基于动态调整简化粒子群优化的组合测试用例生成方法

包晓安[1] 鲍 超[1] 金瑜婷[1] 陈春宇[1] 钱俊彦[2] 张 娜[1]
(浙江理工大学信息学院 杭州 310018)[1]
(桂林电子科技大学广西可信软件重点实验室 广西 桂林 541004)[2]

摘 要 优化的组合测试中的一个关键是生成的测试用例能够覆盖更多的组合,而粒子群算法在生成强组合覆盖用例方面有其独特的优势和能力。文中提出了一种基于动态调整简化粒子群优化的组合测试用例生成方法。该方法基于粒子群算法生成测试用例,结合混合的优先级 one-test-at-a-time 策略和基于动态调整的简化粒子群算法生成组合测试用例集,排除了速度因素对粒子优化过程的影响。定义了一个粒子收敛指标,以粒子群早熟收敛程度为依据来动态调整惯性权值,以防止粒子陷入局部最优和后期出现收敛速度慢的情况,从而提高粒子群算法所生成的覆盖表的覆盖组合能力。通过对比实验表明,基于动态调整的简化粒子群优化算法在用例规模和时间成本上具有一定的优势。

关键词 简化粒子群算法,测试用例,惯性权值
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.11.031

Combinatorial Test Case Generation Method Based on Simplified Particle Swarm Optimization with Dynamic Adjustment

BAO Xiao-an[1] BAO Chao[1] JIN Yu-ting[1] CHEN Chun-yu[1] QIAN Jun-yan[2] ZHANG Na[1]
(School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China)[1]
(Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China)[2]

Abstract One of the keys for the optimized combinatorial test is that the generated test case can cover more combinations, and the particle swarm algorithm has the distinctive advantage and capability in generating strong combinatorial coverage cases. This paper proposed a combinatorial test case generation method based on simplified particle swarm optimization based on dynamic adjustment. In this method, test case is generated based on particle swarm algorithm, and the mixed priority one-test-at-a-time strategy and simplified particle swarm optimization algorithm based on dynamic adjustment are combined to generate combinatorial test case set, excluding the influence of velocity factors on the process of particle optimization. Then, a particle convergence criterion is defined, and the inertia weight is dynamically adjusted based on the premature convergence degree ofparticle swarm, so as to prevent that the particles fall into the local optimum and its convergence is slow later, thus improving the capability of coverage combination of the coverage table generated by the particle swarm algorithm. Experiments show that the simplified particle swarm optimization algorithm based on dynamic adjustment has certain advantages in the aspect of case scale and time cost.

Keywords Simplified particle swarm algorithm, Test case, Inertia weight

1 引言

软件测试是软件生命周期的一个重要过程,随着软件规模和复杂度的不断增大,这个过程变得越来越重要,同时也是控制整个软件开发进程和质量的重要环节。一个合理且高效的测试方法是节约测试成本以及提高软件质量的关键。组合测试[1]作为一种基于规约的测试方法,具有用例规模小、检错能力强的特点,能够以较低的成本完成相应的测试工作。经典的元启发式搜索算法,如蚁群算法、模拟退火算法、遗传算法等,都被用于生成组合测试用例集的研究中。粒子群算法具有参数设置少、执行速度快、实现更加便捷的特点。粒子群算法实现测试用例生成[2]的研究也一直在进行,关于组合测试中粒子群算法的应用凸显了该算法在执行效率上的优势。

在粒子群算法实现组合测试用例生成的研究中,关于更

到稿日期:2017-10-09 返修日期:2018-02-05 本文受国家自然科学基金(61502430,61379036,61562015),广西自然科学重点基金(2015GXNSFDA139038),浙江理工大学 521 人才培养计划资助。

