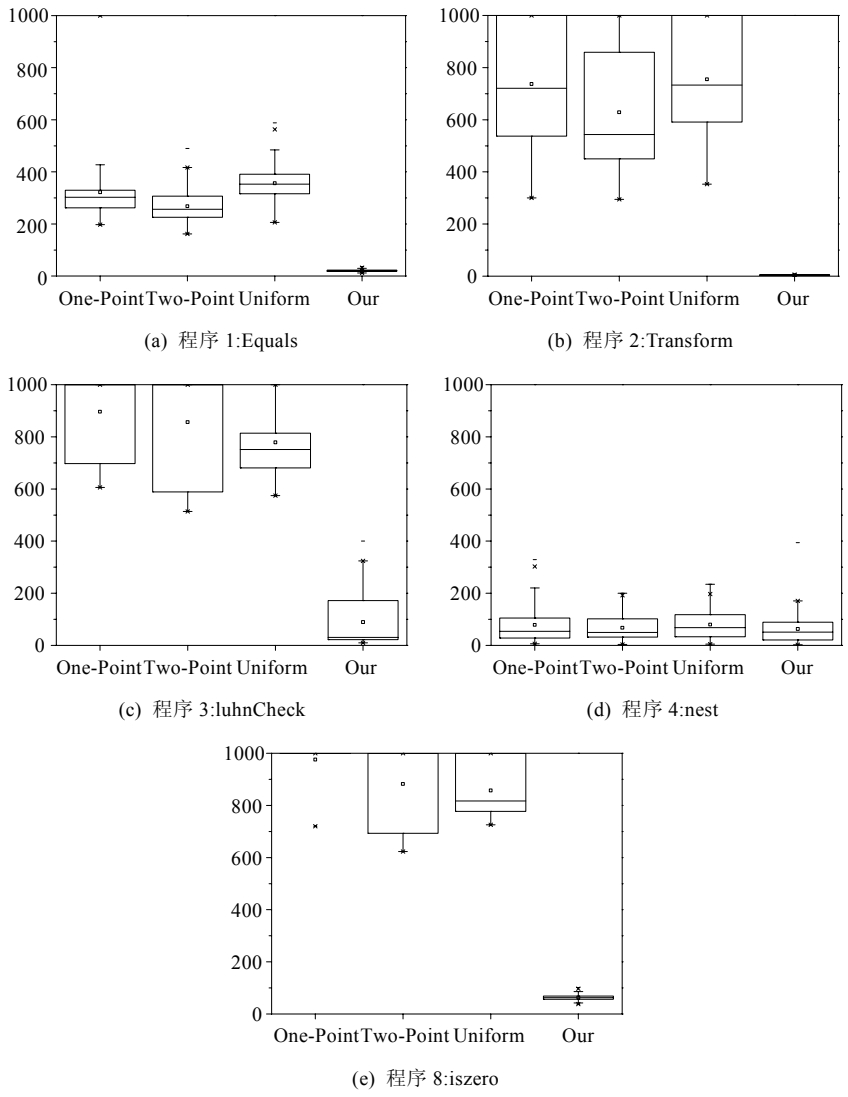


(a) 程序 1:Equals (b) 程序 2:Transform

(c) 程序 3:luhnCheck (d) 程序 4:nest

(e) 程序 8:iszero

Fig.9 Comparision of generations between different crossovers

图 9 不同交叉方式平均进化代数对比图

3.4.2 本文方法与其他文献方法的比较

为了验证本文方法 CL-PSO 相对于其他方法是否有优势,将 CL-PSO 与原始 PSO 方法、APSO 和我们前期的 OL-PSO 方法进行了对比,表 5 为分别使用 4 种方法对每个测试程序执行 100 次后的覆盖率和平均进化代数.

由表 5 可以看出:OL-PSO 和本文方法 CL-PSO 明显优于 APSO 方法和 PSO 方法,将覆盖率提升至了 100%,尤其针对程序 2、程序 3 和程序 8 等较难覆盖的程序来说,其平均进化代数也有了较大程度的降低,极大地提高了测试用例生成效率.而针对原本就可达到 100%覆盖率的程序而言,OL-PSO 和 CL-PSO 方法的平均进化代数有一定程度的降低.

