一种基于频谱信息并结合碰集和遗传算法的缺陷定位方法

周明泉 江国华

(南京航空航天大学计算机科学与技术学院 南京 211100)

摘 要 在软件研制过程中,缺陷定位是一个重要的研究课题。但是,实际软件中的缺陷数量无法被预先判定,且已有的单缺陷定位方法不易使用,已有的多缺陷定位方法存在定位效率不高的问题。基于此,文中对多缺陷定位方法GAMFL进行了研究和改进,提出了基于频谱信息并结合碰集和遗传算法的缺陷定位方法GAHIT。该方法定义了定位基本块,并用其替代语句进行缺陷定位,缩小了搜索范围;在初始种群的构造过程中,提出了采用求解失败用例执行路径碰集的方法,优化了初始种群的生成,并给出了新的适应度函数的计算方法,提高了算法的整体执行效率;最后针对遗传算法的结果,给出了缺陷检查策略,提高了在最优种群中查找缺陷的准确性。实验结果表明,所提方法能够有效处理缺陷数量未知情况下的定位问题,在单缺陷和多缺陷程序中都有较好的定位效果。

关键词 缺陷定位,碰集,遗传算法,缺陷数量未知

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.09.034

New Spectrum-based Fault Localization Method Combining Hitting Set and Genetic Algorithm

ZHOU Ming-quan JIANG Guo-hua

(School of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 211100, China)

Abstract Fault localization is an important research topic in the process of software development. However, the number of faults in the actual software cannot be determined in advance. The available single fault localization technique is not convenient to be used, and the available multi-fault localization technique is of low locating efficiency. This paper studied and improved the multi-fault localization technique GAMFL, and proposed a new spectrum-based fault localization methoid combining hitting set and genetic algorithm(GAHIT). In this method, the basic block for localization is defined and used to replace statements to localize faults, narrowing the search range. In the process of constructing initial population, the method of solving the hitting sets of execution path of failure test cases is presented to optimize the generation of initial population, and a new method for calculating fitness function is also presented to improve the total efficiency of the algorithm. According to the results of genetic algorithm, the fault detecton strategy is presented to improve the accuracy of localizing faults in the optimal group. The experiment results show that the proposed method is effective in solving the problem of localizing programs with unknown number of faults, and has good performance when localizing faults in both single fault programs and multi-fault programs.

Keywords Fault localization, Hitting set, Genetic algorithm, Uncertainty of fault number

在软件开发的生命周期中,软件调试是保证软件质量的重要环节。然而,软件调试通常会花费大量的人力和物力,如何减少软件调试环节的开销成为了研究热点。其中,软件缺陷定位技术是减少调试开销的一种常用方法。

现有的软件缺陷定位技术可以概括为以下几类:1)基于频谱信息的缺陷定位方法主要是统计测试用例的执行信息,根据失败测试用例和成功测试用例在程序语句上的覆盖差异计算每个语句的可疑值,并根据可疑度的高低确定语句的检查次序[1-3],主要包括Tarantula,Jaccard,Ochiai,GP19等方法[4];Shu等[5]还利用程序语句的执行频率替换覆盖信息进行可疑度的计算;Zhang等[6]通过约简测试用例进一步优化了基于频谱的缺陷定位方法。2)基于程序切片的缺陷定位方法主要是将程序切片技术或其改进技术应用于缺陷定位中,识别影响变量的程序语句或者分析程序的执行路径,缩小缺陷定位的范围。例如,Zhang等[7]将动态切片技术应用于Java程序的缺陷定位中。3)基于状态依赖的软件缺陷定位方法主要是将程序控制流图看作有向图,程序的执行是从一个状态转移到另一个状态,根据状态转移的概率确定状态的检查次序。例如,Gong等[8]分别计算成功用例和失败用例下状态之间的转移概率,并通过比较两者之间的差异来决定各个

