浙江理工大学学报(自然科学版),第 39 卷,第 2 期,2018 年 3 月
Journal of Zhejiang Sci-Tech University (Natural Sciences)
Vol. 39, No. 2, Mar. 2018

DOI:10.3969/j.issn.1673-3851(n).2018.02.016

基于多种群进化算法的测试用例优先级排序研究

张　娜,胡国亨,金瑜婷,史佳炳,包晓安
(浙江理工大学信息学院,杭州 310018)

摘　要:为了提升软件测试的效率,加快软件研发的进度,提出了一种基于多种群进化的测试用例排序算法。该算法首先针对单种群遗传算法容易产生早熟收敛的问题,提出了一种多种群并行进化模型,以增强算法的全局寻优能力;然后根据该模型,结合软件需求覆盖和软件缺陷检测率,综合考虑代码覆盖率、测试用例设计信息和历史执行信息三个方面的因素,提出了一种动态调整测试用例优先级的计算方法。实验结果表明:与传统的面向单一目标覆盖的测试用例优先级排序算法相比,该算法的测试速率和软件缺陷检测能力得到一定的提升。

关键词:回归测试;测试用例优先级;多种群;动态调整
中图分类号:TP311.5　　**文献标志码**:A　　**文章编号**:1673-3851 (2018) 03-0218-06

0　引　言

回归测试作为一种基本的测试手段在软件开发的各个阶段扮演着重要的角色。为了节省回归测试的成本,确保软件在每一版本更新后的回归测试用例集仍具有较强的缺陷检测能力,测试用例优先级技术(Test case prioritization,TCP)得到了广泛的关注和发展。

TCP 的核心技术是通过调整测试用例执行的次序进而提升测试效率[1]。一方面,从测试用例满足测试需求的情况研究测试用例优先级问题已有不少研究,如:Wong 等[2]最先将优先级技术运用于回归测试中,根据测试用例对待检测代码的覆盖程度排序,然后依次执行测试用例;Srikanth 等[3]提出根据测试用例覆盖测试需求的情况以及需求的权重设定其优先级;Kavitha 等[4]在 Srikanth 等的研究基础上,根据需求的实现难度及更迭情况提出基于需求覆盖的测试用例优先级调整策略;Jasz 等[5]则是从测试用例的语句和分支覆盖以及检错能力等,提出了动态信息反馈优先级的排序策略。

另一方面,基于搜索方法解决测试用例的排序问题近年来也得到国内外许多学者的关注,如:Rothermel 等[6]提出了基于缺陷覆盖的优先级技术,与随机排序等其它优先级技术相比,明显提高了测试的效率。Jeffrey 等[7]研究了基于切片的优先级技术,相比于现存的采用需求覆盖的优先级技术,该技术的缺陷检测性能更优秀。贪婪算法是求解 TCP 问题的经典方法之一,利用测试用例执行后的覆盖信息对程序中各个代码模块设置相应的权值,在假设提高程序元素的早期覆盖率有助于提高缺陷检测率的条件下应用贪婪法指导测试用例排序,根据是否引入反馈机制,可分为 Total 策略和 Additional 策略[6]。Zheng 等[8]使用贪婪算法、额外贪婪算法和元启发搜索算法(爬山算法和遗传算法)解决 TCP 调整问题,从软件代码覆盖率、分支覆盖率、语句覆盖率三个方面作为覆盖标准评估比较,实验结果表明:遗传算法和额外贪婪算法具有良好的优化性能,但代码覆盖率不能完全代替代码程序的错误检测能力。Kaur 等[9]提出了遗传算法和粒子群算法相结合的进化方法,以解决回归测试的测试用例优化问题,实验结果从平均错误检测率和代码路径覆盖率两项指标作为评估标准分析验证了该方法的