OL-PSO 是将正交搜索与局部搜索策略相结合的方法;而 CL-PSO 是利用交叉算子对最优模式进行组合,再使用局部搜索算法进行搜索.这两种方法在覆盖率上均为 100%,无差异.而在平均进化代数上,只有程序 2、程序 3 和程序 4 差异较大一些.这是由于这 3 个程序均含有独立模式,适合用交叉算子对其进行模式组合,因此适

合使用 CL-PSO 方法进行测试用例生成.而针对于不含独立模式的程序,比如程序 5、程序 6 和程序 7,二者差异不大,甚至 OL-PSO 的效果要稍微比 CL-PSO 方法好一些.这是由于,针对不含独立模式的程序,CL-PSO 无法发挥其交叉算子的作用,而 OL-PSO 依然可以对其进行正交搜索,使算法的全局搜索性能和局部搜索性能得以协调,因此,CL-PSO 对存在独立模式的程序而言更加有效.

Table 5 Average coverage and average generations of the four approaches

表 5 4 种方法的平均覆盖率和平均进化代数

测试程序	PSO		APSO		OL-PSO		CL-PSO	
	覆盖率(%)	平均进化代数	覆盖率(%)	平均进化代数	覆盖率(%)	平均进化代数	覆盖率(%)	平均进化代数
1	81	320.9	100	185.87	100	3.07	100	4.01
2	65	578.24	62	479.36	100	120.76	100	6.47
3	39	783.07	63	728.71	100	283.7	100	25.9
4	100	403.56	100	177.33	100	67.82	100	7.49
5	100	267.1	100	156.18	100	134.5	100	138.14
6	100	377.04	100	159.05	100	63.49	100	70.9
7	100	63.07	100	32.23	100	8.42	100	9.1
8	19	921.47	54	935.5	100	22.7	100	23.61

为了更好地对比本文方法与其他 3 种方法的区别,我们以每种方法收集到的 100 次进化代数为样本,对实验结果进行了统计检验和效应量分析.表 6 为利用 Wilcoxon 秩和检验和 Cohen *d* 的效应量对各种方法进行分析的结果.

