计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191100113

基于多目标优化的测试用例优先级排序方法

夏春艳[1]　王兴亚[2]　张　岩[1]
1 牡丹江师范学院计算机与信息技术学院　黑龙江 牡丹江 157011
2 南京大学计算机软件新技术国家重点实验室　南京 210023
(xia-chun-yan@163.com)

摘　要　回归测试是软件测试中使用最频繁、成本最昂贵的测试方法。测试用例优先级排序是一种能够有效降低回归测试成本的方法,其目的是通过优先执行高级别的测试用例来达到提升软件故障检测的能力。文中提出了一种基于多目标优化的测试用例优先级排序方法,该方法在遗传算法的个体评价机制中融入了选择函数,设计了合理的编码方式以及合适的选择、交叉和变异策略,以故障检测率、语句覆盖率和有效执行时间为优化目标,采用非支配排序遗传算法对测试用例优先级排序。基于4个基准程序和4个工业程序的实验结果表明:与其他方法相比,所提方法能够提高软件测试的有效性。

关键词:软件测试;测试用例优先级;多目标优化;非支配排序遗传算法;选择函数
中图法分类号　TP311

Test Case Prioritization Based on Multi-objective Optimization

XIA Chun-yan[1], WANG Xing-ya[2] and ZHANG Yan[1]
1 School of Computer and Information Technology, Mudanjiang Normal University, Mudanjiang, Heilongjiang 157011, China
2 State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China

Abstract　Regression testing is the most frequently used and expensive testing method in software testing. Test case prioritization is an effective way to reduce the cost of regression testing. Its purpose is to improve the ability of software fault detection by prioritizing the execution of high-level test cases. In this paper, a method of test case prioritization based on multi-objective optimization is proposed. The method integrates choice function into individual evaluation mechanisms of genetic algorithm. By designing a reasonable coding method and appropriate selection, crossover and mutation strategies, taking fault detection rate, sentence coverage rate and effective execution time as optimization objectives, non-dominated sorting genetic algorithm is used to optimize test case sort. The experimental results based on four benchmark programs and four industrial programs show that the proposed method can improve the effectiveness of software testing compared with other methods.

Keywords　Software testing, Test case priority, Multi-objective optimization, Non-dominated sorting genetic algorithm, Choice function

1　引言

软件测试是软件生命周期的重要组成部分,是保证软件质量、提高软件可靠性的重要手段,其主要目标是确保可交付的软件没有缺陷[1]。研究资料表明,软件测试过程大约占软件开发总成本的一半以上[2]。在软件开发完成后,由于客户需求等发生改变,还需要对软件执行修补等操作。为确保软件维护工作不损害应用程序的预期性能,软件工程师会对维护后的软件进行重新测试,以提高软件的可靠性。回归测试的任务就是对修改后的软件进行重新测试,主要目的是检测应用程序发生改变后是否引入了新的错误。已有研究显示,回归测试占软件测试成本的比例高达80%以上[3]。因此,研究合适的回归测试方法,对降低软件测试成本是非常有意义的。

由于开发人员对软件不断地进行修改,导致测试用例集变得越来越庞大和复杂,因此,回归测试面临的主要问题就是如何高效地利用已有的测试用例对修改后的软件进行测试?针对这一问题,研究人员提出了一系列测试用例优化技术,包括测试用例集约简(Test Suite Minisation,TSM)、测试用例选择(Test Case Selection,TCS)和测试用例优先级排序(Test Case Prioritization,TCP)。TSM和TCS是指通过相关准则放弃冗余测试用例,从而达到提高回归测试效率的目的,但在一定程度上存在遗漏重要测试用例而导致测试用例集检错能

到稿日期:2019-11-15　返修日期:2020-05-09　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:黑龙江省教育厅基本科研业务费(1353MSYYB005);牡丹江师范学院科学技术研究(2018-KYYWF-0419)
This work was supported by Science Research Project of Heilongjiang Provincial Education Department (1353MSYYB005) and Research Project of Mudanjiang Normal University (2018-KYYWF-0419).
通信作者:王兴亚(xingyawang@nju.edu.com)

力降低的风险。TCP是在不减少测试用例数量的情况下，按照既定的测试目标设置测试用例的优先级别，使得具有较高优先级的测试用例优先执行，目的是尽早地检测到故障，以加快故障定位，从而降低回归测试的成本。

