基于故障定位的测试用例优先排序方法

陈 静[1] 舒 强[2] 谢昊飞[3]
(重庆第二师范学院经济与工商管理学院 重庆 400067)[1] (重庆邮电大学 重庆 400065)[2]
(重庆邮电大学自动化学院 重庆 400065)[3]

摘 要 协议一致性测试是检验被测实现是否与标准协议规范相一致的方法,可确保符合协议的设备或者系统互联与互通。在被测设备调试、升级和修复等过程中,往往需要重新执行所有测试案例,以确保协议一致性测试的完备性。在协议实现的过程中,需要频繁地进行测试和修复,直至被测设备的协议完全符合协议的标准规范。而在每次的回归过程中,没有策略地执行测试案例集中所有的测试案例会增加测试的工作量。只有所有的测试案例执行结束,才能确定测试故障是否被正确修复,或者检测出其他新出现的故障。这导致了某些可以检测到故障的测试案例不能尽早执行,无法将测试重点放在易出错的部分,测试执行开销较大,会影响测试效率。因此在协议一致性测试过程中,如何对庞大的测试案例集进行优化并减少测试成本?在保证测试需求的前提下,使用尽可能少的测试案例尽快检测出系统中存在的故障以提高测试的故障检测率,成为了亟待解决的问题。文中在对现有的测试用例优先排序方法进行研究的基础上,对基于故障定位的测试用例优先排序算法进行了改进,以提高故障检测效率。该方法结合测试需求间的依赖关系、执行序列进行动态调整,对检错概率高的测试案例进行动态选取。在搭建的无线传感器网络的协议一致性测试系统上,对该算法进行了有效性验证。相较于Additional和FTP算法,所提方法的故障检测平均百分比*APFD*和测试效率*TCFD*分别至少提高了9.2%和7.6%。

关键词 协议一致性测试,故障检测,排序方法,效率提升,方法改进
中图法分类号 TP13 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.08.039

Priority Ranking Method of Test Cases Based on Fault Location

CHEN Jing[1] SHU Qiang[2] XIE Hao-fei[3]
(School of Economics and Business Administration, Chongqing University of Education, Chongqing 400067, China)[1]
(Chongqing University of Posts and Telecommunications, Chongqing 400065, China)[2]
(School of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)[3]

Abstract Protocol conformance testing is a method to verify whether the tested implementation is consistent with the standard protocol specification, which can ensure the interconnection and interworking of the equipment or system in accordance with the protocol. In the process of debugging, upgrading and repairing the tested equipment, it is often necessary to re-execute all test cases to ensure the completeness of protocol conformance testing. In the process of protocol implementation, it is necessary to test frequently and repairs this process until the protocol implementation of the tested equipment fully conforms to the protocol standard specification. In each regression process, the unstrategic execution of all test cases in the test case set will increase the workload of the test. Only at the end of all test cases, whether the test failure has been repaired correctly, or if other new failures have been detected, can be determined. As a result, some test cases that can detect faults can not be executed as soon as possible, and the test can not focus on the error-prone parts. The cost of test execution is large, which affects the test efficiency. Therefore, in the process of protocol conformance testing, how to optimize the huge test case set and reduce the test cost. Under the premise of ensuring the test requirements, using as few test cases as possible to detect the faults in the system as soon as possible, and improving the test fault detection rate has become an urgent problem to be solved. In this paper, based on the research of the existing test case priority sorting methods, the test case priority sorting algorithm based on fault location was improved, so as to improve the efficiency of fault detection. Combined with the dependence between test requirements, the dynamic adjustment of sequence is performed, and the test cases with high error detection probability are selected dynamically. The algorithm is verified effectively on the protocol conformance test system of wireless sensor networks. Compared with the Additional and FTP algorithms, its average percentage of fault detection APFD and test cost TCFD increases by at least

到稿日期:2019-04-29 返修日期:2019-06-27

陈 静(1978-),女,硕士,副教授,主要研究方向为计算机应用、控制科学与工程,E-mail:dixuancc@163.com;舒 强(1981-),男,博士,副研究员,主要研究方向为网络化控制技术、计算机应用;谢昊飞(1978-),男,博士,教授,主要研究方向为网络化控制技术、无线传感器网络、协议测试、嵌入式系统。

9.2% and 7.6% respectively.