收稿日期:2017－10－11　　网络出版日期:2017－12－12
基金项目:国家自然科学基金项目(61502430,61379036,61562015);浙江理工大学 521 人才培养计划
作者简介:张　娜(1977－),女,浙江奉化人,副教授,硕士,主要从事自适应软件、软件测试与智能信息处理方面的研究。
通信作者:包晓安,E-mail:baoxiaoan@zstu.edu.cn

有效性。Li 等[10]结合爬山算法,从错误检测率的角度验证了遗传算法用于解决测试用例优化问题的有效性,但局部搜索的效果较差。

李龙澍等[1]提出了多种群的遗传算法处理测试用例排序问题,实验结果较标准遗传算法具有更强的性能和更高的效率。然而,利用传统遗传算法求解 TCP 问题存在早熟收敛、算法后期寻优效率低等缺陷;此外,当前这些测试用例排序方法对排序标准的选取过于单一,往往只考虑代码覆盖率层面的因素,忽视了需求覆盖与缺陷关联对测试用例执行过程中优先级调整的影响。

本文针对以上不足,在传统遗传算法的基础上,引进多种群进化的模型,优化全局搜索能力,避免早熟现象发生。基于该模型,考虑测试用例与软件需求及软件缺陷的关联影响,提出一种以代码覆盖率、测试用例设计信息、测试用例历史执行信息为多目标导向的动态调整测试用例优先级的策略。

1　测试用例优先级评价准则

测试用例优先级是指依据一定的排序准则,对测试用例集进行排序,使得满足测试要求的测试用例优先被执行。TCP 问题的经典形式化描述为:对于给定的测试用例集 T 中所有可能排序的集合 D,以及排序目标函数 f(定义域为 D),找出 T'属于 D,并且满足:

$$(\forall T'')(T''\in D)(T''\neq T')[f(T')\geqslant f(T'')] \quad (1)$$

其中:f 是衡量排序有效性的定量描述,f 值越大,表明测试用例集排序的效果越显著,检测软件缺陷的能力越强。

相比于与使用随机顺序的测试,经过优先级排序后的测试序列能够更快地检测出软件缺陷。研究人员常采用缺陷检测加权百分比(Average of the percentage of faults detected,APFD)度量标准来验证测试用例排序方法的有效性和优越性,由 APFD 可以计算出测试用例驱动程序所检测到的缺陷累计平均比率。APFD 的计算公式为:

$$T_{\mathrm{APFD}}=1-\frac{T_{\mathrm{F1}}+T_{\mathrm{F2}}+\cdots+T_{\mathrm{F}m}}{nm}+\frac{1}{2n} \quad (2)$$

其中:n 表示测试用例集 T 中测试用例数目,m 表示该测试用例集可检测软件缺陷的数量,$T_{\mathrm{F}i}$ 表示经过排序后的测试用例集 T'中首次发现缺陷 i 的测试用例在该序列中的次序。APFD 为非负值,并且 APFD 的值越大,代表该测试排序的检错速度越快。

由于测试人员不可能提前知道测试用例检测的有效性。通常,测试用例的覆盖率越高,其检测软件缺陷的可能性就越大。因此,可以借用测试用例的覆盖率信息来量化目标。现有的评价标准有块覆盖率(Average percentage block coverage,APBC)、语句覆盖率(Average percentage statement coverage,APSC)和判定覆盖率(Average percentage decision coverage,APDC),这些评价方法的计算公式与 APFD 类似[1]。以 APBC 计算公式为例,该计算公式可以表示为:

$$T_{\mathrm{APBC}}=1-\frac{T_{\mathrm{B1}}+T_{\mathrm{B2}}+\cdots+T_{\mathrm{B}m}}{nm}+\frac{1}{2n} \quad (3)$$