目前，已有很多学者对TCP技术进行了研究，Wong等[4]最早在回归测试中运用优先级技术，以语句覆盖率为目标设置测试用例的优先级；Kavitha等[5]和Nayak等[6]提出了基于检错率的测试用例排序方法；Walcott等[7]提出了基于测试用例执行时间的优先级方法；Srikanth等[8]研究了依据测试用例对需求的满足情况赋予测试用例优先级的方法。上述方法在一定程度上提高了回归测试的效率，但其研究内容都是以单目标优化为主。随着软件程序的日趋复杂，单目标的TCP技术已经难以满足回归测试的需求，多目标测试用例优先级排序已经成为当前回归测试中亟待解决的一个重要问题[9]。

多目标测试用例优先级排序(Multi-Objective Test Case Prioritization，MOTCP)是指按照多个测试目标找到一个符合测试需求的测试用例序列，提高测试套件在软件测试过程中的故障检测率，从而达到成本与效益需求相平衡的目的，基已被证明是一个NP-hard问题[10]。已有的MOTCP研究主要集中在加权法，但该方法中各优化目标的权值分配受人的主观影响较大。Mahmood等[11]通过考虑错误检测率、需求覆盖率、软件复杂性和历史信息等16个优化目标对测试用例排序，虽然此方法考虑了影响回归测试的众多因素，但对于每个目标分配相等的权值，并不一定适合回归测试的实际需求。目前，对于MOTCP较为有效的方法是元启发式搜索算法，但基于帕累托(Pareto)最优的智能搜索算法在此问题上的研究较少，多数研究又局限于两个优化目标，而在实际的软件测试过程中，通常需要同时满足多个测试准则，也就需要考虑多个优化目标，因此这是值得关注的问题[12]。MOTCP技术需要解决的另一个较为重要的问题是多目标的选择，Rajendrani等[13]对2001—2018年的90篇关于TCP的学术论文进行了研究，结果显示错误检测率(Average Percentage of Fault Detection，APFD)、语句覆盖率(Average Percentage of Statement Coverage，APSC)和有效执行时间(Effective Execute Time，EET)是处理TCP问题最有效的3项度量指标。

综上，本文提出一种基于多目标优化的测试用例优先级排序方法，该方法以基于Pareto最优的非支配排序遗传算法(Non-dominated Sorting Genetic Algorithms - II，NSGA-II)为基础，引入选择函数评价个体，以*APFD*，*APSC*和*EET*为优化目标对测试用例优先级进行排序。实验表明，所提方法能够提高被测软件的故障检测率和回归测试的效率，进而达到降低回归测试成本和提高软件可靠性的目的。

本文第2节介绍了与研究内容相关的基本概念；第3节详细阐述了所提方法；第4节通过实验验证了所提方法的有效性；最后总结全文，提出进一步的研究方向。

2 基本概念

2.1 选择函数

选择函数(Choice Function，CF)是采用自适应方式来评定个体质量的一种排序方法。最初是由Cowling等[14]和Kendall等[15]基于信用得分提出的个体评价函数，其表达式为：

$$\phi(x_i)=\alpha\cdot\phi_1(x_i)+\beta\cdot\phi_2(x_i,x_j)+\gamma\cdot\phi_3(x_i) \tag{1}$$

其中，x_i是被评价的个体；$\phi(x_i)$是x_i的信用得分；$\phi_1(x_i)$是x_i的信用改进值；$\phi_2(x_i,x_j)$是在执行x_j后，x_i的信用改进值；$\phi_3(x_i)$是x_i的执行时间。α，β和γ代表函数的权重值。

式(1)中，ϕ_1和ϕ_2用于强化搜索目的，即在当前搜索空间中，倾向有利于产生最佳结果的个体；与此相反，ϕ_3用于解决方案的多样化，即通过支持一定时间内没有使用的个体来更好地探索搜索空间。

Maashi等[16]使用了CF的简化版本，只包含两个函数ϕ_1和ϕ_2，其公式如下：

$$\phi(x_i)=\alpha\cdot\phi_1(x_i)+\beta\cdot\phi_2(x_i) \tag{2}$$

其中，$\phi(x_i)$表示x_i的信用得分；ϕ_1反映x_i的改进得分；ϕ_2反映x_i的执行时间；α和β代表函数的权重值。函数和其权重主要用于强化搜索目的和平衡搜索过程中的多样性，如果需要强化搜索目的，则增加α值；如果需要搜索空间具有多样性，则增加β值。