Keywords Protocol conformance test, Fault detection, Sort method, Efficiency improvement, Method improvement

1 引言

在协议一致性测试过程中,对故障进行有效且地快速的查找与定位是一个繁琐复杂的过程。对于被测设备中隐藏的故障,测试人员通常是执行设计好的测试用例,根据输出的测试结果确定有错误的地方。这种方法需要所有测试用例执行完成后才可以查找出被测设备中存在的故障,若需要测试的案例数较多,则这一过程非常耗时。为了降低测试的执行成本,研究人员开始寻找提高故障查找效率的方法。

Jiang 等[1]研究了基于连续集成(Continuous Integration)的故障定位技术,以降低测试与调试成本。其研究成果表明,若对 60%的测试用例进行优先排序,则可以显著提升故障定位的效率。Jones 等[2]统计了测试需求被执行成功或执行失败的测试用例覆盖的情况,并得出需求的可疑度,从而进行故障定位。Perelman 等[3]研究了基于动态基本块的故障定位方法,通过实验表明动态基本块的数量会影响故障定位的精度。Hao 等[4]研究了基于模糊集理论的故障定位技术,消除了相似测试用例对故障定位精度的影响。Zhen 等[5]通过保留小部分冗余测试用例,保证了各个需求在测试过程中被覆盖次数的均衡,实验结果表明该方法可以提高故障定位的有效性。Masri 等[6]研究了使用散点图识别偶然性正确的测试用例,提高了故障检测效率。Miao 等[7]通过 K-means 算法对需求覆盖信息进行了聚类,通过识别偶然性正确的测试用例,提高了故障定位效率。

然而在传统的协议一致性测试过程中,测试用例集过大或者测试资源有限时,全面执行所有测试用例将导致某些检测到故障的测试用例较晚执行,严重影响测试效率。主要原因如下:

(1)现有的测试用例优先排序方法排序准则单一,不能从多方面综合衡量测试用例的优先级,故障检测效率有待提高;

(2)在协议一致性测试研究中,测试需求之间存在着依赖关系,测试需求的覆盖满足顺序也变得至关重要,因此随机的测试用例执行策略会影响测试效率;

(3)现有的协议一致性测试软件很少在测试执行前对测试用例的执行序列进行优化,因此测试用例优先排序技术在协议一致性测试系统中的实现问题亟待解决。

为此,本文将对基于故障定位的测试用例优先排序算法进行改进。在测试用例优先排序过程中,该方法首先通过历史执行记录以及测试需求间的依赖关系计算测试需求重要度,从而得出测试用例重要度,进而对初始执行序列进行优化;然后在测试过程中根据测试的执行情况对未执行测试用例进行优先级动态调整,对检错概率高的测试用例进行动态选取,达到了提高故障检测效率的目的。

2 基于故障定位的测试用例优先排序算法

目前,Tarantula 故障定位技术是此领域研究的重点。Tarantula 故障定位技术从执行历史信息中统计了测试用例的执行情况[8]。其理论基础是,在与一个测试需求具有覆盖映射的测试用例中,执行失败的测试用例数多于执行成功的测试用例数,那么这个测试需求存在故障的可能性更大。Tarantula 故障定位方法使用了测试用例和测试需求之间的覆盖映射信息以及测试执行结果信息。对于每个需求,使用需求的可疑度来衡量其出现故障的程度值。根据测试需求与测试用例之间的覆盖映射关系以及测试用例的执行情况,需求 r_j 的可疑度值计算方式如下:

$$\tau(r_j)=\frac{\frac{fail(r_j)}{totalfail}}{\frac{pass(r_j)}{totalpass}+\frac{fail(r_j)}{totalfail}} \tag{1}$$

其中,$fail(r_j)$和 $pass(r_j)$表示与需求 r_j 具有覆盖映射关系的案例执行失败和执行成功的次数,$totalpass$ 表示测试通过的测试用例总数,$totalfail$ 表示测试失败的测试用例总数。

文献[9]提出了一种基于故障定位的测试用例优先排序方法(Fault Location Priorization,FLP),此方法假设在测试执行过程中,测试执行序列中第 i 个测试用例 t_i 执行失败,T_{i-1} 为已执行过的测试用例集$\{t_1,\cdots,t_{i-1}\}$,$T_i=T_{i-1}\cup\{t_i\}$,$\tau(r_j\mid T_i)$表示需求 r_j 在测试用例集 T_i 执行之后的可疑值,则需求 r_j 存在故障的近似概率 $P_{T_i}(B(r_j))$的计算公式如下:

$$P_{T_i}(B(r_j))=\frac{\tau(r_j\mid T_i)}{\sum_{j=1}^{m}\tau(r_j\mid T_i)} \tag{2}$$