到稿日期:2017-08-18 返修日期:2017-11-27

周明泉(1993-),男,硕士生,主要研究方向为软件缺陷定位,E-mail:1436693658@qq.com;**江国华**(1963-),男,副教授,主要研究方向为软件工程、软件测试技术、嵌入式系统及应用,E-mail:jianggh@nuaa.edu.cn(通信作者)。

状态的优先检查顺序。4)基于数据挖掘的软件缺陷定位方法主要是构造程序的执行序列,利用关联规则挖掘执行序列与测试用例成功和失败的关联关系,从而定位缺陷位置。例如,Cao 等[9]将动态切片和关联分析结合,通过分析混合谱矩阵确定代码的检查顺序。另外还有一些学者对多缺陷程序的定位方法进行了研究。例如,Lamraoui 等[10]结合可满足准则和基于模型诊断的方法提出了基于准则的多缺陷方法;王赞等[11]将缺陷问题转化为搜索问题,利用遗传算法进行多缺陷位置的搜索(GAMFL)。

在实际开发中,程序的缺陷数量往往是未知的,单缺陷的定位方法在多缺陷程序中进行定位时不易使用,而已有的多缺陷的方法存在定位效率不高或者计算较复杂的问题。本文研究了缺陷数量未知情况下的缺陷定位问题,对现有的多缺陷定位方法 GAMFL 进行了改进,提出了基于频谱信息并结合碰集和遗传算法的缺陷定位方法 GAHIT,其在单缺陷和多缺陷程序中都有较好的定位效果。该方法主要分为以下几步:首先,通过定义定位基本块的概念,将在语句中搜索缺陷优化为在定位基本块中搜索,减小了搜索范围;然后,通过求解失败用例覆盖基本块集合的碰集,猜测包含缺陷的定位基本块的可能组合,优化了初始种群的生成方法,并对 GAMFL 中遗传算法的适应度函数进行了改进,提高了算法的整体执行效率;最后,给出了在遗传算法结果中进行缺陷搜索的策略,提高了缺陷定位的准确性。

1 基本概念

本节给出了基于频谱的软件缺陷定位技术的一些基本概念[12]。

定义 1(测试用例集) 对程序进行测试的用例构成的有序集合,表示为 $T=\{t_1,t_2,\cdots,t_n\}$,其中 t_i 表示第 i 个测试用例,n 表示测试用例数。

定义 2(待测程序集) 程序语句或语句块组成的集合,表示为 $E=\{e_1,e_2,\cdots,e_m\}$,其中 e_i 表示第 i 个程序单元,程序单元可以是语句或语句块。

定义 3(覆盖矩阵) 程序执行所有测试用例的语句覆盖信息组成的集合,表示为 $M=\begin{bmatrix} M_{11} & \cdots & M_{1m} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nm} \end{bmatrix}$,其中 M_{ij} 表示第 i 个测试用例对第 j 个程序单元的覆盖情况。若 $M_{ij}=1$,则表示第 i 个测试用例覆盖第 j 个程序单元;若 $M_{ij}=0$,则表示第 i 个测试用例未覆盖第 j 个程序单元。

定义 4(测试用例结果向量) 每个测试用例失败与成功的结果集合,表示为 $R=\begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}$,其中 r_i 表示测试用例 t_i 的测试结果。若 $r_i=1$,则表示第 i 个测试用例失败;若 $r_i=0$,则表示第 i 个测试用例成功。失败用例数 $N_F=\sum_{i=1}^{n} r_i$,成功用例数 $N_P=n-N_F$。覆盖程序单元 e_i 的失败测试用例数表示为 $a_{11}(e_i)$,$a_{11}(e_i)=\sum_{k=1}^{n}(M_{ki}\times r_k)$;覆盖程序单元 e_i 的成功测试用例数表示为 $a_{10}(e_i)$,$a_{10}(e_i)=\sum_{k=1}^{n}[M_{ki}\times(1-r_k)]$;未覆盖程序单元 e_i 的失败测试用例数表示为 $a_{01}(e_i)$,$a_{01}(e_i)=N_F-a_{11}(e_i)$;未覆盖程序单元 e_i 的成功测试用例数表示为 $a_{00}(e_i)$,$a_{00}(e_i)=N_P-a_{10}(e_i)$。