包晓安(1973—),男,硕士,教授,主要研究方向为自适应软件、软件测试与智能信息处理;鲍 超(1990—),男,硕士,主要研究方向为软件工程、软件测试;金瑜婷(1994—),女,硕士,主要研究方向为软件测试、软件形式化方法;陈春宇(1994—),男,硕士,主要研究方向为软件工程、软件测试;钱俊彦(1973—),男,博士,教授,CCF 高级会员,主要研究方向为软件工程、形式化验证;张 娜(1977—),女,硕士,副教授,主要研究方向为软件工程、软件测试,E-mail:zhangna@zstu.edu.cn(通信作者)。

高覆盖力度以及交互力度不统一情况下的研究仍然存在不足。王子元等[4]提出的可变力度组合测试方法虽然实现了因素间实际交互关系的覆盖,但并没有考虑参数的选择对算法生成测试用例覆盖力度的影响;同时,也没有尽可能地规避过多的参数值设置对整个算法优化搜索的影响,算法的执行成本和受参数影响的程度依旧较高。文献[5]虽然在高维覆盖力度方面进行了一定的研究和创新,但也没有规避更多参数对算法效率的影响。针对这个问题,在算法具有良好鲁棒性的前提下,针对算法参数的设置,可以使用动态调整的策略;同时可以消减对算法影响程度较低的参数,以实现参数的合理调整。粒子群算法中参数设置的目的是实时调节算法的搜索范围和能力,从而得到最优的搜索结果,但过多复杂的参数设置也可能得到相反的结果,增加了算法的执行时间并降低了执行效率。

综上所述,本文将尽可能地规避过多的参数设置对粒子群算法优化结果的影响,同时根据粒子早熟的收敛程度对惯性权值进行实时、动态的调整。因此,提出一种结合混合的优先级 one-test-at-a-time 策略与动态调整简化粒子群算法的组合测试用例集生成方法。

2 基于简化粒子群优化的组合测试算法的框架

2.1 简化粒子群算法(SPSO)

粒子群优化算法是由 Eberhart 和 Kennedy 于 1995 年提出的一种源于鸟群和鱼群觅食行为的模拟。该算法可以用于解决多类复杂问题,已被应用于工程科学领域(如神经网络训练、模糊系统控制等)。关于粒子群的改进研究有很多,所有研究中对 BPSO(基本粒子群算法)的改进[6-7]是基于粒子的速度和位置的概念进行的,所有的演化公式都包含位置和速度的变量;同时,增加的许多复杂的参数取值使得算法描述变得越来越复杂,也使得 BPSO 的收敛性定量分析变得更复杂。经研究[8]发现,粒子速度的概念不是必需的,从而避免了由于确定粒子速度区间而影响粒子的收敛速度和精度。

BPSO 速度和位置的更新公式如下:

$$v_{i,d}(t+1)=\omega v_{i,d}(t)+c_1 r_1[pBest_{i,d}(t)-x_{i,d}(t)]+c_2 r_2[gBest_{i,d}(t)-x_{i,d}(t)] \tag{1}$$

$$x_{i,d}(t+1)=x_{i,d}(t)+v_{i,d}(t+1) \tag{2}$$

BPSO 中,$x_{i,d}$ 表示粒子 i 在 d 维空间的位置是当前问题的一个解,搜索过程是为了使 $x_{i,d}$ 无限接近全局最优解。而粒子速度 $v_{i,d}$ 表示粒子运动的快慢,粒子速度的大小并不影响粒子最终逼近最优解的过程,但可能影响粒子逼近最优解的进化方向,使其出现发散现象而影响粒子后期的收敛性。