其中:$T_{\mathrm{B}i}$ 代表首次覆盖第 i 个代码模块的测试用例在本次执行序列中的次序。

2　基于多种群进化算法的测试用例优先级排序

2.1　模型设计

标准遗传算法(Standard genetic algorithm,SGA)因其自身的优良特性被广泛研究应用,但其问题和不足也逐渐暴露出来,最典型的是个体特性同一化、早熟收敛问题。为了改善上述问题,多种群遗传算法(Multiple population genetic algorithm,MPGA)被应运提出。MPGA 在 SGA 的基础上进一步优化,其计算步骤为:初始化时通过引入多个并行进化的种群,扩大了种群规模,进化过程中各子种群根据自身进化趋势的不同,引入特定的遗传算子,达到不同的搜索目的;通过设置迁移算子保证各个种群之间的优良信息的分享交流,保证种群将个体多样性的同时,加快收敛速度;设置合适的人工算子选择并保存进化产生的优秀个体,通过计算各种群中最优个体的最大保持代数,判断算法是否终止,一定程度上避免算法陷入局部最优。本文设计的多种群并行进化模型如图 1 所示。

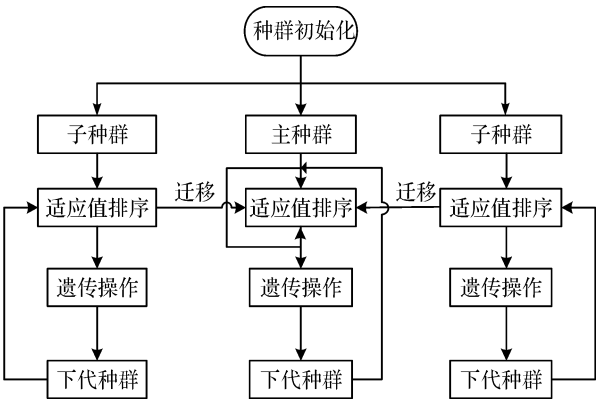

图 1　多种群并行进化模型

本文的多种群模型采用三个种群并行进化的方式搜索最优解，主种群采用基本的遗传搜索方式，并以主种群为基准给不同的子种群设置不同的交叉率和变异率(一个高于主种群，另一个低于主种群)，从而统筹局部与全局搜索能力，避免算法陷入早熟收敛。在遗传过程中，两个子种群通过迁移将各自种群内的优良个体移入主种群中，这样具有平衡局部和全局的搜索能力。将不同的子种群替代原来单一种群，而各个子种群并行进化。主种群和两个子种群间相互独立，它们各自进化，遗传操作不受影响。主种群是随机初始化得到，而两个子种群的初始化为其中一个种群随机得到，另一个子种群是由这个子种群中每个的二进制编码取反得到。这样就达到了整个多种群模型的优秀个体共享和主种群多样性高的目的。

2.2 测试用例优先级动态调整计算方法

本文对多目标优化的TCP问题已有研究的基础上[11]，从测试需求与软件缺陷关联的角度，进一步结合智能搜索的多种群遗传算法，综合考虑覆盖率、测试用例设计信息和历史执行信息，研究多目标优化条件下的测试用例排序算法。为了更直观的描述TCP问题，现对涉及到的概念及相关变量作如下定义：

需求用例关联矩阵 $\boldsymbol{R}$：可以根据测试用例的相关信息得到测试需求与测试用例之间的关联矩阵 $\boldsymbol{R}=(\delta_{i,j})_{m\times n}$，矩阵元素 $\delta_{i,j}$ 在测试用例 t_j 满足覆盖测试需求 r_i 时值为1，否则值为0。

缺陷用例关联矩阵 $\boldsymbol{F}$：可以由测试用例的历史执行信息得到测试用例检测软件缺陷的情况，记为 $\boldsymbol{F}=(\theta_{i,j})_{k\times n}$，若测试用例 t_j 检测到缺陷 f_i，则在相应位置 $\theta_{i,j}$ 标为1，否则，标为0。