为了把需求可疑值规范化为概率分布,本文使用先验信息熵来表示故障存在的信息:

$$H_{T_i}(r)=-\sum_{j=1}^{m}P_{T_i}(B(r_j))*\log P_{T_i}(B(r_j)) \tag{3}$$

若执行完测试用例集 T_N 时信息熵值 $H_{T_N}=0$,则概率 $P(B(r'))=1$,即需求 r'中存在故障。H_{T_N} 值越小,表明此时执行完测试用例集 T_N 时检测出故障的概率越大。因此此方法的目的就是,在测试用例执行失败后,调整测试用例执行序列,使得当前信息熵 H_{T_N} 值为最小,提高故障检测的有效性。

3 改进的优化算法

3.1 改进算法的思路

在协议一致性测试过程中,被测设备的测试故障被修复过后,需要进行回归测试,以确保被测协议实现了由标准协议规范所规定的各种功能。每一轮的回归测试结束后,根据检测到的故障对被测设备进行修复与调试(即故障定位和修复)。在被测设备的迭代测试过程中,每次故障修复后,不仅要确定故障已经完全修复,还要确保其他模块没有引入信息故障。在测试过程中,怎样确定测试用例的执行序列并快速定位故障,成为了需要解决的问题。而已执行的回归测试历史信息对后续的测试用例执行序列具有一定的参考价值,例如历史测试中测试需求是否得到满足,测试用例执行情况是否稳定等。此外,一些测试需求的执行依赖于其他测试需求,只有某些测试需求在其他测试需求得到满足的情况下才能测试通过[10]。

基于故障定位的测试用例优先排序方法使用测试用例对需求的覆盖信息以及测试用例的执行情况作为排序依据,忽

略了历史执行信息对后续回归测试过程的参考价值。另外,此方法在测试执行开始前没有对测试用例的执行序列进行初始优化,虽然在测试过程中检测到故障时会对未执行测试用例的执行序列进行调整,但是执行开始前采用的是随机测试策略。在测试开始前对初始测试用例的执行序列进行优化,可以在一定程度上提高故障检测效率。

本文在基于故障定位的测试用例优先排序方法的基础上,对测试执行前的初始测试用例执行序列进行了优化。首先,通过历史记录分析以及需求设计文档对测试需求重要度进行分析,以对高风险、易出错的测试需求分配更多的测试资源;然后,通过测试需求与测试用例之间的覆盖映射关系得出测试用例重要度,并据此生成初始测试用例执行序列;接着,在测试过程检测到故障后,收集执行信息,通过基于故障定位的方法选出下一个要执行的测试用例,并动态调整未执行测试用例的优先级。

改进算法的详细流程如图 1 所示。

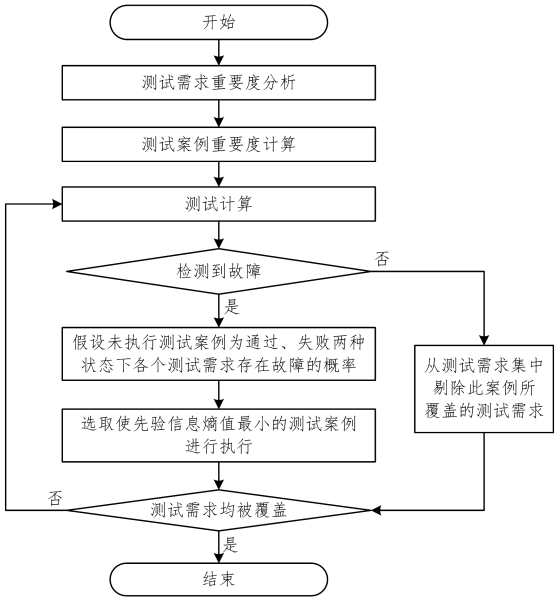

图 1　改进算法的流程

Fig. 1　Process of improved algorithm

3.2　算法设计

本算法在计算测试用例优先级之前,对需求重要度进行了计算,进而利用测试需求与测试用例之间的覆盖映射关系计算出与测试需求相关联的测试用例的优先级,然后在执行过程中根据执行情况对未执行的测试用例进行优先级调整,从而提高故障定位的准确性。