Tarantula 可疑度系数 S_T 和 Ochiai 可疑度系数 S_O 的计算公式如下:

$$S_T(e_i)=\frac{\frac{a_{11}(e_i)}{N_F}}{\frac{a_{11}(e_i)}{N_F}+\frac{a_{10}(e_i)}{N_P}} \tag{1}$$

$$S_O(e_i)=\frac{a_{11}(e_i)}{\sqrt{N_F\times[a_{11}(e_i)+a_{10}(e_i)]}} \tag{2}$$

下面通过实例来解释以上概念。以下函数代码用于计算 4 个数字中第二大的数,其中包括两个缺陷。

```
1. int findSecond(int a,int b,int c,int d){
2.     int max,min,temp;
3.     if(a<=b){
4.                 max=b;
5.                 min=a;
6.     }else {
7.                 max=a;
8.                 min=b;
9.         }
10.    if(c>max){
11.                temp=max;
12.                max=c;
13.    }else if(c<min){
14.                temp=b;/*缺陷。应为 temp=min */
15.                min=c;
16.    }else{
17.                temp=b;/*缺陷。应为 temp=c */
18.    }
19.    if(d<max&&d>temp){
20.                return d;
21.    }else if(d>max){
22.                return max;
23.    }else {
24.                return temp;
25.    }
26.}
```

根据测试用例的覆盖情况计算各统计量及可疑度,并按照可疑度由高到低进行缺陷检查,如表 1 所列。其中,t_1-t_{16} 为成功测试用例,$t_{17}-t_{24}$ 为失败测试用例,T 表示 Tarantula,O 表示 Ochiai,T* 表示改进的 Tarantula。表中的数据显示,这 3 种方法均能够较快地发现第一个缺陷的位置,但是找出所有缺陷检查的代码占总代码的比值较高。表 1 中的 3 种方法能够有效地应对单缺陷的情况,但是无法有效地处理多缺陷情况。针对实际中无法提前确定代码中缺陷数量的情况,本文提出了 GAHIT 方法,自动猜测可能的缺陷组合,根据猜测组合中的缺陷数量分别计算可疑度并进行缺陷定位。

表 1　findSecond 函数的测试用例覆盖路径及各统计量的计算结果

Table 1　Coverage path information of test cases for findSecond function and calculation results of various statistics

可执行代码行	t_1	t_2	…	t_{16}	t_{17}	t_{18}	…	t_{24}	a_{10}	a_{00}	a_{11}	a_{01}	T	O	T*
3	1	1	…	1	1	1	…	1	16	0	8	0	0.5	0.577	0.500
4	0	0	…	1	1	0	…	1	6	10	6	2	0.667	0.612	0.500
5	0	0	…	1	1	0	…	1	6	10	6	2	0.667	0.612	0.500
7	1	1	…	0	0	1	…	0	10	6	2	6	0.286	0.204	0.179
8	1	1	…	0	0	1	…	0	10	6	2	6	0.286	0.204	0.179
10	1	1	…	1	0	1	…	1	16	0	7	1	0.467	0.516	0.467
11	1	0	…	1	0	0	…	0	8	8	0	8	0	0	0
12	1	0	…	1	0	0	…	0	8	8	0	8	0	0	0
13	0	1	…	0	1	1	…	1	8	8	8	0	0.667	0.707	0.667
14	**0**	**1**	…	**0**	**1**	**0**	…	**1**	**5**	**11**	**3**	**5**	**0.545**	**0.375**	**0.205**
15	0	1	…	0	1	0	…	1	5	11	3	5	0.545	0.375	0.205
17	**0**	**0**	…	**0**	**0**	**1**	…	**0**	**3**	**13**	**5**	**3**	**0.769**	**0.625**	**0.481**
19	1	1	…	1	1	1	…	1	16	0	8	0	0.500	0.577	0.500
20	0	0	…	0	0	1	…	0	4	12	1	7	0.333	0.158	0.083
21	1	1	…	1	1	0	…	1	12	4	7	1	0.538	0.568	0.471
22	0	1	…	0	0	0	…	0	6	10	0	8	0	0	0
24	1	0	…	1	1	0	…	1	6	10	7	1	0.700	0.686	0.613
测试用例结果	0	0	…	0	1	1	…	1							

