using a metaheuristic technique based on tabu search[C]. Proceedings of IEEE International Conference on Automated Software Engineering, Montreal, Canada, 2003: 310-313.

[90] Srivastava P R, Khandelwal R, Khandelwal S, et al. Automated test data generation using Cuckoo Search and Tabu Search(CSTS) algorithm[J]. Journal of Intelligent Systems, 2012, 21(2): 195-224.

作者简介

姚香娟(1975−), 女, 教授, 博士, 博士生导师, 从事基于搜索的软件工程、智能优化理论与方法等研究, E-mail: yaoxj@cumt.edu.cn;

田甜(1987−), 女, 副教授, 博士, 硕士生导师，从事基于搜索的软件工程研究, E-mail: tiantian@sdjzu.edu.cn;

党向盈(1978−),女,副教授, 博士, 从事基于搜索的软件工程、进化算法及应用的研究, E-mail: dangpaper@163.com.

孙百才(1990−), 男, 师资博士后, 博士, 从事基于搜索的软件工程、数据驱动进化优化的研究, E-mail: baicaisun@gmail.com.

巩敦卫(1970−), 男, 教授, 博士, 博士生导师, 从事智能优化理论与方法、基于搜索的软件工程、综合能源系统运行优化等研究, E-mail: dwgong@vip.163.com.