Table 6 *p-value* and effect size of the four approaches

表 6 4 种方法的 *p-value* 值和效应量值

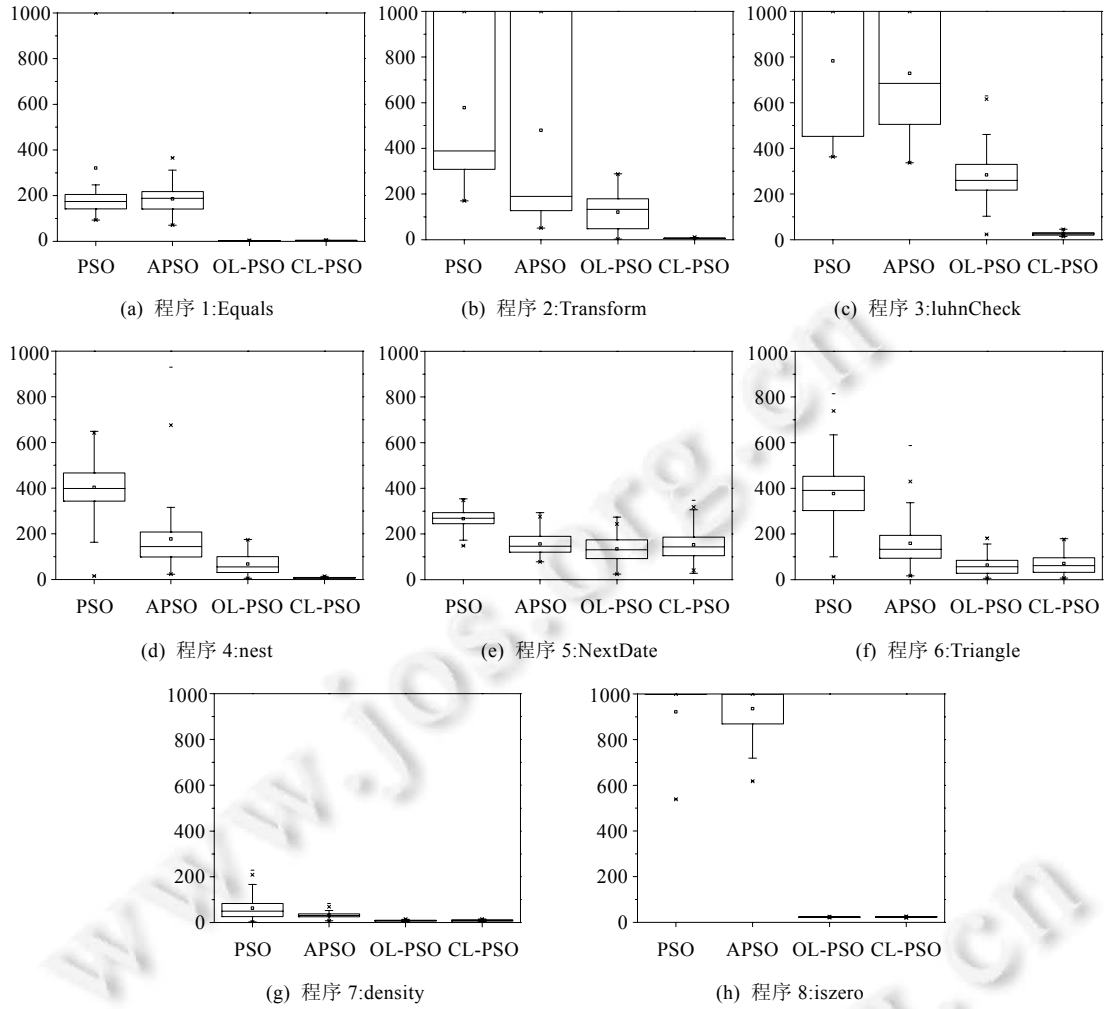
测试程序	PSO vs. CL-PSO		APSO vs. CL-PSO		OL-PSO vs. CL-PSO	
	p-value	Cohen *d*	*p-value*	Cohen *d*	*p-value*	Cohen *d*
1	<0.001	1.35 (L)	<0.001	4.06 (L)	1	−2.44 (L)
2	<0.001	2.52 (L)	<0.001	1.61 (L)	<0.001	2.03 (L)
3	<0.001	3.92 (L)	<0.001	4.05 (L)	<0.001	3.63 (L)
4	<0.001	5.09 (L)	<0.001	1.84 (L)	<0.001	1.87 (L)
5	<0.001	2.91 (L)	0.009	0.38 (S)	0.763	−0.07
6	<0.001	2.78 (L)	<0.001	1.16 (L)	0.827	−0.17
7	<0.001	1.64 (L)	<0.001	2.47 (L)	0.928	−0.23 (S)
8	<0.001	7.77 (L)	<0.001	13.33 (L)	1	−0.82 (L)

在表 6 中的统计检验部分,本文提出的空假设分别为 PSO,APSO,OL-PSO 的平均进化代数,与本文的 CL-PSO 方法差异不显著.而对于 PSO 和 APSO 方法来说,所有程序的 *p-value* 均小于 0.05,即可以拒绝原假设,说明本文方法 CL-PSO 的平均进化代数与 PSO 和 APSO 方法存在显著性差异.而对于 OL-PSO 方法来说,只有程序 2、程序 3 和程序 4 这 3 个程序的 *p-value* 小于 0.05,而其他 5 个程序都不能拒绝原假设,即,本文方法 CL-PSO 与 OL-PSO 差异不显著.