(1)测试需求的重要度计算

本文确定影响测试需求重要度的因素有:需求稳定性 *RSR*,需求依赖深度 *RDD* 和需求相关联的测试用例数量 *RN*。

Step1　需求稳定性 *RSR* 的计算

一些被测需求可能存在风险,在回归测试中应加以重视,检测修改后的被测设备版本存在的故障是否已被移除。因此,在每次测试结束后,应记录所有测试用例的执行情况,并标记出未通过的测试需求。需求稳定性 *RSR* 的计算式如下:

$$RSR_j=\frac{rs_{j,k}}{\sum_{j=0}^{m} rs_{j,k}} \tag{4}$$

其中,m 代表测试需求集中测试需求的个数;$rs_{j,k}$ 表示在同一被测设备上近 k 次的回归测试过程中,测试需求 r_j 测试不通过的次数;RSR_j 表示在近 k 次回归测试过程中,测试需求 r_j 不稳定的次数在所有测试需求不稳定执行次数中的比重。RSR_j 值越大,表明测试需求 r_j 的功能越不稳定,需要在后续回归测试中加以重视,以提高其优先级。

Step2　需求依赖深度 *RDD* 的计算

协议一致性测试中,被其他需求依赖的测试需求应该具有更高的测试优先级。本文使用 Floyd 算法从需求依赖结构中计算任意两点之间的路径最大值。需求依赖深度 *RDD* 的计算方法如下:

首先,根据协议开发文档,建立代表测试需求间功能依赖关系的有向无环图 DAG(Directed Acyclic Graph),即 $G=(V,E)$。其中,V 代表测试需求集合,E 代表测试需求间功能依赖关系的有向边集合。如果测试需求 r_q 的执行依赖于测试需求 r_p,则有向边从代表测试需求 r_p 的节点指向代表测试需求 r_q 的节点。

其次,根据 DAG 建立布尔邻接矩阵 $\boldsymbol{M}$,大小为 $m\times m$,$\boldsymbol{M}_{pq}=1$ 表示需求 r_p 依赖于需求 r_q,$\boldsymbol{M}_{pq}=0$ 则表示需求 $\mathrm{r_p}$ 和需求 $\mathrm{r_q}$ 之间不存在直接依赖关系。其中,p 代表行数,q 代表列数,p 和 q 的取值范围为 1～m。

接下来,对矩阵 $\boldsymbol{M}$ 中的每一项进行 $\boldsymbol{M}[p,k]+\boldsymbol{M}[k,p]$ 运算后再与 $\boldsymbol{M}[p,q]$ 做比较,将最大值赋给 $\boldsymbol{M}[p,q]$。其中,k 代表中间节点,取值范围为 $1\sim m$。

最后,得出矩阵 $\boldsymbol{H}$ 第 p 行的最大值,其代表需求 r_p 的依赖深度。

需求依赖深度 *RDD* 计算的相关伪代码如算法 1 所示。

算法 1　需求依赖深度 *RDD* 的计算伪代码

```
Input:矩阵 M(测试需求之间直接依赖关系的布尔邻接矩阵,大小为
      m×m)
Output:矩阵 H(测试要求之间间接依赖深度的整数邻接矩阵,大小为
       m×m)
       H:=M
  for k:=1 to n do
    for i:=1 to n do
      for j:=1 to n do
      H[i,j]:=max(H[i,j],H[i,k]+H[k,j])
    end for
  end for
end for
return H
```

Step3　与需求相关联的测试用例数 *RN* 的计算

设 $R=\{r_1,r_2,\cdots,r_m\}$ 和 $T=\{t_1,t_2,\cdots,t_n\}$ 分别代表包含 m 个测试需求的需求集和包含 n 个测试用例的测试用例集,测试用例和测试需求之间的覆盖关系映射用二进制信息覆盖矩阵 $\boldsymbol{D}$ 来表示,矩阵大小为 $n\times m$,其中:

$$D_{ij}=\begin{cases}1, & t_i \text{ 能够覆盖 } r_j\\ 0, & t_i \text{ 不能够覆盖 } r_j\end{cases} \tag{5}$$