2.2 多目标优化

在多数情况下，多目标优化问题通常不是一个单独的子目标优化，而是一组相互冲突的子目标优化。参考Zheng[17]对多目标优化的相关定义，给出如下说明。

多目标优化问题的一般描述为：给定决策变量$x=(x_1,x_2,\cdots,x_n)$，满足下列两个约束：

(1)$g_i(x)\geqslant 0(i=1,2,\cdots,k)$；

(2)$h_i(x)\geqslant 0(i=1,2,\cdots,l)$。

设有m个相互冲突的子目标，则优化目标可表示为：

$$f(x)=(f_1(x),f_2(x),\cdots,f_m(x)) \tag{3}$$

寻求$x^*=(x_1^*,x_2^*,\cdots,x_n^*)$，使$f(x^*)$在满足两个约束条件的同时达到最优。

将各个子目标优化函数统一转化为求优化目标的最大化，即：

$$\begin{aligned}&\max\ f(x)=(f_1(x),f_2(x),\cdots,f_m(x))\\&\text{s.t.}\ \ x\in\Omega\end{aligned} \tag{4}$$

其中，Ω为满足多目标优化中两个约束的可行解集，称为决策变量空间(简称决策空间)，即$\Omega=\{x\in R^n\mid g_i(x)\geqslant 0,h_j(x)\geqslant 0;(i=1,2,\cdots,k;j=1,2,\cdots,l)\}$。

通常情况下，多目标优化中的最优解称为Pareto最优解。Pareto最优解的一般描述为：给定一个多目标优化问题$\max f(x)$，若$x^*\in\Omega$，且不存在其他的$x\in\Omega$，使得$f_i(x^*)\leqslant f_i(x)(i=1,2,\cdots,m)$成立，且其中至少一个是严格不等式，则称$x^*$是$\max f(x)$的Pareto最优解。

多数情况下，Pareto最优解不止一个，而是一个Pareto最优解集(Pareto optimal set)。Pareto最优解集的一般描述为：给定一个多目标优化问题$\max f(x)$，其最优解集为$P^*=\{x^*\in\Omega\mid\exists x\in\Omega,f_i(x^*)\leqslant f_i(x)(i=1,2,\cdots,m)\}$。

多目标进化算法的优化过程是针对每一代进化群体寻找当前最优解，称当前进化群体的最优解为非支配解(non-dominated solution)，所有非支配解的集合称为当前进化群体的非支配集(non-dominated set)，优化的目的是使非支配集不断逼近真正的最优解。

2.3 测试用例优先级排序

Rothermel 等[18]首先给出了 TCP 问题的正式定义：给定的测试用例集为 T，T 中测试用例所有可能的执行顺序为 PT，PT 到实数集的映射函数为 $f:PT\rightarrow R$，测试用例优先级的研究目标就是找到其中的一个排序 $T'\in PT$，使得对于任意的 $T''\in PT$ 和 $T''\neq T'$，都有 $f(T')>f(T'')$。其中，f 是对排序目标的定量描述，用来度量排序的有效性，f 值越大，测试用例的排序效果就越好。

MOTCP 是在 TCP 的基础上，根据多个目标对测试用例执行序列进行优化，其形式化定义如下：给定的测试用例集为 T，T 中测试用例的全排列集合为 PT，目标函数向量为 $\vec{\boldsymbol{F}}=[f_1(x),f_2(x),\cdots,f_i(x),\cdots,f_m(x)]$，$x\in PT$，$1\leqslant i\leqslant m$，$f_i$ 表示第 i 个优化目标的目标函数，$f_i:PT\rightarrow R$。MOTCP 的目的是找到一个 $PT'\subset PT$，使得 $\forall x'\in PT'$，并且 $\vec{\boldsymbol{F}}(x')$ 达到 Pareto 最优。其中，目标函数向量 $\vec{\boldsymbol{F}}$ 是对测试用例优化目标的数学抽象表示，分量 f_i 是第 i 个优化目标的数学抽象，它表示一个从测试用例全排列空间到实数空间的映射。