式(1)与式(2)中的个体极值 $pBest_{i,d}$ 和全局极值 $gBest_{i,d}$ 与各维的搜索空间均有联系,每一维空间的更新都是相互独立的,不失一般性,我们可以在一维空间中考虑此问题。不妨设 $a_1=c_1r_1$,$a_2=c_2r_2$,$a=a_1+a_2$。将 $\rho=\dfrac{a_1 pBest_0+gBest_d}{a_1+a_2}$ 代入式(1)与式(2)中,得式(3)和式(4):

$$v(t+1)=\omega v(t)+a(\rho-x(t)) \tag{3}$$

$$x(t+1)=x(t)+v(t+1) \tag{4}$$

对式(3)和式(4)进行迭代,可得式(5):

$$x(t+2)+(a-\omega-1)x(t+1)+\omega x(t)=a\rho \tag{5}$$

可以看出,式(5)是一个不含速度变量的公式,可以简化为式(6):

$$x_{1,d}(t+1)=\omega x_{1,d}(t)+c_1 r_1(pBest_{1,d}(t)-x_{1,d}(t))+c_2 r_2(gBest_{i,d}(t)-x_{i,d}(t)) \tag{6}$$

其中,等号右边第一项表示历史位置对现在位置的影响,可以通过惯性权值来进行调整;第二项表示自我认知部分,表示粒子对自我的思考;第三项表示社会部分,体现了粒子之间的信息共享与合作。上式即为基本粒子群算法演化而成的 SPSO 的更新公式。

2.2 基于简化粒子群优化的单个测试用例生成方法

粒子群算法基于适应度和群体的概念,采用适应度函数 $fitness(x)$ 来衡量粒子的优劣,粒子群中的粒子代表优化问题的一个可能的解。在使用简化粒子群优化算法生成测试用例之前,首先对搜索的空间进行空间建模,粒子 i 在第 d 维空间上的位置 $x_{i,d}$ 表示第 d 个参数,该位置上所对应的搜索空间为 d 参数的取值集 $D_d=\{1,2,\cdots,j_d\}$,$x_{i,d}\in D_d$。为了有效防止粒子搜索过程中出现越界的情况,本文采用 Robinson 等提出的吸收墙策略,当粒子越界时,粒子将被吸收在相应的边界上,处理公式如下:

$$f(x_{i,d})=\begin{cases} l_i, & \text{if } x_{i,d}>l_i \\ 1, & \text{if } x_{i,d}<1 \\ x_{i,d}, & \text{else} \end{cases} \tag{7}$$

文中提出的单个测试用例生成方法在充分考虑算法执行效果和效率的基础上,排除了速度因素对测试用例生成过程的影响,简化了算法的步骤,从而降低了复杂度并提高了效率。

算法 1 基于简化粒子群优化的单个测试用例生成方法

```
输入:种群规模 m,参数个数 n,待覆盖组合 S
输出:测试用例 gBest
1. t=0;gBest=null;
2. for(i=1;i<m;i++){
3. 初始化所有粒子的位置矢量 x_i;
4. while(t<t_max){
5. for(i=0;i<m;i++){
6. 计算适应值 fitness(x_i);
7. if(fitness(x_i)==C(n,d)) return x_i;
8. if(fitness(x_i)>f(pBest_i)) pBest_i=x_i;
9. if(fitness(x_i)>f(gBestr_i)) gBestr_i=x_i;
10. }
11. for(i=1;i<=m;i++){
12. for(d=1;d<=n;d++) }
13.  使用式(6)更新位置 x_{i,d};
14. x_{i,d}=round(x_{i,d});//取整运算
15. x_{i,d}=f(x_{i,d});//边界处理
16. }
17. }
18. t++;//迭代次数
19. }
20. return gBest.
```

2.3 基于混合优先级的 one-test-at-a-time 策略

凭借自身简单、容易扩展等特点,one-test-at-a-time 策

略[9]已被广泛应用于组合测试用例生成的方法研究中[10-11]。该策略与粒子群算法结合生成测试用例集的思想是：构造一个空测试用例集 T，任意选择一个需要覆盖的组合，利用粒子群算法固定部分因素取值，生成单个测试用例 t，删除 t 所满足的覆盖组合，并将 t 并入 T 中，直至所有组合被覆盖，返回用例集 T。