与需求 j 相关联的测试用例数 RN 的计算方式如下:

$$RN_j=\sum_{i=1}^{n} D_{ij} \tag{6}$$

在执行回归测试用例之前,可以根据实际情况赋予 3 个因素不同的权重 FW(Factor Weight),那么测试需求 j 的重

要度 RW 的计算方式如下：

$$RW_j = FW_1 * RSR_j + FW_2 * RDD_j + FW_3 * RN_j \quad (7)$$

其中，权重 FW_1，FW_2，FW_3 分别代表 RSR，RDD，RN 的权重，可根据测试人员的实际需要进行调整，并且 $\sum_{i=1}^{3} FW_i = 1$。

(2)测试用例的重要度计算

根据测试用例和测试需求之间的覆盖关系映射矩阵 $\boldsymbol{D}$，以及测试需求 j 的重要度 RW，得出测试用例的测试重要度 $TITC$(Test Importance of Test Case)的计算公式：

$$TITC_i = \sum_{j=1}^{m} RW_j * D_{ij} \quad (8)$$

根据测试用例的重要度大小，对测试用例进行优先级排序，得到的测试用例执行序列即为测试用例初始执行序列。

(3)执行序列的动态调整

当检测到故障后，须合理地从未执行案例集中选择下一个测试用例，以提高故障检测率。若测试用例 t_{i+1} 执行后信息熵值 $H_{T_{i+1}}$ 较小，则此测试用例检测出故障的可能性较大，那么 t_{i+1} 即为下一个需要选择的测试用例。本文采用了一种改进的测试用例执行序列动态调整方法来提高故障定位的有效性，具体步骤如下。

Step1 当检测到故障时，记录覆盖各个需求的测试用例执行情况。

记 $\{t_1, \cdots, t_{i-1}\}$ 为已执行用例，t_i 为测试失败用例，余下的未执行测试用例序列为 $\{t_{i+1}, \cdots, t_{i+a}, \cdots, t_n\}$。假设测试用例 t_{i+1} 测试通过，则需求 r_j 的可疑值度量表示为 $P_{T_{i+1}}(B(r_j) | \neg F(t_{i+1}))$；假设测试用例 t_{i+a} 测试失败，则需求 r_j 的可疑值度量表示为 $P_{T_{i+1}}(B(r_j) | F(t_{i+1}))$。两种假设情况下，需求 r_j 的可疑度可由式(2)计算。

Step2 分别计算各个需求存在故障的条件概率，计算方式如下：

$$P_{T_{i+1}}(B(r_j)) = P_{T_{i+1}}(r(s_j) | \neg F(t_{i+1})) * (1-\alpha) + P_{T_{i+1}}(B(r_j) | F(t_{i+1})) * \alpha \quad (9)$$

其中，α 表示测试用例 t_{i+1} 测试失败的概率，可根据已执行测试用例的执行情况进行计算：

$$\alpha = P_{T_{i+1}}(F(t_{i+1})) \approx \frac{TF_i}{TP_i + TF_i} \quad (10)$$

其中，TF_i 表示已执行测试用例集 $\{t_1, \cdots, t_i\}$ 中测试失败的用例数，TP_i 表示测试通过的用例数。

Step3 把需求可疑值规范化为概率分布，使用香农熵来表示故障存在的信息：

$$H_{T_{i+1}}(r) = -\sum_{j=1}^{m} P_{T_{i+1}}(B(r_j)) * \log P_{T_{i+1}}(B(r_j)) \quad (11)$$

Step4 根据 Step1 到 Step3，计算其他未执行测试用例在测试通过和测试失败的假设状态下的先验信息熵值，选取使先验信息熵值最小的测试用例作为测试序列的下一个案例。

4 算法测试与验证

4.1 测试环境

本文测试平台的拓扑结构如图 2 所示。被测设备实现了 GB/T 30269.301-2014《信息技术 传感器网络第 301 部分：通信与信息交换：低速无线传感器网络网络层和应用支持子层规范》标准规定的协议，测试路由器协同被测设备共同执行 GB/T 30269.803－2017《信息技术传感器网络第 803 部分：测试：低速无线传感器网络网络层和应用支持子层》规定的测试用例，协议测试软件分发测试命令和收集测试数据。