基于 Pareto 最优的 TCP 技术的含义是测试用例集的任意两个排序方案 $x_1,x_2\in PT$，$\vec{\boldsymbol{F}}_1$ 和 $\vec{\boldsymbol{F}}_2$ 分别是 x_1 和 x_2 的目标函数向量，若存在至少一个 $f_i(x_1)$ 不如 $f_i(x_2)$，且同时存在至少一个 $f_j(x_2)$ 不如 $f_j(x_1)$，其中 $1\leqslant i,j\leqslant m$，则说明 x_1 和 x_2 互为非支配解。多目标测试用例优化的目的就是在测试用例的全排列集合中，寻找最优非支配解集，也就是 Pareto 最优解集。

3 进化方法

本文提出一种融合选择函数的非支配遗传算法(Choice Function_Non-dominated Sorting Genetic Algorithms - II，CF_NSGA-II)，对测试用例的优先级进行排序，主要涉及编码及算子的设计、个体评价函数的计算和优化目标的选择 3 部分。第 1 部分的目标在于合理地设计进化方法的编码方式，以及选择、交叉和变异策略；第 2 部分的目标在于有效融合选择函数，给出遗传算法的个体评价机制；第 3 部分的目标在于选择合适的优化目标，采用 CF_NSGA-II 实现测试用例的优先级排序。多目标进化方法的流程如图 1 所示。

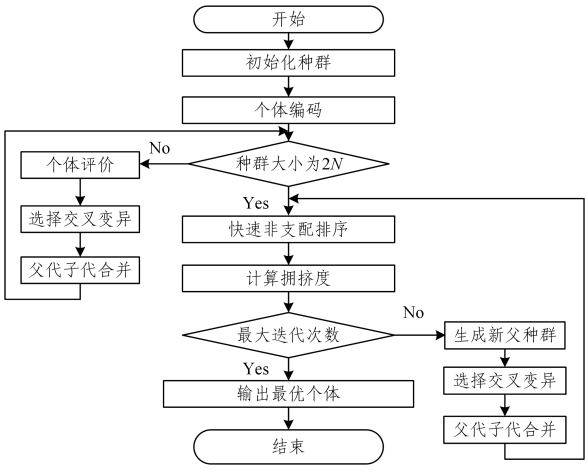

图 1 多目标进化方法的流程图

Fig. 1 Flow chart of multi-objective evolutionary method

3.1 编码及算子

在多目标进化方法中，种群个体代表的是测试用例集的排序序列，生成的最优解为基于多个优化目标对测试用例优先级排序后的序列，个体编码及相关算子的设计方法如下。

(1)编码策略。种群个体采用的是十进制编码方式，基本基因位代表相应的测试用例序号。因此，种群中的每个个体对应的都是一组测试用例序列。例如，个体 x_1 和 x_2 分别含有 9 个测试用例，编码方式如图 2 所示。

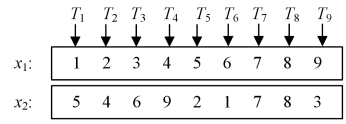

图 2 个体的编码方式

Fig. 2 Individual coding mode

(2)选择算子。选择操作采用的是锦标赛选择方法，即随机选择一定数量的个体，从中选择最好的一个作为父代个体。

(3)交叉算子。交叉操作采用的是部分映射交叉方法，即随机选择基因段，将其交换，并对其余基因做冲突检测，按照映射进行匹配。以图 2 所示的一对染色体为父代，则交叉过程如图 3 所示，具体过程如下：首先，随机选择几个基因的起止位置；其次，交换两组相关基因的位置；然后，对剩余基因做冲突检测，根据交换的两组基因建立一个映射关系；最后，形成新的一对子代基因。

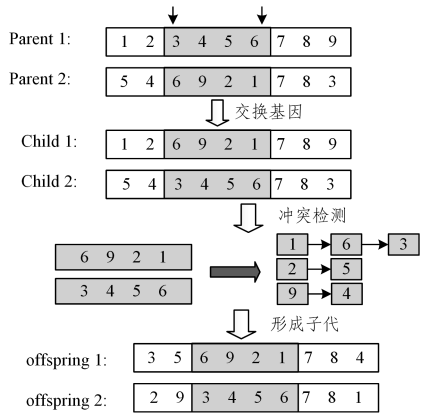

图 3 部分映射交叉过程

Fig. 3 Partial mapping cross process

(4)变异算子。变异操作采用的是序号变异方法，即随机选择两点，将两点的值进行交换。

3.2 个体评价

本文以 CF 简化版的形式为依据，提出新的个体评估机制。不失一般性，在遗传算法的种群中，两个个体的相似度越大，个体的相关度就越大；两个个体的相异度越大，个体的相关度就越小。因此，在引入选择函数对遗传算法进行个体评价时，ϕ_1 和 ϕ_2 是基于个体的相似度和相异度进行计算的，下面给出 ϕ_1 和 ϕ_2 的计算方法。