2.2.1 基于测试用例设计信息的优先级计算

测试用例覆盖需求的情况取决于测试用例相关设计信息，而测试需求的重要度对于待测软件而言就是某一功能的使用频率，在软件结构测试中可以体现为某一代码块。因此，对于实现不同软件功能的代码块对应不同测试需求，由测试人员根据整个系统各功能的具体情形，将测试需求的重要程度划分为：重要、较重要、一般、几乎不重要。为了方便计算，本文借鉴文献[11]的数值化处理方式，量化处理为1.0、0.7、0.4、0.1与之对应。这样，可得测试需求与测试用例关联的计算矩阵 $\boldsymbol{C}_{w_i}$。

$$\boldsymbol{C}_{w_i}=\begin{pmatrix} w_1\delta_{1,1} & \cdots & w_1\delta_{1,n} \\ \vdots & \ddots & \vdots \\ w_m\delta_{m,1} & \cdots & w_m\delta_{m,n} \end{pmatrix}_{m\times n} \tag{4}$$

其中：w_i 表示反映测试需求重要程度的权重系数。

为了更早达到测试覆盖率的要求，尽可能优先使用覆盖重要度较高的需求所对应的测试用例。实际上，测试需求的重要程度决定了测试用例的重要程度，因此将第 i 个测试用例所覆盖的各个可能引发软件失效(满足测试需求检测到软件缺陷)的平均值代表基于测试用例设计信息需求覆盖的优先级影响因子 P_{RT}^i，计算公式为：

$$P_{RT}^i=\frac{\sum_{j=1}^{m} w_j\delta_{j,i}}{\sum_{j=1}^{m}\delta_{j,i}} \tag{5}$$

2.2.2 基于测试用例历史执行信息的优先级计算

在软件开发迭代的过程中，必须对每个版本的软件测试评估。一般地，已完成回归测试的结果将影响后续回归测试的检测，历史检测信息对后期的测试具有一定参考价值。因此，在进行优先级排序时可以参考，测试用例历史缺陷覆盖的范围和检测缺陷的稳定程度。

历史缺陷检测能力(History fault proneness，HFP)是指如果一个测试用例在软件的最近一次版本更新中通过回归测试检测到软件的缺陷，那么软件的下一次或未来版本更新中会针对此缺陷引入开发人员的修复代码[12]。所以，在后续检测时提高该测试用例的优先级不仅能够及时排查对应的缺陷是否修复，还可以测试更新的代码是否植入新缺陷。

根据测试用例的历史检测信息，可以筛选出检测缺陷次数较多的测试用例，这些测试用例所检测的软件功能相对不够完善，因此需提高优先级进一步测试。这样，基于历史执行信息的第 i 个测试用例缺陷检测能力优先级影响因子 P_{HFP}^i 可以按式(6)计算：

$$P_{HFP}^i=\frac{H_{i,j}}{\sum_{i=1}^{n} H_{i,j}} \tag{6}$$

其中：n 表示全部测试用例的数量，$H_{i,j}$ 表示第 i 个测试用例在最近的 j 次回归测试中检测到缺陷数量。测试用例每一次的缺陷检测情况可由软件迭代更新的缺陷与测试用例关联矩阵 $\boldsymbol{F}$ 得到。

综合测试用例设计信息和历史执行信息的优先级计算因子，考虑测试用例在结构测试中的路径覆盖率(一般情况下，覆盖更多路径的测试用例检测出软件缺陷的可能性更大，排序的优先级更高)，可得到动态调整优先级的计算公式：

$$P_{t_i}=\alpha\cdot P^i_{RT}+\beta\cdot P^i_{HFP}+\gamma\cdot\frac{m}{n} \quad (7)$$

其中:m 表示测试用例 t_i 所覆盖目标路径的数目,n 表示结构测试中目标路径集总的路径数目。而 α、β、γ 分别为基于测试用例设计信息、测试用例历史执行信息和测试用例结构测试的路径覆盖率对应的测试用例优先级计算的权重系数,其权重系数值之和为 1。本文实验采取均衡策略,即$(\alpha,\beta,\gamma)=\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$。