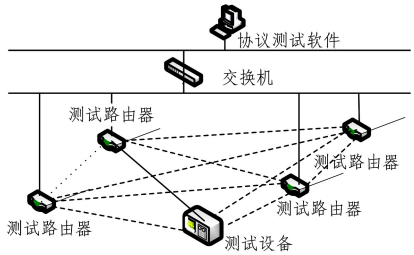

图 2 测试平台的拓扑结构

Fig. 2 Topology of test platform

协议测试软件采用课题小组研发的基于 Web 低速无线传感器网络的测试系统 2.0，装载该软件的 PC 机的 CPU 的主频是 2.4GHz，内存为 2GB，操作系统为 Windows XP，Windows Server2003，Windows Server2005，Windows Server 2012。测试路由器 CPU 采用 TI 公司的 LM3S8962，被测设备的 MCU 是 CC2530。

4.2 算法测试与验证

本文采用故障检测平均百分比 APFD (Average Percentage of Faults Detected)以及检测到所有故障时所需的测试成本 TCFD (Test Cost of All Faults Detected)作为度量指标。根据不同方法所生成的测试用例执行序列的 $APFD$ 值越大，代表其故障检测速率越快；$TCFD$ 值越小，代表检测出所有故障时所消耗的测试资源最少，其优化效果更好。

本文对比 Additional 算法、FLP 算法与本文所提算法的 APFD 和 TCFD 性能指标。图 3 所示为低速无线传感器网络协议中网络层的测试用例在 3 种算法下 $APFD$ 值的对比情况，横坐标为所需测试用例所占的比例，纵坐标为 $APFD$。从图中结果可以得出，本文所提出算法的 $APFD$ 值为 78.5%，FLP 的 $APFD$ 值为 69.3%，Additional 算法的 $APFD$ 值为 59.2%，改进后算法在故障检测效率上比改进前的基于故障的测试用例优先排序方法提高了 9.2%。

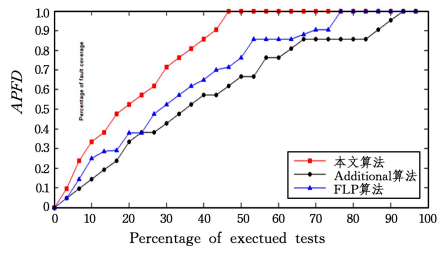

图 3 故障检测速率的对比

Fig. 3 Comparison of fault detection rates

从低速无线传感器网络协议规范中随机选取不同规模的测试需求和测试用例，分别使用 3 种算法进行分析，结果如图 4 所示。图中横轴为测试用例规模的大小，纵轴为算法的 $TCFD$ 值。在不同的测试用例规模下，相比其他两种算法，本文所提出的改进算法的 $TCFD$ 值至少降低了 7.6%(所提算法的平均 $TCFD$ 值为 58.7%，FLP 算法的平均 $TCFD$ 值为

76.3%，Additional 算法的平均 *TCFD* 值为 84.7%），显然本算法具有较好的故障检测效率。

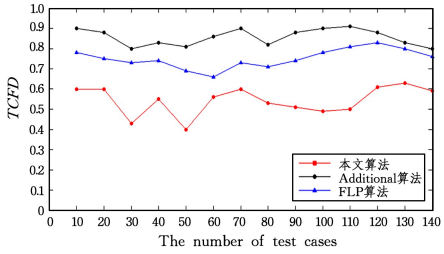

图 4　不同规模下算法的 *TCFD* 值对比

Fig. 4　Comparison of *TCFD* values of algorithms under different scales

重新选用 3 个不同的被测设备，根据测试需求和测试用例规模，对 3 种算法进行 *APFD* 值对比，结果如图 5 所示。本文提出的改进算法的 *APFD* 值要高于其他两种算法，且随着测试用例规模的增大，其 *APFD* 值还略有上升，证明所设计的测试用例优先排序软件有效提升了协议一致性测试的故障检错效率。

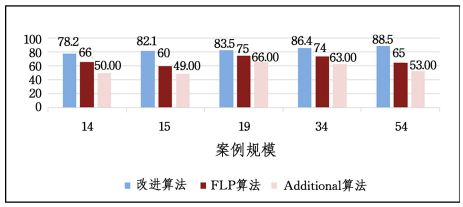

图 5　不同算法的优化效果对比

Fig. 5　Comparison of optimization effects of different algorithms

分析以上结果可发现：

（1）本文所提出的改进的基于故障定位的测试用例优先排序方法相比于 Additional 算法和 FLP 算法，在测试用例相同的情况下，具有较高的故障检测平均百分比，即具有更快的故障检测速率。

