文章编号:1671-4598(2015)07-2247-05 DOI:10.16526/j.cnki.11—4762/tp.2015.07.003 中图分类号:TP206.3 文献标识码:A

基于云环境的分布式软件接口自动化测试

殷 琦[1],杜 明[1],刘晓强[1],常 姗[1],蔡立志[2],刘振宇[2]
(1.东华大学 计算机科学与技术学院,上海 201620;
2.上海计算机软件技术开发中心 上海市计算机软件评测重点实验室,上海 201112)

摘要:软件测试是保证软件质量,提供可靠服务的重要技术手段;目前基于 Web Service 的分布式软件越来越多,其测试技术手段也越来越受到关注;Web Service 的分布性和多样性使手工测试变得非常低效,因而需要不断提高 Web Service 测试的自动化程度;另外,云计算因其计算成本低、可伸缩性强的特点为自动化测试提供了新的支持环境;本文结合 Web Service 的测试需求,首次提出了基于云环境的 Web Service 接口自动化测试的技术框架,分析了框架内原子 Web Service、组合 Web Service 测试的关键技术,并研发了基于 CloudStack 云平台的自动化测试的原型系统;实验结果表明,文章所提出的基于云平台的 Web Service 自动化测试方案可行且提高了测试效率。

关键词:云计算;自动化测试;Web Service;功能测试

Distributed Software Interface Automated Testing Based on Cloud Environment

Yin Qi[1], Du Ming[1], Liu Xiaoqiang[1], Chang Shan[1], Cai Lizhi[2], Liu Zhenyu[2]
(1. School of Computer Science and Technology, Donghua University, Shanghai 201620, China)
(2. Shanghai Key Laboratory of Computer Software Testing and Evaluating, Shanghai Development Center of Computer Software Technology, Shanghai 201112, China)

Abstract: Software testing is an important means to ensure software quality and reliable service. Currently, Web service-based distributed software gets increasing number and its test methods also concern of many people. Manual testing methods of web service are very inefficient due to Web Service's distribution and diversity, thus continuously improving the Web Services' test automation is necessary. Cloud computing provides a new supporting platform of automated testing featuring low computational cost and strong scalability. In this paper, we analyze the Web Services' test requirements, and firstly propose a technical framework of test automation for Web Service interface based on cloud environment. What's more, we analyze the key technologies of atomic Web Service and composite web services test in the proposed framework, and we have developed a prototype system for automated test based on CloudStack platforms. Finally, the experimental results show that the proposed cloud-based automated test scheme for web service is feasible and greatly improves the testing efficiency.

Keywords: cloud computing; automatic test; Web Service; function test

0 引言

分布式软件系统由于在 Internet 应用方面有特别的优势,长期以来受到人们的广泛关注。Web Service 是针对因特网上分布式计算提出的一种基于开放标准、松散耦合及跨平台的新型软件构件[1],随着电子商务等大型分布式系统的发展,Web Service 已经普遍应用在分布式系统的接口方面。

如何确保 Web Service 的服务质量是当前工程领域关注的焦点问题。由于 Web Service 技术的复杂性、应用部署的分布性以及在线运行形态的多变瞬时性等特点,给传统的系统功能测试技术提出了新的挑战,特别是实现分布式的组合 Web Service 测试,因为其需要对测试资源和环境进行快速构建。

云计算[2]有着资源自动生成、弹性分配等特性,结合云平台的特性恰好为 Web Service 的复杂自动化测试提供了新的平台支撑,且提高测试效率、降低测试成本。本文给出了基于云环境的 Web Service 接口自动化测试的技术框架,并分别针对原子 Web Service、组合 Web Service 测试的实现给出可行且有效的测试方法。

收稿日期:2014-11-16; **修回日期**:2015-01-06。
基金项目:上海市教育委员会科研创新项目(12ZZ060);上海市科委启明星项目(12QB1402300)。
作者简介:殷 琦(1988-),男,安徽枞阳县,学生/硕士,主要从事云计算、软件测试方向的研究。