2　GAHIT 缺陷定位方法

2.1　缺陷组合猜测

基本块是指程序中一个顺序执行的语句序列，其中只有一个入口和一个出口。在基于频谱信息的缺陷定位方法中，同一个基本块中的语句具有相同的覆盖信息，即它们的可疑度相同。在可疑度列表中，同一基本块中的语句无法区分其被检查的优先级。同时，以基本块为获取覆盖信息的基本单元，能够约简频谱矩阵。因此，本文将基本块作为缺陷定位的基本单元。本文研究的对象主要是 C 语言，C 语言中包含了选择结构 switch 和循环机构 for，while 等，因此需要对基本块的定义进行扩展。

定义 5（定位基本块）　在高级语言中，定位程序中连续执行且不会被打断的语句序列为基本块。如图 1 所示，给出 C 语言中各种结构基本块的划分方法。

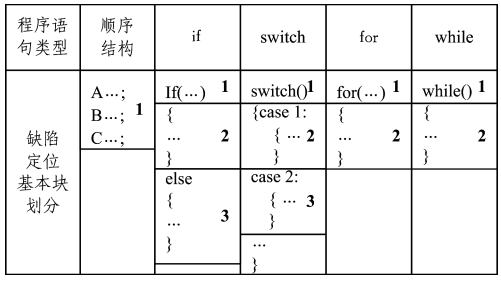

图 1　定位基本块的划分方法

Fig. 1　Method of dividing basic blocks for localization

例如，对于 switch 选择结构的语句，在无确定数据的情况下，switch(…)的执行无法确定 case 是否执行，因此将 switch(…)划分为一个基本块，每个 case 分别划分一个基本块。对于存在程序语句类型嵌套的代码，将该基本块内的代码再按照划分规则进行划分。例如，switch 语句中 case1 内包含 if 结构语句，则按照划分方法再将定位基本块 2 内的代码划分为多个定位基本块。在单元级的缺陷定位中，将整个代码划分为定位基本块，能够有效地合并覆盖矩阵中的重复信息，精简覆盖矩阵。findSecond 函数的基本块划分结果为{{2},{3},{4,5},{7,8},{10},{11,12},{13},{14,15},{17},{19},{20},{21},{22},{24}}，将 17×17 的覆盖矩阵简化为 14×14 的覆盖矩阵。

定义 6（碰集）　对于集合簇 $Z=\{S_1,S_2,\cdots,S_n\}$，若集合 S_H 满足 $S_1\cap S_H\neq\varnothing, S_2\cap S_H\neq\varnothing,\cdots,S_n\cap S_H\neq\varnothing$，则称集合 S_H 是集合簇 Z 的碰集。若 S_H 的任一子集都不是集合簇 Z 的子集，那么 S_H 是集合簇 Z 的最小碰集[13]。例如，集合簇{{1,2,3},{3,4},{2,5}}的最小碰集为{1,4,5},{2,3},{3,5},{2,4}。

定义 7（缺陷组合）　对于一个定位基本块构成的集合 $F=\{f_1,f_2,\cdots,f_n\}$，若所有失败测试用例都至少执行 F 中的一个定位基本块 f_i，那么 F 就是一个缺陷组合。一个包含缺陷的程序存在多个缺陷组合，在所有的缺陷组合中，存在一个只包含所有缺陷的缺陷组合，该缺陷组合被称为 F_{accuracy}。F_{accuracy} 是缺陷定位的准确解，缺陷定位就是逼近 F_{accuracy}。