（2）在实际的测试用例优先排序过程中，测试用例优先级的排序可能会受到较多因素的影响，如被测协议实现的类型、测试开销和测试环境等，因而可能需要借助于经验判断是否需要以及用何种方式来对测试用例进行优先级的排序。

结束语　本文分析了基于故障定位的测试用例优先排序方法的不足，并设计出了一种改进的测试用例优先排序方法。该方法考虑了多个影响测试需求重要度的因素[11-13]，并结合测试需求间的依赖关系以及需求稳定性计算出测试用例的重要度，生成初始测试用例的执行序列。为提高测试故障查找的准确度，在测试过程中收集测试执行信息，使用基于故障定位的测试用例优先排序方法来查找余下测试用例中检测出故障概率较高的测试用例，以达到提高故障检测速率的目的。最后，通过设计验证系统分析了所提方法的有效性。

但本文在计算测试用例重要度时只考虑了测试需求间的依赖关系以及历史执行等信息，未考虑测试用例的执行时间。未来的工作可以考虑将测试用例的执行成本问题作为测试用例优先排序的准则之一，从多个方面衡量测试用例的重要度。

参 考 文 献

[1] JIANG B, CHAN W K. On the integration of test adequacy, test case prioritization, and statistical fault localization[C]// Proceedings of the 10th International Conference on Quality Software (QSIC). Zhangjiajie: IEEE, 2010: 377-384.

[2] JONES J A, HARROLD M J. Empirical evaluation of the tarantula automatic fault-localization technique[C]// International Conference on Automated Software Engineering. Long Beach, USA: IEEE, 2005: 273-282.

[3] PERELMAN L S, ABBAS W, KOUTSOUKOS X, et al. Sensor placement for fault location identification in water networks: A minimum test cover approach[J]. Automatica, 2016, 72(C): 166-176.

[4] HAO D, ZHANG L, ZANG L, et al. To be optimal or not in test-case prioritization[J]. IEEE Transactions on Software Engineering, 2016, 42(5): 490-505.

[5] ZHEN W, TANG Y H, LIU M, et al. Travelling wave fault location test technique and its applications using high speed travelling wave generator[C]// Power and Energy Engineering Conference. Shanghai: IEEE, 2012: 1-4.

[6] MASRI W, ASSI R A, et al. Prevalence of coincidental correctness and mitigation of its impact on fault localization[J]. Acm Transactions on Software Engineering & Methodology, 2014, 23(1): 1-28.

[7] MIAO Y, CHEN Z, LI S, et al. Identifying coincidental correctness for fault localization by clustering test cases[M]// Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services. Berlin: Springer International Publishing, 2012: 158-165.

[8] BODE F, SACHS F, FRANZ M R. Tarantula peptide inhibits atrial fibrillation[J]. Nature, 2001, 409(6816): 35-36.

[9] YOO S, HARMAN M, CLARK D. Fault localization prioritization: Comparing information-theoretic and coverage-based approaches[J]. Acm Transactions on Software Engineering & Methodology, 2013, 22(3): 1-29.

[10] ZHAO S, FANG Y, LI W, et al. Design and implemention of an emulation node for space network protocol testing[C]// International Conference on Machine Learning and Intelligent Communications. Cham: Springer, 2017: 658-667.

[11] WANG R, TIAN Y L, ZHOU D H, et al. Test-suite Reduction Based on MC/DC in Software Fault Localization[J]. Computer Science, 2015, 42(10): 170-174. (in Chinese)
王瑞，田宇立，周东红，等. 面向故障定位的基于 MC/DC 的测试用例约简方法[J]. 计算机科学，2015，42(10)：170-174.

[12] LI X, ZHANG C, LUO X. Naive Bayesian Applied in Automatic Test Cases Generation[J]. Journal of Chongqing University of Technology(Natural Science), 2012, 26(2): 76-78. (in Chinese)
李欣，张聪，罗宪. 朴素贝叶斯应用于自动化测试用例生成[J]. 重庆理工大学学报(自然科学)，2012，26(2)：76－78.

[13] WU D P, LI Y, WANG R Y. A link failure localization strategy based on knight's tour for Mesh optical network[J]. Journal of Chongqing University of Posts and Telecommunications(Natural Science Edition), 2011, 23(1): 1-5. (in Chinese)
吴大鹏，李阳，王汝言. 基于骑士巡游的 Mesh 光网络链路故障定位策略[J]. 重庆邮电大学学报(自然科学版)，2011，23(1)：1-5.