1 相关工作

目前国内外对 Web Service 的测试研究取得了初步的成果。文献[3]提出了基于形式化模型树的原子 Web Service 测试用例生成方法,该方法先将 WSDL[4]文档解析成一棵树,然后根据树中的约束条件,按照等价类的方法生成测试数据。该方法虽然能够生成测试数据,但 WSDL 中对参数、返回值 XML Schema[5]结构的约束较少,产生的数据有限。文献[6]提出了基于合约的测试数据自动生成的方法,该方法在服务提供者和使用者之间增加合约,根据等价类和边界值分析方法,生成测试数据。该方法增加了合约的负担,实际应用困难。针对组合 Web Service 的测试主要有基于数据流的分析和基于变异的测试方法。文献[7]提出了基于 BPEL[8]的组合 Web Service 的数据流分析测试方法,根据 BPEL 内部的数据流动,产生满足既定测试标准的测试路径。但该方法只是静态分析,并没有执行 BPEL,所以达不到可靠性测试要求。文献[9]提出了测试数据选择方法,此方法采用接口和路径变异算子生成变异体,根据初始测试数据能否杀死变异体来筛选测试数据。但该方法中变异体模拟的错误不够真实,且大量无效的变异体部署测试,过程耗时,可操作性差。文献

[10] 详细列举出了 BPEL 文档可行的 26 种变异算子，很好的模拟了真实的错误。目前，以上的所有方法都是在物理机上进行测试的，还没有在云环境中进行的。

近些年来，随着对云计算的深入研究，基于云平台的软件测试作为一种服务被提出，即 TaaS [11]（测试即服务），用户不再需要准备测试环境，可以方便的按需获得自动化测试服务。通过云测试系统，用户只需要通过浏览器提交测试任务，测试云将自动进行资源调度 [12] 分配，把测试任务分配到云端执行，最终将测试结果反馈给用户。云测试帮助用户节省巨大的硬件、软件以及人力成本，并为复杂多变的测试环境提供支持。

本文在上述工作的基础上，首先，针对原子 Web Service 提出了基于云环境的自动化测试方法。该方法解析 WSDL 文档，让用户补充约束条件，然后根据等价类边界值方法生成测试用例，并在云环境中进行自动化并行测试，从而提高测试效率。其次，针对组合 Web Service 测试，提出了改进的基于 BPEL 变异的测试数据选择方法，并结合云环境的弹性部署进行测试。该方法能够筛选出覆盖变异体的最终测试数据，然后再利用这些测试数据对组合 Web Service 进行测试。通过弹性云，自动化部署测试，使该方法得到了应用并提高了 Web Service 的测试效率。

本文第 2 节给出基于云平台的 Web Service 测试原型系统的架构；第 3 节介绍了原子 Web Service 的自动化测试流程和方法；第 4 节介绍了组合 Web Service 的自动化测试流程和方法；第 5 节介绍了在测试原型系统上的实验以及实验结果；最后，第 6 节给出了本文的总结与展望。

2 基于云环境的 Web Service 自动化测试架构

本文基于云环境的 Web Service 测试系统，使用了分层架构，如图 1 所示。云测试环境主要分为五层：用户接口层、测试用例生成层、任务管理层、云管理层和任务执行层。