在测试用例中，失败用例覆盖信息集合簇 $L=\{C_1,C_2,\cdots,C_n\}$，其中，C_i 表示第 i 个失败用例的定位基本块覆盖信息集合。在进行缺陷定位之前，假设存在缺陷组合 F，F 需要能够解释所有的失败测试用例，即每个失败测试用例必须至少覆盖一个 F 中的定位基本块，因此，缺陷组合 F 是失败用例覆盖信息集合簇 L 的碰集。本文将失败用例覆盖信息集合簇的碰集作为缺陷组合的猜测。算法 1 给出了文中求解失败用例覆盖信息集合的碰集的具体步骤，其主要思想是由一个集合构成集合簇，针对集合中的每个元素构建碰集；在集合簇中添加一个新集合，若之前的碰集不是新集合簇的碰集，则对该碰集添加新元素，新元素为属于新集合但不属于上一步集合簇中的元素；重复上述步骤，直到所有的失败测试用例集合都包含在集合簇中。

算法 1　GetHit 算法

输入：N_F，失败用例覆盖信息集合簇 L，定义变量的定位基本块集合 C_d

输出：缺陷组合 F_H

1. i←1，F=Ø，C_{all}=Ø

2. If(i=1)

3. for all c∈ C_i-C_d

4. createhit(c)
5. /* 以 c 构成碰集 F 并保存于 F_H 中 */
6. $C_{all}=C_{all}\cup C_i$
7. Repeat
8. i←i+1
9. for all F∈ F_H
10. if($F\cap C_i=\emptyset$)
11. for all c∈C_i-C_{all}
12. add(c,F)
13. /* 将 c 加入 F 中并将新的 F 保存于 F_H 中 */
14. $C_{all}=C_{all}\cup C_i$
15. until i=N_F
16. Return

2.2 遗传算法搜索最优种群

本文将缺陷定位问题转化为组合优化问题，即搜索可疑度最大的缺陷组合。缺陷定位的最终目的是搜索出缺陷组合的正确解，但搜索到的最优组合不一定是正确解，需要检查人员继续检查其他可疑度大的解。与其他解决组合优化问题的算法相比，遗传算法能够求解出最优种群，为检查人员提供其他可疑度大的可能解。在遗传算法的其他应用中，可能存在编码和解码复杂的问题，而在缺陷定位中，编码和解码较容易实现。因此，本文采用遗传算法来搜索可能的缺陷组合。

2.2.1 染色体编码方式

GAHIT 算法以定位基本块为单位，将整个被测程序表示为一个二进制向量：$C=\{c_2,c_2,\cdots,c_n\}$。其中，n 为被测程序的定位基本块的数量，c_i 表示第 i 个基本块是否存在缺陷。若 $c_i=1$，则在该个体中第 i 个基本块存在缺陷；若 $c_i=0$，则在该个体中第 i 个基本块不存在缺陷。例如，有一个个体 $C=\{0,0,0,0,0,1,1,0,0,0\}$，其表示被测程序包含 10 个基本块，其中第 6 个和第 7 个基本块存在缺陷，其余基本块不存在缺陷。

初始种群的生成方法：本文以缺陷组合生成初始种群，缺陷组合中的元素相对应的数字就是二进制向量 C 对应序号位置设为 1。本文定义初始种群个体数目为 N_p，若求解出的碰集数小于 N_P，则随机重复相差数目的个体；若碰集数大于 N_p，则随机选择 N_p 个碰集构成初始种群。