针对该策略如何有效地选择任意需要覆盖的组合，从而优化算法的执行效率和提高算法的时间性能的问题，文献[5]给出了一种思路：通过构造度量函数来度量需覆盖组合的优先级。该方法在满足一对多覆盖（即一条组合生成的测试用例尽可能多地满足需覆盖组合）的前提下可以提高算法的执行效率，但如果需覆盖组合中满足一对多覆盖的情况很少或者没有，以及包含的组合数相同时，需覆盖组合优先级策略就失去了意义。考虑一个待测系统 $F=\{2^3\}$，使用该策略运行到某一步时，需覆盖组合如表 1 所列（“－”表示未取值的因素），4 个所需覆盖组合对应测试用例包含的组合数都是 2，无法判断其优先级，该方法失效。

表 1　需覆盖组合中测试用例的包含情况

Table 1　Inclusion situation of test case in combination to be covered

No.	UncovCombset	对应测试用例	包含组合
1	$f_1=1,f_2=1$	(1,1,－)	1,3
2	$f_2=2$	(－,2,－)	2,4
3	$f_2=1,f_3=1$	(－,1,1)	1,3
4	$f_1=2,f_2=2$	(2,2,－)	2,4

组合测试生成用例的过程包含多个因素的不同取值，一条用例就是各个因素取值的组合。因此，可以构造一个度量函数来度量因素的优先级，其目的是优先处理那些涉及更多需覆盖组合的因素，提高测试用例对需覆盖组合的贡献度。具体的因素度量函数如下：

$$T_k=\sum_{i=1}^{d}f(c_i,t_k) \tag{8}$$

$$f(c_i,t_k)=\begin{cases}1, & t_k\in c_i\\ 0, & t_i\notin c_i\end{cases} \tag{9}$$

其中，c_i 表示需覆盖表中一条不完整的测试用例；t_k 表示第 k 个未取值的因素；T_k 表示第 k 个未取值因素的优先级，即频率数值，频率数值越高则优先级越高，反之相反。通过式(8)和式(9)的计算得到表 1 中 f_3 的频率数值为 3，大于 f_1 和 f_2，因此应尽可能早地对含有因素 f_3 的需覆盖组合处理赋值。

因此，本文提出了一种基于混合优先级的 one-test-at-a-time 策略。为防止单独使用覆盖组合优先级和因素优先级这两种方法时失效，可以交叉使用这两种方法。在因素优先的基础上应用组合优先策略，既保证了优先处理涉及较多覆盖需求的因素，也最大程度地保证了生成组合覆盖率较高的测试用例。

算法 2　基于混合优先级的 one-test-at-a-time 策略

输入：未覆盖组合 UncovCombset

输出：测试用例 T

```
1. T=null;S=bull;r=0;
2. if(UncovCombet-null){
3. n=count(UncovCombset)
4. for(i=1;i<=n;i++){
5. 根据式(8)、式(9)计算因素优先级 Tk;
6. Ck=sorting(Tk) //根据 Tk 将 UncovCombset 分成 k 个组合类别;
7. 分组计算不同 Ck 中每个未覆盖组合的优先级 rk;
8. if(rk>r){
9. r=rk;S=UncovCombsetk;
9. }
10.}
11. 将 S 代入算法 1 中生成测试用例 t;
12. 提取 t 所满足的同因素优先级的组合 Combett;
13. T=T∪{t};
14.}
15. return T.
```

3　动态调整简化粒子群算法

粒子群算法中通过调整参数来控制算法的搜索范围和精度的研究十分广泛，因为粒子群算法具有较好的鲁棒性，可以通过动态调整参数来提高算法的灵活性，进而提高算法解决问题的能力。

在可调的参数中，ω（惯性权值）是最重要的参数，较大的权值能够提高算法的全局搜索能力，较小的权值能够提高算法的局部搜索能力。大量学者针对如何平衡好这种取值关系进行了研究，Shi 等[12-13]先后提出了线性递减权值策略、模糊权值策略、随机惯性权值策略，但这几种方法均有各自的局限性。线性递减策略过于简单，收敛速度过快；模糊策略需要建立一定的模糊规则，实现困难；而随机惯性权值策略只是被用于求解动态系统。因此，这些方法未被广泛应用。