(1)ϕ_1 的计算方法

在遗传算法中，个体的相似度越大，其相关度就越大。因此，依据个体的相似度定义个体评价函数中的 ϕ_1，能够强化搜索目的并提高收敛速度，其表达式为：

$$\phi_1(x_i)=-\frac{1}{n-1}\sum_{j=1,j\neq i}^{n}S(x_i,x_j) \tag{5}$$

其中，$S(x_i,x_j)$表示 x_i和 x_j的相似度，其计算方法为 x_i和 x_j相同编码的长度与个体编码总长度 n 的比值，能够反映两个个体的相似程度，其表达式为：

$$S(x_i,x_j)=\frac{1}{n}\sum_{k=1}^{n}s_k(x_i,x_j) \tag{6}$$

其中，

$$s_k(x_i,x_j)=\begin{cases}1, & \text{前 } k \text{ 个节点真值相同}\\ 0, & \text{否则}\end{cases}$$

(2)ϕ_2的计算方法

在遗传算法中，个体的相异度越大，其相关度就越小。因此，依据个体的相异度定义个体评价函数中的ϕ_2，能够增强搜索空间的多样性，避免早熟现象，其表达式为：

$$\phi_2(x_i)=\frac{1}{n-1}\sum_{j=1,j\neq i}^{n}D(x_i,x_j) \tag{7}$$

其中，$D(x_i,x_j)$表示 x_i和 x_j的相异度，其计算方法为 x_i和 x_j不同编码的个数与个体编码总长度 n 的比值，能够反映两个个体的相异程度，其表达式为：

$$D(x_i,x_j)=\frac{1}{n}\sum_{k=1}^{n}d_k(x_i,x_j) \tag{8}$$

其中，

$$d_k(x_i,x_j)=\begin{cases}0, & \text{第 } k \text{ 个节点真值相同}\\ 1, & \text{否则}\end{cases}$$

3.3 优化目标

MOTCP 的目的是在多个目标冲突且不存在单一最优解的情况下，找到 TCP 的折中方案。这里选择 *APFD*，*APSC* 和 *EET* 3 项最常用的度量指标作为 TCP 的优化目标。

(1)平均故障检测率，主要用于计算测试用例执行过程中检测到缺陷的平均累计比例，其计算公式如下：

$$APFD=1-\frac{TF_1+TF_2+\cdots+TF_m}{mn}+\frac{1}{2n} \tag{9}$$

其中，n 是测试用例集 T 中的测试用例数量，m 是被测软件缺陷总数量，TF_i表示排序后测试用例集 T'中首次检测到缺陷 i 的测试用例在该执行次序中所处的次序。*APFD* 的值域为[0,1]，排序在前的测试用例检测出的缺陷越多，则 *APFD* 越高，缺陷检测速度越快，测试集的排序效果就越好。

(2)语句覆盖率，主要用于计算在测试用例执行时，所用到的可执行语句数与全部可执行语句数的比例，其计算公式如下：

$$APSC=1-\frac{TS_1+TS_2+\cdots+TS_m}{mn}+\frac{1}{2n} \tag{10}$$

其中，n 是测试用例集 T 中测试用例的数量，m 是被测软件语句总数量，TS_i表示排序后测试用例集 T'中首次覆盖第 i 行代码的测试用例在该执行次序中所处的次序。

(3)有效执行时间，主要用于计算测试用例的故障检测率和语句覆盖率达到最大时的执行总时间，其计算公式如下：

$$EET=\sum_{i=1}^{n}ET_i \tag{11}$$

其中，ET_i表示执行第 i 个测试用例时所消耗的时间。

4 实验

本文提出了一种基于多目标的测试用例优化方法，用于对生成的测试用例进行优先级排序。为了评估所提方法的性能，本文以 8 个不同规模的评测程序为实验对象开展实证研究，并且与其他方法的实验结果进行对比，验证所提方法的有效性。对比方法选择 NAGA-II 方法，原因是 CF_NAGA-II 方法是在 NAGA-II 方法中融入了选择函数作为个体评价机制，因此这两种方法更具有可比性。实验环境如下：Windows 7 操作系统，计算机主频 2.80 GHz，内存 4 GB，所有程序均用 C++语言编写，在 Visual C++ 6.0 环境下运行。