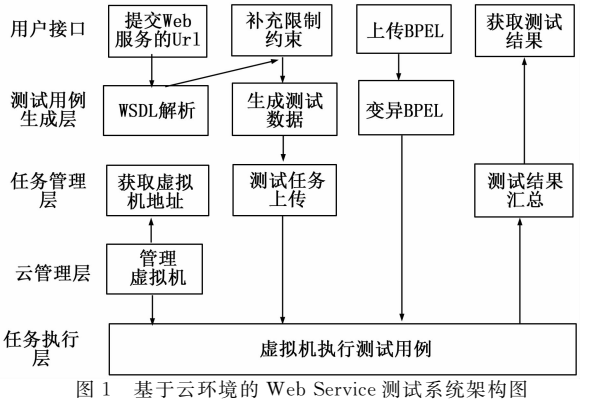

图 1 基于云环境的 Web Service 测试系统架构图

用户接口层：实现了 Web Service 地址的提交，上传测试文档，用户补充约束，查看测试结果等功能。

测试用例生成层：实现了 WSDL 文档的解析，获取接口信息和参数约束，根据约束条件自动生成大量测试用例，以及对 BPEL 进行充分变异。

任务管理层：获取虚拟机地址，将测试任务传输到虚拟机中。

云管理层：选择镜像创建虚拟机。测试任务结束后销毁虚拟机。

任务执行层：负责测试用例的具体执行，并记录测试结果。

针对是原子服务和组合服务，采取不同的方法，图 2 所示为 Web Service 测试流程。

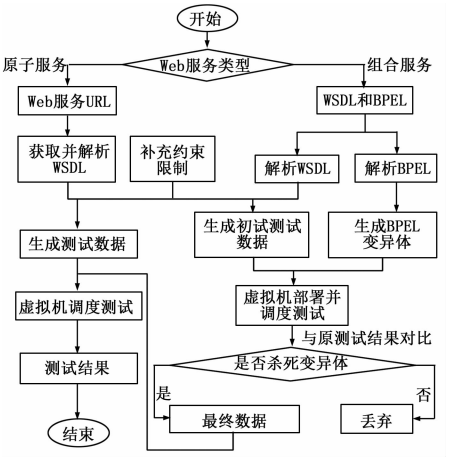

图 2 Web Service 测试流程

3 原子 Web Service 测试

针对原子 Web Service 测试，本文以文献 [4] 中的方法为基础，并扩展到云环境中实现。首先通过 Web Service 的 URL 地址获取 WSDL 文档并采用 DOM 解析得到接口信息，然后获取用户补充的约束条件，如表 1，通过等价类和边界值的方法，自动生成测试用例。这些测试用例会被任务管理层分配到不同的虚拟机中执行，而测试结果将存入数据库，最终通过浏览器向用户返回汇总的测试结果。

表 1 数据约束限制

类型	约束
范围	minInclusive、maxInclusive、minExclusive、maxExclusive
精度	fractionDigits、totalDigits
长度	length、minLength、maxLength
正则	pattern
枚举	enumeration
空白字符	whiteSpace

3.1 解析 WSDL

通过 Web Service 的 URL 获取 WSDL 的文档，使用 DOM 解析 WSDL，获取 Web Service 的所有接口信息，包括接口名、接口参数、返回值等。然后展示给用户，让用户补充每个接口的参数、返回值的约束限制，如 int 类型参数，能提供的约束有 maxInclusive、maxExclusive、minInclusive、minExclusive。

3.2 自动生成测试用例

生成测试用例必然不可缺少测试数据的生成，下面给出测试数据的生成方法。

定义 1：一个 Web Service 的测试信息是一个三元组集合 $W=\{<O_i, R_i, TD_i>1\leqslant i\leqslant n\}$；其中 O_i 为该 *Web Service* 的第 i 个方法；$R_i=<R_{param}, R_{return}>$ 是 O_i 的约束限制，由参数 R_{param} 和 R_{return} 返回值组成；TD_i 为 O_i 的有效测试数据。

R_{param} 定义了方法各个参数应该满足的条件，我们将参数约束转化成 $R_{param}=0\leqslant\hat{j}\leqslant mR_{paramj}$ 合取范式，m 为参数的个数，每个合取项 R_{paramj} 是对方法的单个参数进行约束的析取范

式，形式为 $R_{paramj}=1\leqslant k\leqslant l\ (p\theta_k Value_k)$，其中 p 为参数名称，θ_k 是表 1 中的限制约束，$Value_k$ 是对应 θ_k 约束的常量值。

p 的数据类型以及约束条件确定了它的取值区间，将参数的取值区间映射到输入域上，即针对方法 O_i 的每个参数 p，根据其约束条件，先采用等价类的方法，将 p 的输入域划分成有效等价类和无效等价类。从不同等价类中随机生成初始测试数据集；然后以边界值分析方法作为其补充，选取正好等于、略大于或略小于等价类边界的值加入初始测试数据集。测试数据的生成算法见算法 1 。

算法 1：测试数据的生成输入：p 的数据类型 Datatype，以及含 P 的限制约束条件 R_{paramj}

输出：P 的测试数据集 $TestData_p$

DataRegion＝Initial(DataType)

ValidDataRegion

＝DataRegion

$\bigcap$GetValidRegion(R_{paramj})

InValidDataRegion

＝DataRegion$\bigcap$GetInValidDataRegion(R_{paramj})