本文引入一种可以评价粒子群早熟收敛程度的指标，以反映粒子群的实时收敛程度，防止粒子陷入局部最优和后期出现收敛速度慢的情况。惯性权值根据实际情况动态调整。定义粒子群的大小为 n，在第 k 次迭代过程中，粒子群中粒子 i 的适应度值为 $fitness(i)^k$，最优的粒子适应度值为 $fitness(m)^k$，粒子群的平均适应度值为 $fitness(avg)^k$。对于粒子群中适应度值优于平均适应度值的粒子，根据式(10)求其平均适应度值 $\overline{fitness(avg)^k}$：

$$\varphi=-\left|\frac{fitness(m)^k-\overline{fitness(avg)^k}}{fitness(m)^{k-1}-\overline{fitness(avg)^{k-1}}}\right| \tag{10}$$

其中，φ 为粒子群的收敛程度，φ 越小说明粒子群越收敛，反之相反。

传统粒子群算法一般使用线性递减的策略来改变惯性权值，这样的策略并不能反映搜索过程中复杂的非线性行为且收敛速度过快，从而制约了收敛的精度；并且粒子群搜索过程是相当复杂的，并非是完全的线性问题，这样的策略过于单一，解决问题的能力和范围十分有限。针对此问题，可以结合早熟收敛程度指标实时监控收敛趋向，从而根据趋向动态调整 ω 来提高惯性权值适应问题的程度，提高算法的精度，具体的调整公式如下：

$$\omega=\omega_{\max}-(\omega_{\max}+e^{-\varphi})^{-1} \tag{11}$$

其中，$\omega_{\max}$ 表示初始化时惯性权值的最大值，根据文献[14]，ω 的变化范围为(0.4,0.9)，ω 取值的大小反映了 SPSO 的全局和局部搜索能力的大小。ω 取值较大时，表示全局搜索能力较强而局部搜索能力较弱；ω 取值较小时，表示局部搜索能力较强而全局搜索能力较弱。合理的取值可以提高 SPSO 算法

的性能，提高粒子群的寻优能力，从而减少迭代的次数。

当 φ 取值减小时，即 SPSO 处于收敛状态，需要增大 ω 的取值以增强全局搜索能力；相反，当 φ 取值减小时，需要提升算法的局部搜索能力以减小 ω 的取值。由式(11)和图 1 可知，φ 与 ω 在区间[0.4,0.9]内的单调性相反，符合本文的权值调整逻辑。

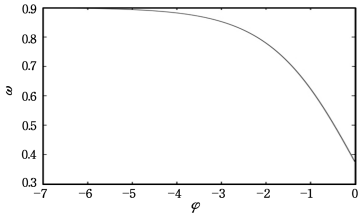

图 1 惯性权值 ω 与 φ 值的变化趋势

Fig. 1 Change trends of inertia weights ω and φ

4 实验与分析

为了评价本文改进策略的实际效果，将采用 MATLAB 编程实现 BPSO 算法和 SPSO 算法。分别结合早熟粒子群评价指标以及由其确定的惯性权值取值策略，对比两种算法下的实验结果。为了提高实验的准确性，将采用 15 组拥有不同复杂程度的实例(见表 2)进行实验分析，其中覆盖矩阵(CA)、混合覆盖矩阵(MCA)以及可变力度覆盖矩阵(VSCA)分别为 5 组。