4.1 研究问题

为了对所提方法的有效性进行评价，以 APFD，APSC 和 EET 为优化目标，比较 CF_NAGA-II 和 NAGA-II 两种方法的实验结果，并回答如下两个问题。

问题 1：相比 NAGA-II，CF_NAGA-II 能否提高测试用例集的故障检测率？

问题 2：相比 NAGA-II，CF_NAGA-II 能否提高测试用例优先级排序的有效性？

4.2 实验对象

本文选择 8 个采用 C 语言编程实现的具有典型性的被测程序进行实验，并对实验结果进行分析。其中包含 4 个小规模的基准程序和 4 个来自西门子套件的工业程序，这些程序常被用于软件测试研究领域，也是研究人员常用的评测程序[19-20]。实验程序的基本信息如表 1 所列，对于每个被测程序，表 1 分别列出了程序名称、程序简要描述、代码行数、测试用例数和缺陷数。

表 1 实验程序的基本信息

Table 1 Basic information of experimental procedure

Name	Description	LOC	# Test	# Fault
bubble	Bubble Sort	29	96	4
median	Find median	37	125	5
triangle	Triangle types	42	216	7
nextday	calculate data	83	377	13
tcas	Altitude separation	137	1052	41
tot_info	Information measure	281	1608	23
schedule	Priority scheduler	296	2650	7
print_tokens	Lexical analyzer	343	4130	9

4.3 评价指标

在大多数研究中，MOTCP 的评价指标为 *APFD*，这也是验证 4.1 节提出的问题 1 的评价指标。除此之外，为了评价所提方法的有效性，即问题 2，参照基于语句排序的评测指标[21]，从百分比的角度出发，基于被测程序的测试用例在优先级排序序列中的位置，给出两种互补的评测指标 *Score* 和 *Expense*。

(1)*Score* 评测指标。当被测程序执行排序后的某一个测试用例时，语句覆盖率达到最大值，则记录当前测试用例在排序序列中的位置，并且返回不需要审查的测试用例数占所有测试用例总数的百分比，其计算公式为：

$$Score=\left(1-\frac{Rank(x_i)}{sum(x_i)}\right)\times 100\% \tag{12}$$

(2)*Expense* 评测指标。与上述情况相同，但返回的是必

须审查的测试用例数占所有测试用例总数的百分比，其计算公式为：

$$Expense=\frac{Rank(x_i)}{sum(x_i)}\times 100\% \tag{13}$$

其中，$Rank(x_i)$表示测试用例 x_i 在优先级排序序列中的位置，$sum(x_i)$表示所有测试用例的总数。

4.4 实验设计

为有效开展对比实验，CF_NSGA-II 与 NSGA-II 设置相同的实验参数：交叉概率为 0.8，变异概率为 0.15，最大迭代次数为 200。经过反复实验得出，CF_NSGA-II 方法中 CF 函数的权重值 α 和 β 均设置为 0.5 较为合适，原因是搜索的目的和搜索空间的多样性可以得到更好地强化和平衡。为了保证实验结果的稳定性，每一被测程序运行 30 次，取实验数据的平均值作为实验结果。需要说明的是，将测试用例排序作为多目标优化问题，在其 Pareto 最优解集中可能存在多个解，这时采用随机选择的方法进行筛选。下面对故障检测率进行评价，并比较 CF_NSGA-II 和 NSGA-II 方法的评测指标 *Score* 和 *Expense*，进而验证 CF_NSGA-II 方法的有效性。

4.5 实验结果与分析

图 4 显示了测试用例集执行的百分比与对应的错误检测能力。图 4 中，横坐标表示测试用例集执行的百分比，纵坐标表示故障检测率，蓝色区域为 CF_NAGA-II 的实验结果，黄色区域为 NSG-II 的实验结果。