$TestData_p=\psi$

$TestData_p$

$=TestData_p$

$\bigcup$GenerateData(ValidRegion，Num)

$\bigcup$GenerateData(InValidRegion，InNum)

上述算法使用函数定义如下：

Initial（DataType）返回值是计算机能处理的范围。

而 GetValidRegion（R_{paramj}）为获取限制约束内的有效区间；GetInValidDataRegion（R_{paramj}）为获取限制约束外的无效区间。

GenerateData（ValidRegion，Num）用随机法，按照用户预先定义的数量 Num，返回有效测试数据，ValidRegion 代表符合均匀分布的测试数据集合，包括边界值。

GenerateData（InValidRegion，InNum）同样返回的是无效等价类的数据。

依次对 O_i 的所有参数执行上述算法，可生成每个接口的测试数据。

3.3　在云环境中进行测试

将测试用例打包，分发到不同的虚拟机，通过脚本程序，执行测试用例，收集测试结果，存入数据库，最终汇总展示给用户。在虚拟机调度上采用了任务集均分的调度策略，将服务的测试用例局分到启动的 n 个虚拟机中，进行并行测试，提高测试效率。

4　服务组合的自动化测试

在组合 Web Service 方面，如果仅按照原子 Web Service 的测试方法进行测试，不能清晰的知道测试数据的路径覆盖情况。本文采用的变异测试方法先筛选出能够覆盖大部分变异体的测试数据，然后再进行测试。通过对 BPEL 文档进行变异，将变异体自动化部署到虚拟机上，使用算法 1 来产生初始测试数据对变异体进行测试，筛选出能覆盖变异体的最终测试数据，然后使用最终测试数据再按照原子 Web Service 测试方法对服务进行测试。如果在传统的环境中，这种方法需要多台机器或者单台机器多次手工部署 BPEL 变异体。如果变异体较多，那是相当繁杂的工作。使用云环境，我们只需弹性化启动相应环境的虚拟机，然后使用脚本程序，自动化部署在虚拟机上，进行并行化测试。

4.1　BPEL 变异

在传统的变异测试中，变异算子的设计初衷是用于模拟真实的错误，其能够直接影响生成测试用例的质量。由于组合 Web Service 各个原子服务的源代码不可见，传统的变异算子无法使用，而且 BPEL 都是通过操作工具生成的，它可能出现的错误有区别于传统程序。本文参照文献［10］中总结的 26 种变异算子，对 BPEL 文档进行充分变异。

对 BPEL 文档的变异可分为四大类变异算子 Identifier replacement operators（I）、Expression operators（E）、Activity operators（A）和 Exception and Event operators（X）。并进一步将这四大类细分，如表 2 细分成 26 种变异算子，并列举出变异算子的取值个数和可能的取值。

表 2　BPEL 的所有变异算子

算子	描述	个数	可取值
定义变异(I)			
ISV	用另一个同一类型的变量替换某个变量	N	
表达式变异(E)			
EAA	替换数学操作符	5	＋、－、*、div、mod
EEU	删去一元操作符负号	1	
ERR	替换关系操作符	6	＜、＞、＞＝、＜＝、＝、！＝
ELL	替换逻辑操作符	2	and、or
ECC	替换路径操作符	2	/、//
ECN	增加或者减少数字常量，添加或者删除一个数字	4	＋1、－1、adding、removing
EMD	修改时间的表达代替它 0 或初始值的一半	2	0、half
EMF	修改日期表达代替它 0 或初始值的一半	2	0、half
活动变异(A)			
并行			
ACI	改变 createInstance 属性为 no	1	
AFP	用并行的活动替换顺序的 forEach 活动	1	
ASF	用 flow 活动替换 sequence 活动	1	
AIS	改变一个 scope 内 isolate 属性为 no	1	
非并行			
AIE	从一个 if 活动中删除 elseif 或者 else 元素	1	
AWR	用 repeatUntil 代替 while，反之亦然	1	
AJC	活动中删除 joinCondition 属性	1	
ASI	交换两个 sequense 子活动的顺序	1	
APM	pick 活动中删除 onMessage 元素	1	
APA	pick 活动或者事件处理中删除 onAlarm 元素	1	
	异常和事件变异(X)		
XMF	从错误处理中删除 catch 或者 catchAll 属性	1	
XRF	从 reply 活动中删除 falutName 属性	1	
XMC	删除补偿处理定义	1	
XMT	删除终结处理定义	1	
XTF	用 throw 活动替换 faultName 属性	N	
XER	删除 rethrow 活动	1	
XEE	从事件处理里删除 onEvent 元素	1	