表 2 实例覆盖表

Table 2 Example coverage table

VSCA	MCA	CA
$VSCA_1(3,10,4,MCA(5,4^3))$	$MAC_6(3,9,5^2 3^3 2^4)$	$CA_{11}(3,9,4)$
$VSCA_2(2,14,5^4 3^8,MCA(4,3^3))$	$MAC_7(2,25,4^6 3^{13} 3^{10})$	$CA_{12}(4,9,9)$
$VSCA_3(2,13,5^4 4^3 3^5,MCA(3,3^5))$	$MAC_8(2,8,8^2 7^2 5^2 3^2)$	$CA_{13}(2,9,21)$
$VSCA_4(2,15,2,CA(2,3^5))$	$MAC_9(3,6,8^2 7^1 6^2 5^1)$	$CA_{14}(2,8,9)$
$VSCA_5(3,9,4^3 3^3 2^3,CA(4,3^4))$	$MAC_{10}(5,4,5^3 4^3)$	$CA_{15}(2,4,3)$

为了规避粒子群算法执行过程中随机因素对结果产生的影响，本文将每组实例独立运行 N 次，直至每次运行的平均值趋于一个相对稳定的区间范围，最后取平均值作为实验对比的依据。本文设定的最大迭代次 $T_{\max}=1000$，$c_1=c_2=2$，r_1，r_2 是在[0,1]内服从均匀分布的随机数。ω 通过式(11)计算得到。

观察表 2 可知，基于动态调整的简化粒子群算法在总体用例规模上优于基本粒子群算法的执行结果，尤其在比较复杂的覆盖表的情况下，其用例规模明显小于基本粒子群算法，精简的用例规模能达到 18%左右。而 SPSO 算法在除个别简单覆盖表外，时间成本接近或者略高于 BPSO 算法，绝大部分情况下时间成本也明显低于 BPSO 算法，节约的时间成本最高能达到 72%左右。

表 3 从时间成本和用例规模的维度上说明了 SPSO 具有一定的优势，但前提必须是不改变或者提高生成的用例集合的检错能力。

表 3 实验结果对比

Table 3 Comparison of experimental results

Covering Array	BPSO		SPSO	
	Number of cases	Time/s	Number of cases	Time/s
$VSCA_1$	67.5	129	54.7	102
$VSCA_2$	107.7	582	93.4	208
$VSCA_3$	212.6	873	191.2	284
$VSCA_4$	46.6	47	38.3	28
$VSCA_5$	59.8	142	52.4	40
MAC_6	36.4	103	31.6	38
MAC_7	124	186	101	82
MAC_8	101.3	152	92.7	73
MAC_9	3789.2	17861	3012.1	10416
MAC_{10}	89.7	73	71.4	56
CA_{11}	1132.6	4013	1017.5	1897
CA_{12}	39.5	301	35.2	116
CA_{13}	158.7	253	132.5	111
CA_{14}	132.1	211.3	128.7	101
CA_{15}	9	11	9	12

为此，本文采用 $APFD$ 值来度量 BPSO 和 SPSO 的实例检错能力来验证 SPSO 的有效性[15-16]。$APFD$ 的具体计算公式如下：

$$APFD=1-\frac{TF_1+TF_2+\cdots+TF_m}{nm}+\frac{1}{2n} \tag{12}$$

其中，n 为测试用例的个数；m 为检测出的错误个数；TF_i 为检测出第 i 个错误所需的用例个数；$APFD$ 的取值区间为[0，100%]，数值越高则检错效果越好。对 15 组实例进行模拟测试，测试停止的标准均为需求覆盖率达到 100%。两种方法的 $APFD$ 的具体数值情况如图 2 所示。

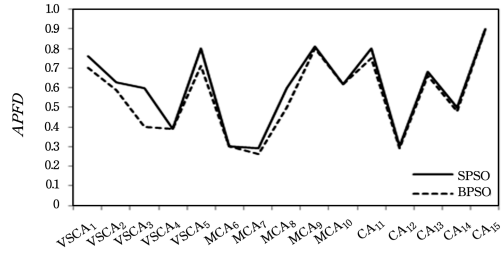

图 2 $APFD$ 数值情况

Fig. 2 $APFD$ values

由图 2 可知，SPSO 的 $APFD$ 值普遍接近或者高于 BPSO，即该算法的检错能力普遍优于 BPSO，能够保证经改进后算法的检错能力有一定提升。

由上述可知，SPSO 算法在提升检错能力的前提下，无论在用例规模还是在时间成本上都具有一定的优势，尤其是在时间成本上，刚好验证了去除多余参数对算法执行时间成本的积极影响；同时结合动态调整参数取值使得算法的执行效率得到了有效提升，实现了在低成本下高效的测试用例生成。