在效应量分析部分,d=0.8,d=0.5 和 d=0.2 分别对应大(L)、中(M)、小(S)这 3 种效应量.对于 PSO 和 APSO 来说,基本上大部分程序的效应量均大于 0.8,效应量程序较高.而对于 OL-PSO 来说,其中 5 个程序的效应量为负值,说明 OL-PSO 平均进化代数的平均值比本文方法 CL-PSO 小,并且在这 5 个程序中,有 3 个程序的效应量程度都是较小的,即样本重叠程度较高,说明本文方法与 OL-PSO 方法差异不大.

图 10 为 8 个程序分别在 PSO,APSO,OL-PSO 和 CL-PSO 这 4 种方法下的平均进化代数分布情况,其中,横坐标标识 4 种方法,纵坐标标识平均进化代数.

由图 10 可以看出,OL-PSO 和本文的 CL-PSO 方法在 8 个程序中平均进化代数都比 PSO 和 APSO 方法要低.同时,对于含有独立模式的程序 2、程序 3 和程序 4 来说,使用了交叉算子对最优模式进行组合的 CL-PSO 方法明显要比 OL-PSO 方法更有优势,说明 CL-PSO 针对于此类程序是有效的.



(a) 程序 1:Equals (b) 程序 2:Transform (c) 程序 3:luhnCheck

(d) 程序 4:nest (e) 程序 5:NextDate (f) 程序 6:Triangle

(g) 程序 7:density (h) 程序 8:iszero

Fig.10 Evolution algebra of different approaches

图 10 不同方法平均进化代数对比图

3.4.3 实验结果分析

通过前面两节的实验结果,我们对实验设计部分提出的两个问题有如下回答.

回答问题 1. 本文设计的用于进行模式组合的交叉算子相对于其他交叉方式,如单点交叉、双点交叉、均匀交叉是否有优势?

通过表 3 和图 9 可以看出:本文设计的交叉算子针对含有独立模式的程序效果均优于单点交叉、双点交叉和均匀交叉这 3 种交叉方式,且使用我们的方法也比其他 3 种交叉方式的结果要稳定一些,数据分布较为集中.

回答问题 2. 本文方法相比于其他文献的方法是否有优势?

从表 5 和图 10 可以看出:与原始 PSO 方法、APSO 方法相比,本文方法 CL-PSO 在覆盖率上有明显的优势;而对于 OL-PSO,二者覆盖率都达到了 100%,覆盖率上无差异.只有在含有独立模式的程序 2、程序 3 和程序 4 中可以看出:CL-PSO 比 OL-PSO 在平均进化代数上有明显的减小,在测试用例生成方面更具有优势.这是由于,CL-PSO 使用了交叉算子法对模式进行了组合.而对于其他程序而言,CL-PSO 和 OL-PSO 在平均进化代数上基本上无差异.

4 有效性分析

影响本文实验有效性的因素包括两个方面.

一方面是内部因素,即粒子群优化算法本身对实验结果的影响.因为粒子群优化算法是一种随机性算法,会使实验结果有一定的随机性.因此,为了降低算法随机性带来的影响,本文实验结果取 100 次重复实验的平均值;

另一方面是外部因素,如程序本身对实验结果的影响.目前,本文方法只适用于存在独立模式的程序,而独立模式的基础是假设所有分支都是独立的,若分支之间存在相互关系,则其无法满足独立模式的条件,即为互斥模式.因此,今后的工作将研究如何解决分支间相互影响的问题.