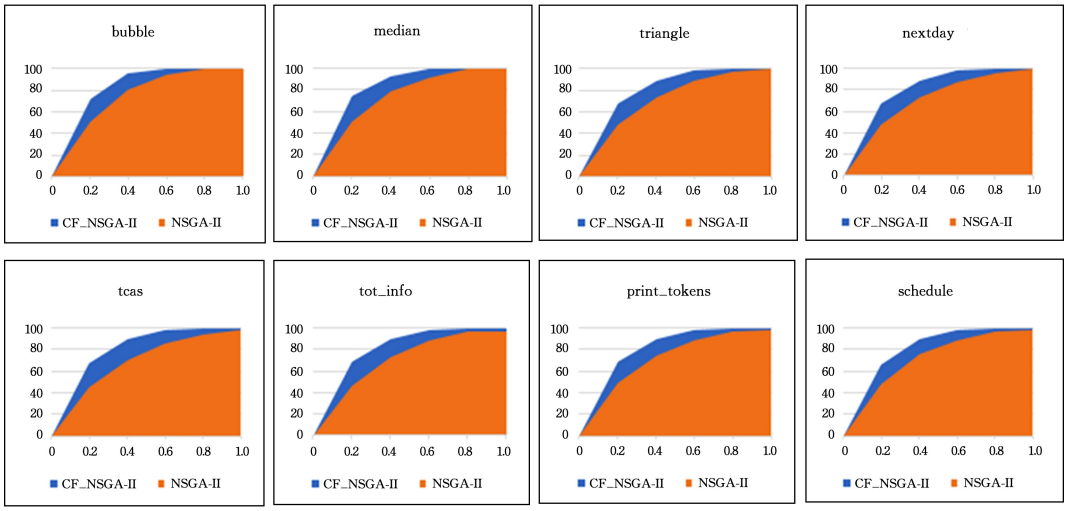

图 4 实验结果对比图(电子版为彩色)

Fig. 4 Comparison of experimental results

实验结果显示，在执行同样比例的测试用例时，与 NSG-II 方法相比，CF_NAGA-II 方法具有较好的错误检测能力，即提出的基于多目标优化的测试用例优先级排序方法，能够以测试用例集中优先执行的较少的测试用例检测到较多的软件故障，提高了软件的故障检测率，达到了提高软件可靠性的目的，验证了问题 1。为了更好地说明所提方法的有效性，给出 NSG-II 和 CF_NAGA-II 两种方法的评测指标 *Score* 和 *Expense*，如图 5 所示。图 5 中，横坐标为被测程序的名称，纵坐标为评测指标的值，蓝色表示 CF_NAGA-II 的实验结果，黄色表示 NSG-II 的实验结果。*Score* 和 *Expense* 为语句覆盖率达到最大值时，对应的不需要执行的测试用例和必须执行的测试用例在测试用例集中的位置比率。*Score* 值越高，对应的测试用例集的优先级排序就越好；相反，*Expense* 值越高，对应的测试用例集的优先级排序就越差。由 *Score* 和 *Expense* 的对比结果可知，与 NSG-II 相比，CF_NAGA-II 的测试用例优先级排序的有效性更好，也就是说本文方法能够提高软件测试的有效性，达到降低回归测试成本的目的，验证了问题 2。

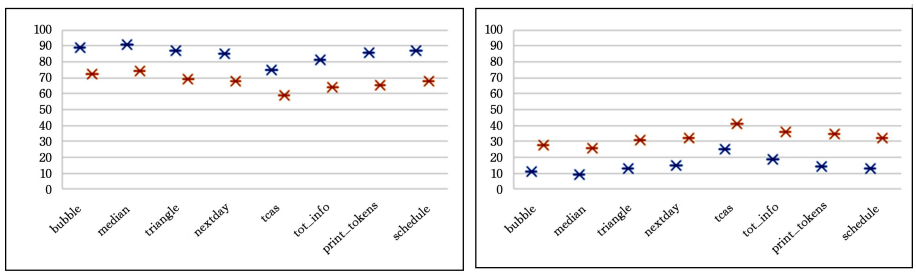

图 5 *Score* 和 *Expense* 的对比(电子版为彩色)

Fig. 5 Comparison of *Score* and *Expense*

结束语 为了解决单目标优化确定因素过于单一的问题，以及多目标优化加权法中设置的权值受人为主观因素影响的问题，本文提出一种基于多目标优化的测试用例优先级排序方法，将选择函数融入遗传算法的个体评价机制，以提高遗传算法种群数据的收敛性和多样性，并以故障检测率、语句覆盖率和有效执行时间为优化目标，采用非支配排序遗传算

法实现了测试用例优先级的排序。实验结果表明,该方法能够提高软件的故障检测率和回归测试的效率,进而降低软件的测试成本。后续工作将执行更复杂的并行程序来衡量本文方法的有效性,以更好地提高软件测试的效率。

参 考 文 献

[1] TIAN T,GONG D W. Evolutionary generation approach of test data for multiple paths coverage of message-passing parallel programs[J]. Chinese Journal of Electronics,2014,23(2):291-296.