结束语 关于粒子群优化组合测试用例生成的研究大多是针对固定力度覆盖表或者是可变力度交互覆盖进行的单一研究，而不存在满足任意复杂程度覆盖表的方法。对此，本文在文献[10]的基础上进行了深入的研究，排除了更多的干扰因素，优化了参数的取值方法，避免粒子陷入局部最优和后期出现收敛性不足的情况，使 SPSO 满足并提升了基本检错能力，满足生成任意强度覆盖表的能力，还提高了算法的执行效率并缩减了成本。本文创新性地利用简化粒子优化解决组合测试问题，不仅提高了解决问题的效率，同时也为关于粒子群优化的组合测试用例生成研究提供了新的思路和方法，对后

续的学术研究工作具有积极的指导意义。

本文提出的基于混合优先级的 one-test-at-a-time 策略弥补了现有研究的一些缺陷,但是否存在普遍适用和效率更高的用例生成策略还需要进一步的探索和研究。

参 考 文 献

[1] WANG Z Y,XU B W,NIE C H,et al. Survey of combinatorial test generation [J]. Journal of Frontiers of Computer Science and Technology,2008,2(6):571-588. (in Chinese)
王子元,徐宝文,聂长海,等. 组合测试用例生成技术[J]. 计算机科学与探索,2008,2(6):571-588.

[2] JIA J T. Research of Automatic Testcase Generation Functions Based onParticle Swarm Optimization Algorithm [J]. Computer Technology and Development,2010,20(9):24-27. (in Chinese)
贾冀婷. 基于粒子群算法的测试用例自动生成方法研究[J]. 计算机技术与发展,2010,20(9):24-27.

[3] WANG Z Y,NIE C H,XU B W,et al. Optimal Test Suite Generation Methods for Neighbor Factors Combinatorial Testing [J]. Chinese Journal of Computers,2007,30(2):200-211. (in Chinese)
王子元,聂长海,徐宝文,等. 相邻因素组合测试用例集的最优生成方法[J]. 计算机学报,2007,30(2):200-211.

[4] WANG Z Y,QIAN J,CHEN L,et al. Generating Variable Strength Combinatorial Test Suite with one-test-at-a-time Strategy [J]. Chinese Journal of Computers,2012,35(12):2541-2552. (in Chinese)
王子元,钱巨,陈林,等. 基于 One-test-at-a-time 策略的可变力度组合测试用例生成方法[J]. 计算机学报,2012,35(12):2541-2552.

[5] BAO X A,YANG Y J,ZHANG N,et al. Test Case Generation Method Based on Adaptive Particle Swarm Optimization [J]. Computer Science,2017,44(6):177-181. (in Chinese)
包晓安,杨亚娟,张娜,等. 基于自适应粒子群优化的组合测试用例生成方法[J]. 计算机科学,2017,44(6):177-181.

[6] BERGH F V D,ENGELBRECHT A P. Cooperative learning in neural networks using particle swarm optimizers[J]. South African Computer Journal,2000,26:84-90.

[7] RATNAWEERA A,HALGAMUGE S K,WATSON H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. IEEE Transactions on Evolutionary Computation,2004,8(2):240-255.

[8] HU W,LI Z S. A Simpler and More Effective Particle Swarm Optimization Algorithm [J]. Journal of Software,2007,18 (4):861-868. (in Chinese)
胡旺,李志蜀. 一种更简化而高效的粒子群优化算法[J]. 软件学报,2007,18(4):861-868.