5 相关工作

基于搜索的测试用例生成是基于搜索的软件工程(search-based software engineering,简称 SBSE)[13]领域的一个重要分支,是软件测试用例生成领域的一个重要的研究方向.1992 年,Xanthakis 等人[14]首次将遗传算法应用于软件测试用例生成.Sthamer[15]在其博士论文中,使用遗传算法研究了分支测试、循环测试等问题.Wegener 等人[16]建立了基于遗传算法的测试用例生成环境,通过对一些工业程序的实验,其结果表明,基于遗传算法的测试用例生成较随机法具有更高的覆盖率.Fraser 和 Arcuri[17−19]介绍了一种针对 Java 程序的测试用例集自动生成工具——EvoSuite.该工具主要采用遗传算法产生测试用例集[18],文献[19]通过实验对随机选取的工业 Java 程序进行实验,结果表明,使用 EvoSuite 生成测试用例可以达到较高的分支覆盖率.在文献[20]中,Fraser 等人使用文化基因算法代替 EvoSuite 中的遗传算法,加入局部搜索算子,提高了分支覆盖率.相较于遗传算法在测试数据生成中的应用研究[21−24]较为完善有所不同,PSO 算法在测试数据生成中的研究才刚起步.Windisch 等人[5]将 PSO 算法应用到测试数据生成中,通过对一些程序进行实验研究,结果表明,PSO 算法在代码覆盖和执行效率上要优于遗传算法.毛澄映等人[25]以分支覆盖为准则,选取了 4 种典型 PSO 算法的变体与遗传算法和模拟退火算法进行测试数据生成效果的实证分析,实验结果表明,使用 PSO 算法在分支覆盖率等性能上要优于遗传算法和模拟退火算法.

针对程序因结构复杂而导致其分支条件无法得到评价这一问题,McMinn[2]对源代码进行了修改,通过中间变量得到内层分支的评价信息,可使所有的分支条件得到完全评价.而本文在不修改源代码的基础上,仅针对这些未能被评价到的条件的插桩方式进行了改变,将插桩语句放到最外层,可有效解决适应度函数不能完全评价所有分支的问题,且没有过多地引入时间消耗.

McMinn[2]对大量程序进行了实验,并证明:对于含有模式的程序,遗传算法由于其特有的交叉运算,可以对模式进行组合,生成更优的个体,因此优于其他算法.但由于遗传算法中的交叉算子在进行交叉时容易破坏程序中的模式,且由于遗传算法存在选择操作,含有最优模式的个体有可能因为其适应度值低而被淘汰.因此,本文将交叉算子引入到 PSO 中,并设计了一种新的交叉算子,将所有最优模式均集中到种群中最优秀的个体上,而其他个体均向该优秀个体进行学习,加快了进化过程.实验部分通过与其他的交叉方式进行对比,证明本文的交叉方式不但不会破坏个体中的最优模式,还可以因其他个体向最优个体的学习,从而较快地收敛到全局最优解.

Zhu 等人[10]提出了一种 APSO 算法进行测试用例生成,主要根据当前个体的适应度值对惯性权重进行调整,并且对个体进行早熟收敛判断,超过一定次数时,则重新初始化该粒子.史娇娇等人[26]提出了一种自适应 PSO 算法,并将其应用于测试用例生成中.我们的前期工作[6]提出了一种基于正交搜索和局部搜索相结合的粒子群优化测试用例生成方法,利用奇异值分解的方法预测种群进化方向,在其正交方向生成新种群,增强算法的全局搜索能力.此外,还使用局部搜索策略 AVM 来增强算法的局部搜索能力,使其全局和局部搜索能力相协调,更高效地生成测试用例.

上述文献中,应用 PSO 算法生成测试用例时,主要集中改进 PSO 算法,或者调节算法中的参数,或者增强算法的全局或局部搜索能力,然而并没有考虑程序的特殊结构对测试用例生成效率的影响.而本文主要针对含有独立模式的程序设计了一种新的交叉算子,并且通过实验与相关文献中的方法进行了比较.

6 结束语

本文首先针对存在独立模式的程序,改变了其分支函数插桩的方法,解决了其模式所对应的分支无法得到评价的问题;然后设计了一种新的交叉算子,通过获取的每个模式的适应度值,将所有个体中的最优模式通过交叉算子集中到一个个体中,再使用粒子群优化算法进行测试用例生成,其他的个体在进化过程中都可以向这个最优的个体学习,加速粒子群进化;并且在进化过程中,对于每代的最优个体还使用了局部搜索策略,以进一步提高测试用例生成效率;最后,为了评价我们设计的交叉算子对于含有独立模式的程序的有效性,在实验部分与单点交叉、双点交叉和均匀交叉这 3 种交叉方式进行对比,结果表明:我们的交叉算子在交叉过程中由于没有破坏最优模式,而是将其当作一个整体进行交叉,因此效果远好于其他 3 种交叉算子.并且还将本文的方法 CL-PSO 与 PSO,APSO 和 OL-PSO 进行了对比,实验结果表明:CL-PSO 方法在对含有模式的程序进行测试用例生成时,相对于其他方法在覆盖率和平均进化代数上均有一定的优势.