4.2 分析 BPEL，生成变异体

这里主要是分析 BPEL 文档，筛选出适合具体 BPEL 文档的变异算子以及变异点。表 3 是 BPEL 文档的部分代码，分析得出此段 BPEL 文档能够进行 ERR，AIE，ECN 三种变异算子，并且每种变异算子有两个变异点。例如，buy ＞ 5000，这里＞就可以选择 ERR 变异算子进行变异，5000 可以选择 ECN 变异算子进行变异。

表 3　BPEL 文档

BPEL 部分代码	
＜if name＝"discount"＞	＜/condition＞
＜condition＞	＜invoke name＝"5off"... /＞
buy ＞ 5000	＜/elseif＞
＜/condition＞	＜else＞
＜invoke name＝"10off"... /＞	＜reply name＝"nooff"... /＞
＜elseif＞	＜/else＞
＜condition＞	＜/if＞
buy ＞ 2500	

从分析出的变异算子中，选择变异算子并按照此种变异算子可能的取值，对变异点进行替换，达到对 BPEL 进行变异的目的。

4.3 测试数据筛选并测试

组合服务的初试测试数据的生成采用与原子服务相同的方法。但是由于需要数据覆盖尽可能多的路径，这里可根据变异体覆盖率调节初试测试数据的密度。

将生成的变异体、初试测试数据以及测试脚本一同上传到 Web 服务器，按照分配策略分配相应的虚拟机，通过脚本语言 Python，自动部署变异的 BPEL，对其进行测试，将测试结果存入数据库，然后与未变异的 BPEL 测试结果进行对比，根据结果是否相同，来筛选测试数据。本文对虚拟机调度采用了任务集均分方法，将不同的变异体以及测试用例均分到不同的虚拟机上，筛选出最终测试数据。最后按照原子 Web Service 的测试方法进行测试。

5 测试实验

5.1 测试步骤

本文测试云环境由 3 台物理机，配置为 i5 四核处理器，8GB 内存，Ubuntu12.04 系统。一台管理节点，两台计算节点。虚拟机配置为双核处理器，2G 内存，Ubuntu12.04 系统。测试工具使用 WebInject [13]

实验一，原子 Web Service 测试：

实验对 Web 服务 AddService、SubService、MulService、DivService 进行测试，其提供了加减乘除功能。

用户接口层的前端界面，填写约束条件；测试用例生成层根据算法 1 对服务中的每个函数生成测试数据。任务管理层，请求云管理层开启虚拟机；云管理层返回虚拟机的 IP 到任务管理层；任务管理层将任务分配到虚拟机。再由任务执行层，执行测试脚本，将结果保存数据库。最后通过用户接口层返回测试结果。

最后，使用同配置物理机做对比实验。

实验二，组合 Web Service 测试：

将实验一中的 4 个服务进行一定的组合成 CaculateService。实验二的基本步骤跟实验一相似，主要区别就是需要将变异体部署到不同的虚拟机，筛选测试数据。

5.2 实验结果与分析

实验一的实验结果如图 3 所示。

由实验一中的 1、2，物理机的测试时间比虚拟机的测试时间要小，这主要原因是虚拟机的使用效率不及物理机；但是由 3、4 可以很明显看出，物理机的执行时间基本上是 2 台虚拟机的两倍。这充分说明只要虚拟机任务分配得当，测试效率会有提高。

表 4　实验一测试性能分析

序号	服务名称	虚拟机数	函数个数	每个参数个数	每个参数测试数据数	时间 1，时间 2	物理机测试时间
1	AddService	2	2	2，3	10	31.346s，298.760s	283.432s
2	SubService	2	2	2，3	16	51.562s，894.476s	750.371s
3	AddService	2	2	3，3	10	710.568s，720.763s	1203.582s
4	SubService	2	2	3，3	16	887.342s，890.192s	1624.151s