2.3　算法步骤

本文所设计的测试用例优先级排序算法步骤描述如下:

a) 初始化三个种群,对遗传算子的相关参数进行设置;

b) 按式(6)计算每个种群中全部个体的适应度值并进行排序;

c) 将两个子种群的最优个体按一定的迁移间隔替换主种群的最差个体;

d) 对主种群和子种群分别进行遗传操作,得到下一代种群;

e) 检验主种群是否满足终止条件(检测出所有的缺陷或达到一定的 APFD 值),若满足则停止搜索,输出排序后的最优测试用例序列,否则返回步骤 b)。

3　实验分析

为了检验本文算法(MPGA)针对测试用例优先级排序问题的有效性,选取了一组来自 SIR 测试平台的西门子套件的评测程序作为实验基准[13],本文实验配置环境为,处理器英特尔第四代酷睿 i5-4690 @3.50 GHz,内存 16 GB,操作系统 Windows 7 64 位,在 Visual Stduio 2013 开发平台上采用 C 语言编程实现。其基本信息如表 1 所示。这些基准程序被广泛用于软件测试中对缺陷检测能力的研究。

表 1　评测程序的基本信息

被测程序编号	程序名称	函数个数	缺陷个数	代码行数	已有测试用例集个数最小值/最大值
PG1	tcas	8	5	138	5/12
PG2	schedule	17	8	374	11/16
PG3	replace	21	11	516	25/32
PG4	print_token	21	15	563	19/28

本文实验验证部分主要从以下两种情况去评价算法的性能。

情况 1:比较本文算法与单种群进化算法(SGA)求解 TCP 问题的缺陷检测能力。

情况 2:比较本文算法与采取单一目标的测试用例优先级排序策略的多目标缺陷检测速率。

针对情况 1,以 APFD 作为缺陷检测能力的评价指标设计了实验 1,实验中遗传进化的相关参数设置如下:交叉率 P_c 为 0.8,变异率 P_m 为 0.15(由于本文采取的是多种群并行进化,故控制为主种群交叉率和变异率保持与 SGA 相同,而另外两个子种群 $\Delta P_c=\pm 0.15$,$\Delta P_m=\pm 0.15$,迁移代数间隔为 5)。PG1 和 PG2 种群大小设为 30,PG3 和 PG4 种群大小设为 50。进化的终止准则为实现 100%的缺陷检测率,即检验出所有缺陷时的 APFD 值。另外,本文实验部分采用相同的初始测试用例集,由于选取的测试用例数目越多,其问题规模复杂度越高,更能检验算法的有效性,故选取不同被测程序已有测试用例集的最大用例集作为生成初始种群的待排序目标。而 MPGA 算法与 SGA 算法均采用随机化的初始化过程,值得说明的是,虽然 MPGA 算法与 SGA 算法随机化过程生成的父代种群可能不同,但个体变量的取值服从均匀分布,故二者的初始化条件本质上是相同的。为了尽可能避免随机因素的干扰,采取每组实验独立运行 50 次测试,其实验结果如图 2 所示。