[2] SAHIN O,AKAY B. Comparisons of metaheuristic algorithms and fitness functions on software test data generation[J]. Applied Soft Computing,2016,49:1202-1214.

[3] PRADHAN D,WANG S,ALI S,et al. Employing rule mining and multi-objective search for dynamic test case prioritization [J]. The Journal of Systems and Software,2019,153:86-104.

[4] WONG W,HORGAN J,LONDON S,et al. A study of effective regression testing in practice[C]// Proceedings of the Eighth International Symposium on Software Reliability Engineering. New Mexico,USA,1997,11:264-274.

[5] KAVITHA R,SURESHKUMAR N. Test case prioritization for regression testing based on severity of fault[J]. International Journal on Computer Science & Engineering,2010,2(5):1462-1466.

[6] NAYAK S,KUMAR C,TRIPATHI S. Enhancing efficiency of the test case Prioritization technique by improving the rate of fault detection[J]. Arabian Journal for Science & Engineering,2017,42(11):1-17.

[7] WALCOTT K R,SOFFA M L,KAPFHAMMER G M,et al. Time-Aware test suite prioritization[C]// Pollock L,ed. Proc. of the Int'l Symp. on Software Testing and Analysis. Portland:ACM Press,2006:1-12.

[8] SRIKANTH H,WILLIAMS L,OSBORNE J. System test case prioritization of new and regression test cases[C]// Proceedings of the International Symposium on Empirical Software Engineering. Noosa Heads,Australia,2005:64-73.

[9] CHEN X,CHEN J H,JU X L,et al. Survey of test case prioritization techniques for regression testing[J]. Journal of Software,2013,24(8):1695-1712.

[10] MICHAEL R G,DAVID S J. Computers and intractability:a guide to the theory of NP-completeness[M]. WH Free. Co,San Fr,1979.

[11] MAHMOOD H,HOSAIN S. Improving test case prioritization based on practical priority factors[C]// Proceedings of IEEE International Conference on Software Engineering and Service Science. Beijing:IEEE Press,2017:899-902.

[12] CHEN Y F,LI Z,ZHAO R L. Applying PSO to multi-objective test cases prioritization[J]. Computer Science,2014,41(5):72-77.

[13] MUKHERJEE R,PATNAIK K S. A survey on different approaches for software test case prioritization[J]. Journal of King Saud University - Computer and Information Sciences,2018:1319-1578.

[14] COWLING P,KENDALL G,SOUBEIGA E. A hyperheuristic approach to scheduling a sales summit[C]// Practice and Theory of Automated Timetabling. 2001:176-190.

[15] KENDALL G,SOUBEIGA E,COWLING P. Choice function and random hyperheuristics[C]// Asia-Pacific Conference on Simulated Evolution and Learning. Springer,2002:667-671.

[16] MAASHI M,ÖZCAN E,KENDALL G. A multi-objective hyper-heuristic based on choice function[J]. Expert Syst. Appl.,2014,41(9):4475-4493.

[17] 郑金华. 多目标进化算法及其应用[M]. 北京:科学出版社,2007:2-8.

[18] ROTHEMEL G,UNTCH R H,CHU C,et al. Prioritiz-ing test cases for regression testing[J]. IEEE Transactions on Software Engineering,2001,27(10):929-948.

[19] SUN C A,GUO X L,ZHANG X Y,et al. A Data Flow Analysis Based Redundant Mutants Identification Technique[J]. Chinese Journal of Computers,2019,42(1):44-60.

[20] ZHENG Y,WANG Z,FANG X,et al. Localizing multiple software faults based on evolution algorithm[J]. The Journal of Systems and Software,2018,139:107-123.

[21] YOO S,HARMAN M,CLARK D. Fault localization prioritization:comparing information theoretic and coverage based approaches[J]. ACM Transactions on Software Engineering and Methodology,2013,22(3):1049-1078.

XIA Chun-yan, born in 1980, postgraduate, associate professor, is a member of China Computer Federation. Her main research interests include search-based software engineering, information processing and data mining.

WANG Xing-ya, born in 1990, Ph.D, associate professor, is a member of China Computer Federation. Her main research interests include blockchain analysis and testing, software defect location.