实验二的测试性能分析见图 4。实验二的关键在于筛选测试数据。所以这里只列出了筛选测试数据的部分。很明显单台物理机的测试非常耗时，主要是变异体手工部署很低效。而云环境中的测试是自动化完成，并结合虚拟机并行测试，所以效率显著提升。

表 5　实验二测试性能分析

序号	服务名称	虚拟机数	函数个数	变异体数	参数个数	测试数据数	杀死变异体数	变异体覆盖率	总测试时间	物理机测试时间
1	Service_1	5	1	5	2	10	2	10%	189.756s	10 min
2	Service_2	6	1	6	3	12	15	83.33%	415.658s	13 min

6 结束语

本文结合了云计算与自动化测试技术，针对 Web Service 自动化测试时资源支持和测试效率问题，提出解决方案并设计实现了原型系统。实验结果证明，使用自动化云测试平台对 Web Service 进行测试，能有效减少测试任务的执行时间和测试人员的工作量，降低了测试成本，提高了测试质量。本文中基于用户补充约束自动生成测试数据，可能数据还不够全面，可以进一步扩充。另外在虚拟机任务分配调度上只是简单任务集合均分方法进行分配，若有更好的调度算法，将更能体现云测试的效率优势。

参考文献：

[1] Bi－xin Y L L I. Testing Web Services：A Review [J]. Computer Science，2008，9：072.

[2] Tograph B，Morgens Y R. Cloud computing [J]. Communications of the ACM，2008，51 (7).

[3] Ma C，Du C，Zhang T，et al. WSDL－based automated test data generation for web service [A] . Computer Science and Software Engineering，2008 International Conference on [C]. IEEE，2008，2：731－737.

[4] Christensen E，Curbera F，Meredith G，et al. Web services description language (WSDL) 1.1 [J]. 2001.

[5] W3C，“XML Schema Part 1：Structures Second Edition”，W3C Recommendation，28 October 2004，[EB/OL] . http：//www.w3.org/TR/2004/RECxmlschema－1－20041028/.

[6] Ying J，Guomao X，Jinhui S，et al. A method of automated test data generation for web service [J]. Chinese journal of computers，2005，28 (4)：568－577.
[7] Dong W L，Hu J H. Test method for BEPL—based Web service composition based on data flow analysis [J]. Journal of Software，2009，20 (8)：2102—21 12.
[8] Cuif F，Qiao Y，Chang Y. Web Services Rapid Composition Framework Based on BPEL [J]. Computer Engineering，2010，7：094.
[9] Zhang M H，Jiang Y. Method of test data selection for Web services composition based on mutation [J]. Journal of Computer Applications，2011，9：037.
[10] Estero－Botaro A，Palomo－Lozano F，Medina－Bulo I. Mutation operators for WS－BPEL 2.0 [A] .21th International Conference on Software & Systems Engineering and their Applications [C]. 2008.
[11] Gao J，Bai X，Tsai W T. Cloud testing－issues，challenges，needs and practice [J]. Software Engineering：An International Journal，2011，1 (1)：9－23.
[12] 方锦明. 云计算中虚拟资源调度的决策系统 [J]. 计算机测量与控制，2011 (12)：3145－3148.
[13] http：//www. webinject. org/ [EB/OL] .

(上接第 2246 页)

测试优化选择仿真结果表明：所得最优解为 [0，0，0，0，0，0，0，1，0，0，0，0，0，0，0，0，0，0，0，0，0，0，0，0，0，0，0，1，0，1，0，0，1，1，0，1，0，0]，即最优完备测试集为 $\{t_8, t_{26}, t_{28}, t_{31}, t_{32}, t_{34}\}$，其系统故障检测率为 96.86%，故障隔离率为 96.57%，测试代价为 6。3 种算法的优化结果对比如表 1 所示。各算法在优化过程中的平均测试代价收敛曲线如图 1 所示。