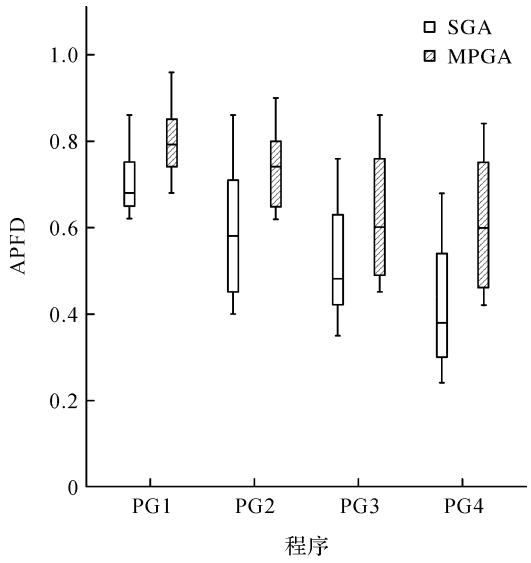

图 2　MPGA 与 SGA 算法针对不同评测程序的 APFD 值实验结果

由图2可以看出:MPGA与SGA算法针对缺陷数目不同、复杂度不同的评测程序进行测试用例优先级排序的缺陷检测能力不同,且随着问题规模增大,MPGA与SGA的APFD值均呈下降趋势;针对同一被测程序(PG1—PG4),本文算法(MPGA)较SGA具有更高的平均APFD值,例如以print_token(PG4)作为基准评测程序实验时,MPGA的APFD均值为0.606,而SGA的APFD均值仅为0.423。另外,就每组50次独立实验的统计结果来看,MPGA与SGA的测试用例优先级排序性能都具有一定波动性,例如以replace(PG3)作为基准评测程序实验时,SGA排序方法的APFD值在0.38到0.76之间,MPGA排序方法的APFD值在0.45到0.86之间,这是由遗传算法本身的随机性所决定的。

针对情况2,以检测到一定数量缺陷所耗费的测试用例数目作为缺陷检测速率的评价指标设计了实验2,一般而言,检测到单位缺陷所执行的测试用例数目越低,其缺陷检测速率越高。本文选取表1中缺陷数目最多的print_token(PG4)作为评测程序,相关参数设置同实验1,这里,测试用例优先级影响因子(α,β,γ)的实验取值参考文献[11]的研究工作,实际上这一部分本文主要探讨多目标的TCP排序方法与单一目标的缺陷检测速率性能,为了方便实验便于比较,本文多目标策略中(α,β,γ)按均衡策略取值,实验比较了本文方法同单一目标的TCP策略(即(α,β,γ)[11]分别取作(1,0,0)、(0,1,0)、(0,0,1)作为优先级计算的适应度函数)的缺陷检测情况,其结果如表2所示。

表2 不同方法的目标缺陷检测所耗费的测试用例实验结果

目标检测缺陷数	MPGA耗费的测试用例数目(α,β,γ)			
	单目标(1,0,0)	单目标(0,1,0)	单目标(0,0,1)	本文方法$\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$
5	6	4	4	4
10	12	15	16	8
15	20	22	15	11

由表2的实验结果可以看出:随着目标被测程序待检测缺陷数目的增加,采用多目标和单一目标的MPGA测试用例排序策略进行缺陷检测所耗费的测试用例数目都相应增加;综合考虑覆盖率、测试用例设计信息和历史执行信息的多目标优先级调整策略相对于单目标排序策略而言,其耗费的测试用例数目是更少的,且随着目标缺陷数提高,优势越明显。就比较的三个单目标排序方法而言,基于测试用例设计信息需求覆盖的单目标排序方法(即(α,β,γ)取(1,0,0))与基于测试用例历史执行信息的单目标排序方法(即(α,β,γ)取(0,1,0))相比,在目标检测缺陷数较多时所耗费的测试用例数目更少;另外,相较于传统方法中采用路径覆盖率作为评价测试用例检测缺陷能力的单目标排序策略(即(α,β,γ)取(0,0,1)),MPGA耗费的测试用例数目因目标检测缺陷数而异。总体来说,三种单目标的测试用例排序策略都无法在以不同缺陷数作为检测目标的测试用例耗费方面达到恒优,而采用多目标的MPGA排序策略有助于缩减测试用例耗费数目,降低测试代价。

综上可知,本文提出的多目标MPGA测试用例排序策略相较于传统单目标的测试用例排序策略的缺陷检测能力更具优势,其检测目标缺陷数所耗费的测试用例数目更低;与采用单种群的方法SGA求解测试用例优先级问题相比,本文方法的缺陷检测速率更高。