由于本文的 CL-PSO 方法只针对含有独立模式的程序,而对于含有互斥模式的程序,由于模式之间存在相互影响,故本文未对其进行分析和处理.因此,我们下一步工作将分析互斥模式之间的关系,探索互斥模式在程序中受哪些因素的影响,并设计新的组合方式针对含有互斥模式的程序进行测试用例生成.

致谢 在此,我们对审稿人和编辑表示感谢,对本文提出建议的同行表示感谢.

References:

[1] Harman M, Jones BF. Search based software engineering. Information and SoftwareTechnology, 2001,43(14):833−839.

[2] McMinn P. An identification of program factors that impact crossover performance in evolutionary test input generation for thebranch coverage of C programs. Information and Software Technology, 2013,55(1):153−172. [doi: 10.1016/j.infsof.2012.03.010]

[3] McMinn P, Binkley D, Harman M. Empirical evaluation of a nesting testability transformation for evolutionary testing. ACM Trans. on Software Engineering and Methodology, 2009,18(3):824−833. [doi: 10.1145/1525880.1525884]

[4] Holland JH. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press, 1992.

[5] Windisch A, Wappler S, Wegener J. Applying particle swarm optimization to software testing. In: Lipson H, ed. Proc. of the 9th Annual Conf. on Genetic and Evolutionary Computation. New York: ACM Press, 2007. 1121−1128. [doi: 10.1145/1276958.1277178]

[6] Wang LS, Jiang SJ, Zhang YM, Yu Q. Test case generation based on orthogonal exploration and particle swarm optimization. Acta Electronica Sinica, 2014,42(12):2345−2351 (in Chinese with English abstract).

[7] Korel B. Automated software test data generation. IEEE Trans. on Software Engineering, 1990,16(8):870−879. [doi: 10.1109/32.57624]

[8] Xiao M, El-Attar M, Reformat M, Miller J. Empirical evaluation of optimization algorithms when used in goal-oriented automated test data generation techniques. Empirical Software Engineering, 2007,12(2):183−239. [doi: 10.1007/s10664-006-9026-0]

[9] Kifetew MF, Panichella A, Lucia AD, Oliveto R, Tonella P. Orthogonal exploration of the search space in evolutionary test case generation. In: Pezzè M, Harman M, eds. Proc. of the 2013 Int'l Symp. on Software Testing and Analysis. Lugano: ACM Press, 2013. 257−267. [doi: 10.1145/2483760.2483789]

[10] Zhu XM, Yang XF. Software test data generation automatically based on improved adaptive particle swarm optimizer. In: Zhang J, ed. Proc. of the 2010 Int'l Conf. on Computational and Information Sciences. Chengdu: IEEE CPS, 2010. 1300−1303. [doi: 10.1109/ICCIS.2010.321]

[11] Wilcoxon F. Individual comparisons by ranking methods. Biometrics Bulletin, 1945,1(6):80−83.

[12] Cohen J. Statistical Power Analysis for the Behavioral Sciences. 2nd ed., Lawrence Erlbaum Assoc. Inc., 1988.

[13] Harman M, Jia Y, Zhang Y. Achievements, open problems and challenges for search based software testing. In: Wotawa F, ed. Proc. of the IEEE Int'l Conf. on Software Testing, Verification & Validation. Graz: GUM Press, 2015. 1−12. [doi: 10.1109/ICST.2015.7102580]

[14] McMinn P. Search_Based software testing: Past, present and future. In: Schieferdecker I, Pretchner A, eds. Proc. of the 2011 4th Int'l Conf. on Software Testing, Verification and Validation Workshops. Berlin: CPS Press, 2011. 153−163. [doi: 10.1109/ICSTW.2011.100]

[15] Sthamer H. The automatic generation of software test data using genetic algorithms [Ph.D. Thesis]. Brirain: University of Glamorgan, 1996.