从表 1 和图 1 可以看出，在求解全局最优解上，IQEA 和 IDPSO 明显优于 HBPSOGA，并且 IDPSO 在 20 次独立试验中都找到了最优解，成功率为 100%。这主要是因为：IDPSO 和 IQEA 在 HBPSOGA 的基础上，根据故障检测率利用启发式规则可行化粒子，然后根据测试性指标构建罚函数计算粒子适应度，从而加强了算法的求解精度。在收敛速度上，IDPSO 优于其他两种算法，HBPSOGA 虽然在算法前期收敛速度优于 IQEA，但在后期容易陷入早熟。在运行时间上，HBPSOGA 和 IQEA 基本是 IDPSO 的两倍。

通过超外差接收器系统仿真实例结果，表明 IDPSO 在收敛速度，收敛精度和运行时间方面均优于 HBPSOGA 和 IQEA，能够在满足系统测试性指标要求的前提下准确快速的获取全局最优解。

表 1　3 种算法的优化统计结果

	HBPSOGA	IQEA	IDPSO
平均最优测试代价	6.80	6.15	6
平均收敛代数	54.18	81.53	48.26
最好收敛代数	26	39	18
最差收敛代数	84	105	72
平均运行时间/s	2.825	2.481	1.148
成功率/(%)	20	85	100

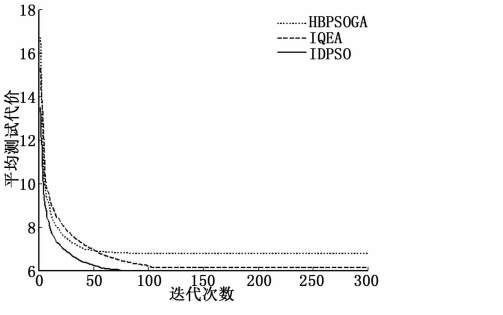

图 1　平均测试代价收敛曲线

4　结论

目前，航空电子设备的性能和规模日益增加，其故障与测试之间的相关性变得日趋复杂，测试优化选择问题也日益突出，提高其优化效率和精度就显得尤为重要。本文根据测试优化选择问题的特点，对 PSO 算法进行了一些改进，提出了一种基于 IDPSO 的测试优化选择方法，该方法的优势体现在：

1）采用混沌搜索方式初始化粒子群的速度和位置，克服了种群初始化的不均匀性，使得初始种群更具有多样性，有利于种群的全局寻优。实验结果表明，相比随机方法，该方法具有较高的初始寻优能力。

2）根据测试性指标采用启发式规则和罚函数方法计算粒子适应度，实验结果表明，该方法能够有效修正越界粒子，使算法收敛到全局最优解。

3）通过采用自适应的惯性权重，实现了粒子群全局搜索，使粒子跳出局部最优解，避免算法陷入早熟。

参考文献：

[1] 邱　静，刘冠军，杨　鹏，等. 装备测试性建模与设计技术 [M]. 北京：科学出版社，2012.
[2] 杨　鹏，邱　静，刘冠军，等. 基于布尔逻辑的测试选择算法 [J]. 测试技术学报，2007，21 (5)：386－390.
[3] Golonek T，Rutkowski J. Genetic－algorithm－based method for optimal analog test points selection [J]. IEEE Trans. on Circuits and Systems II：Express Briefs，2007，54 (2)：117－121.
[4] Pattipati K R，Alexandridis M G. Application of heuristic search and information theory to sequential fault diagnosis [J]. IEEE Transactions On Systems，Man，and Cybernetics，1990，20 (4)：872－887.
[5] 蒋荣华，王厚军，龙　兵. 基于离散粒子群算法的测试选择 [J]. 电子测量与仪器学报，2008，22 (2)：11－15.
[6] 陈希祥，邱　静，刘冠军. 基于混合二进制粒子群－遗传算法的测试优化选择研究 [J]. 仪器仪表学报，2009，30 (8)：1674－1680.
[7] 雷华军，秦开宇. 基于改进量子进化算法的测试优化选择 [J]. 仪器仪表学报，2013，34 (4)：838－843.
[8] 王瑞琪，张承慧，李　珂. 基于改进混沌优化的多目标遗传算法 [J]. 控制与决策，2011，26 (9)：1391－1397.
[9] 朱喜华，李颖晖，李　宁，等. 基于群体早熟程度和非线性周期振荡策略的改进粒子群算法 [J]. 通信学报，2014，35 (2)：182－189.