4 结 语

TCP技术是当前软件测试的研究热点。本文针对白盒测试中的TCP问题,基于多种群的进化模型,综合考虑覆盖率、测试用例设计信息和历史执行信息三方面的因素,提出了一种多目标的MPGA排序策略。实验验证了本文方法较单一目标及单种群的排序方法在缺陷检测能力上更具优势。由于本文方法主要探讨多目标排序方法和单目标排序方法的优先级调整策略,对遗传算法本身的一些参数选取上并没有作过多的优化研究,实际上,由于遗传算法本身具有一定随机性,排序策略的效果可以通过参数优化进一步提高算法的稳定性。另外,本文考虑的多目标排序策略的影响因子是均衡的,未来还将考虑非均衡的带权重的多目标测试用例排序策略,进一步优化缺陷检测能力。

参考文献:

[1] 李龙澍,李森,廖敏,等.基于多种群遗传算法测试用例优先级技术研究[J].计算机技术与发展,2011,21(4):112-114.

[2] Wong W E, Horgan J R, London S, et al. A study of effective regression testing in practice[C]//The Eighth International Symposium on Software Reliability Engineering. IEEE,1997:264-274.

[3] Srikanth H, Williams L. On the economics of requirements-based test case prioritization[J]. Acm Sigsoft Software

Engineering Notes,2005,30(4):1-3.

[4] Kavitha R, Kavitha V R, Kumar N S. Requirement based test case prioritization[C]//IEEE International Conference on Communication Control and Computing Technologies. IEEE,2010:826-829.

[5] Jasz J, Lango L, Gyimothy T, et al. Code coverage-based regression test selection and prioritization in WebKit [C]//IEEE International Conference on Software Maintenance. IEEE,2012:46-55.

[6] Rothermel G, Untch R H, Chu C, et al. Prioritizing test cases for regression testing[J]. Software Engineering IEEE Transactions on,2000,27(10):929-948.

[7] Jeffrey D, Gupta R. Test case prioritization using relevant slices[C]//International Computer Software and Applications Conference. IEEE,2006:411-420.

[8] Zheng L, Harman M, HIERONS R M. Search algorithms for regression test case prioritization[J]. IEEE Transactions on Software Engineering,2007,33(4):225-237.

[9] Kaur A, Bhatt D. Particle swarm optimization with cross-over operator for prioritization in regression testing [J]. International Journal of Computer Applications, 2011,27(10):527-527.

[10] Li S, Bian N, Chen Z, et al. A simulation study on some search algorithms for regression test case prioritization [C]//International Conference on Quality Software. IEEE,2010:72-81.

[11] 张娜,姚澜,包晓安,等.多目标优化的测试用例优先级在线调整策略[J].软件学报,2015,26(10):2451-2464.

[12] 包晓安,谢晓鸣,张娜,等.基于缺陷关联度的 Markov 模型软件优化测试策略[J].软件学报,2015,26(1):14-25.

[13] Do H, Elbaum S, Rorhermel G. Supporting controlled experimentation with testing techniques: an infrastructure and its potential impact[J]. Empirical Software Engineering, 2005,10(4):405-435.

Research on test cases prioritization based on multi-population evolutionary algorithm

ZHANG Na, *HU Guoheng*, *JIN Yuting*, *SHI Jiabing*, *BAO Xiaoan*

(School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract: In order to improve the software test efficiency and accelerate the progress of software development, a method based on the multi-population evolutionary algorithm was proposed in this paper to solve the sorting problem of test cases. For the premature convergence problem in single population genetic algorithm, a multi-population parallel evolution model was first introduced to enhance global optimization capacity of the algorithm. Based on this model, this paper proposed a dynamic adjustment calculation method of test cases priority by combining software requirement coverage with software defect detection rate and comprehensively considering code coverage, test case design information, and historical execution information. The experiment results show that compared with the traditional priority sorting algorithm oriented to single target coverage, the test speed and software defect detection ability of this algorithm are improved.

Key words: regression test; test cases priority; multi-population; dynamic adjustment

(责任编辑:康　锋)