[16] Wegener J, Baresel A, Sthamer H. Evolutionary test environment for automatic structural testing. Information and Software Technology, 2001,43(14):841−854. [doi: 10.1016/S0950-5849(01)00190-2]

[17] Fraser G, Arcuri A. EvoSuite: Automatic test suite generation for object-oriented software. In: Gyimóthy T, Zeller A, eds. Proc. of the 19th ACM SIGSOFT Symp. and the 13th European Conf. on Foundations of Software Engineering (FSE). New York: ACM Press, 2011. 416−419. [doi: 10.1145/2025113.2025179]

[18] Fraser G, Arcuri A. Whole test suite generation. IEEE Trans. on Software Engineering, 2013,39(2):276−291. [doi: 10.1109/TSE.2012.14]

[19] Fraser G, Arcuri A. A large scale evaluation of automated unit test generation using EvoSuite. ACM Trans. on Software Engineering and Methodology (TOSEM), 2014,24(2):8:1−8:42. [doi: 10.1145/2685612]

[20] Fraser G, Arcuri A, Mcminn P. A memetic algorithm for whole test suite generation. Journal of Systems & Software, 2014,103:311−327. [doi: 10.1016/j.jss.2014.05.032]

[21] Michael C, McGraw G, Schatz M. Generating software test data by evolution. IEEE Trans. on Software Engineering, 2001,27(12):1085−1110. [doi: 10.1109/32.988709]

[22] Miller J, Reformat M, Zhang H. Automatic test data generation using genetic algorithm and program dependence graphs. Information and Software Technology, 2006,48:586−605. [doi: 10.1016/j.infsof.2005.06.006]

[23] Watkins A, Hufnagel EM. Evolutionary test data generation: A comparison of fitness functions. Software Practice and Experience, 2006,36(1):95−116. [doi: 10.1002/spe.684]

[24] Hermadi I, Lokan C, Sarker R. Genetic algorithm based path testing: Challenges and key parameters. In: Huang XH, Xu LD, Zhou ZD, eds. Proc. of the 2010 2nd WRI World Congress on Software Engineering. Wuhan: IEEE Computer Society Press, 2010. 241−244. [doi: 10.1109/WCSE.2010.82]

[25] Mao CY, Yu XX, Xue YZ. Algorithm design and empirical analysis for particle swarm optimization-based test data generation. Journal of Computer Research and Development, 2014,51(4):824−837 (in Chinese with English abstract).

[26] Shi JJ, Jiang SJ, Han H, Wang LS. Adaptive particle swarm optimization algorithm and its application in test data generation. Acta Electronica Sinica, 2013,41(8):1555−1559 (in Chinese with English abstract).

附中文参考文献:

[6] 王令赛,姜淑娟,张艳梅,于巧.基于正交搜索的粒子群优化测试用例生成方法.电子学报,2014,42(12):2345−2351.

[25] 毛澄映,喻新欣,薛云志.基于粒子群优化的测试数据生成及其实证分析.计算机研究与发展,2014,51(4):824−837.

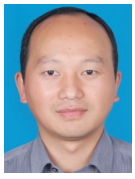
[26] 史娇娇,姜淑娟,韩寒,王令赛.自适应粒子群优化算法及其在测试数据生成中的应用研究.电子学报,2013,41(8):1555−1559.

姜淑娟(1966－),女,山东莱阳人,博士,教授,博士生导师,CCF 会员,主要研究领域为软件分析与测试,编译技术.

张艳梅(1981－),女,博士,讲师,CCF 会员,主要研究领域为软件分析与测试.

王令赛(1989－),女,硕士,主要研究领域为软件测试.

于巧(1989－),女,博士生,主要研究领域为软件分析与测试.



薛猛(1979－),男,讲师,CCF 会员,主要研究领域为软件测试,测试数据生成.

姚慧冉(1989－),女,硕士生,主要研究领域为软件测试,测试数据生成.