[9] COHEN D M,DALAL S R,FREDMAN M L,et al. The AETG System:An Approach to Testing Based on Combinatorial Design [J]. IEEE Transactions on Software Engineering,1997,23(7):437-444.

[10] CHEN X,GU Q,WANG Z Y,et al. Framework of Particle Swarm Optimization Based Pairwise Testing [J]. Journal of Software,2011,22(12):2879-2893. (in Chinese)
陈翔,顾庆,王子元,等. 一种基于粒子群优化的成对组合测试算法框架[J]. 软件学报,2011,22(12):2879-2893.

[11] WILLIAMS A W. Determination of Test Configurations for Pair-Wise Interaction Coverage[C]// International Conference on Testing Communicating Systems:TOOLS and Techniques. DBLP,2000:59-74.

[12] SHI Y,EBERHART R C. Fuzzy adaptive particle swarm optimization[C]// Proceedings of the 2001 Congress on Evolutionary Computation. IEEE Xplore,1997:101-106.

[13] EBERHART R C,SHI Y. Tracking and optimizing dynamic systems with particle swarms[C]// Proceedings of the 2001 Congress on Evolutionary Computation,2001. IEEE,2002:94-100.

[14] YOU B,CHEN G,GUO W. A Discrete PSO-Based Fault-Tolerant Topology Control Scheme in Wireless Sensor Networks[C]// Advances in Computation and Intelligence—International Symposium(Isica 2010). Wuhan,China,DBLP,2010:1-12.

[15] GONG M G,JIAO L C,YANG D D,et al. Research on Evolutionary Multi-Objective Optimization Algorithms [J]. Journal of Software,2009,20(2):271-289. (in Chinese)
公茂果,焦李成,杨咚咚,等. 进化多目标优化算法研究[J]. 软件学报,2009,20(2):271-289.

[16] ZHANG N,YAO L,BAO X A,et al. Multi-Objective Optimization Based On-Line Adjustment Strategy of Test Case Prioritization[J]. Journal of Software,2015,26(10):2451-2464. (in Chinese)
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.

(上接第 192 页)

[13] YIN Q Q. Secure deduplication approach based on Bloom Filter in hybrid cloud storage environments[J]. Computer Engineering and Applications,2018,54(10):73-80. (in Chinese)
尹勤勤. 基于 Bloom Filter 的混合云存储安全去重方案[J]. 计算机工程与应用,2018,54(10):73-80.

[14] BELLARE M,KEELVEEDHI S,RISTENPART T. Message-Lcked Ecryption and Secure Deduplication[M]// Advances in Cryptology — EUROCRYPT 2013. Berlin:Springer,2013:296-312.

[15] BELLARE M,KEELVEEDHI S,RISTENPART T. DupLESS:server-aided encryption for deduplicated storage[C]// Usenix Conference on Security. USENIX Association,2013:179-194.

[16] HALEVI S,HARNIK D,PINKAS B,et al. Proofs of ownership in remote storage systems[C]// ACM Conference on Computer and Communications Security. ACM,2011:491-500.

[17] BLASCO J,DI PIETRO R,ORFILA A,et al. A tunable proof of ownership scheme for deduplication using bloom filters[C]// 2014 IEEE Conference on Communications and Network Security (CNS). IEEE,2014:481-489.

[18] YANG C,ZHANG M,JIANG Q,et al. Zero knowledge based client side deduplication for encrypted files of secure cloud storage in smart cities [J]. Pervasive & Mobile Computing,2017,41:243-258.

[19] LIU X,SUN W,LOU W,et al. One-tag checker:Message-locked integrity auditing on encrypted cloud deduplication storage[C]// IEEE Conference on Computer Communications. IEEE,2017.

[20] LI J,LI Y,CHEN X,et al. A hybrid cloud approach for secure authorized deduplication[J]. IEEE Transactions on Parallel and Distributed Systems,2015,26(5):1206-1216.