2.2.2 适应度值函数

定位基本块之间的关联关系 R_f 是指各个定位基本块同时执行的可能性，本文提出的计算方法为：

$$R_f=\begin{cases}\dfrac{\sum_{i=2}^{e}(i\times N_i)}{\dfrac{(e+2)\times(e-1)/2}{N_f}}+1, & e\geqslant 2\\ 1, & e=1\end{cases} \tag{3}$$

$$e=\sum_{c_i\in C}c_i \tag{4}$$

其中，e 表示个体中包含定位基本块的个数，N_i 表示同时执行 i 个定位基本块的失败用例数，N_f 表示失败用例的总数。若 $e=1$，即 $R_f=1$，则表示该个体中只包含一个定位基本块。若 $e\geqslant 2$，则该个体中的定位基本块数量至少有两个，R_f 值越大，则该个体中定位基本块同时执行的可能性就越大。若缺陷在代码调试阶段未被发现，但在之后的测试中暴露，这种情况下，代码的主要缺陷分布在分支语句或者边界值中。因此，在多缺陷的情况下，若各个缺陷不在同一个定位基本块中，那么缺陷应该分布在不同的分支或者边界值中，本文认为在大部分定位中多个包含缺陷的定位基本块同时执行的可能性不大。

UT 可疑度系数衡量了各个个体的可疑度值，其计算公式为：

$$UT(C)=\frac{1}{R_f}\times\left(\frac{\varphi(C)}{N_f}\right)^k\times\left(\frac{\sum T}{e}\right) \tag{5}$$

其中，k 为常数，$\varphi(C)$ 表示个体 C 解释失败测试用例的能力。若个体 C 与 n 个失败用例覆盖路径存在交集，则 $\varphi(C)=n$。对于一个个体，它必须能够解释所有的失败用例，因此将 $\left(\frac{\varphi(C)}{N_f}\right)^k$ 作为衡量个体适应度值的重要系数，若该个体存在无法解释的失败用例，则其适应度值将较低。$\sum T/e$ 表示个体中定位基本块 Tarantula 可疑度值的平均值。特殊地，当只有一个缺陷时，该适应度函数即为 Tarantula 可疑度值。

2.2.3 遗传操作算子

遗传算法的操作算子包括选择算子、交叉算子和变异算子。选择算子被用来选择进行交叉或变异操作的个体。本文采用轮盘赌选择法，根据个体的适应度值计算其占剩余种群适应度值之和的比例，适应值越大的个体被选择的概率越大。交叉算子是指在选择的个体部分，染色体编码交换产生新个体的过程。变异算子是指个体中的每一个基因按照变异概率 P_m 成为变异点，并对变异的基因进行反转。将新产生的种群进行适应度值排序，选择排序前 N_P 的个体构成新的种群[14-15]。

2.3 缺陷查找策略

经过遗传算法搜索后，得到最优种群，将最优种群的个体按照适应度函数值由高到低依次排序。按照个体可疑度由高到低依次进行检查，若个体中的定位基本块存在缺陷，则记录；若个体中的定位基本块不存在缺陷，则包含该定位基本块的个体的可疑度降低。动态调整个体的可疑度排序，将包含该无缺陷的定位基本块的个体检查次序后移；检查更新排序后的缺陷组合，直到找到所有的缺陷。对包含无缺陷基本块的个体检查次序进行后移，能够更快检查出含缺陷的基本块，提高整体效率。缺陷查找策略算法如算法 2 所示。

算法 2 FLocate 算法

输入：最优种群中互异的解 P=$\{F_1,F_2,\cdots,F_n\}$，检查人员定位基本块的检查结果 Result

输出：存在缺陷的定位基本块 FaultSet

1. Rate(P)//P 按可疑度由高到低排序
2. RightSet=∅，FaultSet=∅，erkey=0
3. //erkey 表示测试用例是否全部通过
4. while(erkey==0) //程序存在缺陷
5. show(F_{first})//显示排序第一的解
6. for all f∈F_{first}
7. if(f. Result is fault)
8. FaultSet=FaultSet∪{f}
9. else
10. FaultSet=FaultSet∪{f}

```
11. P=P−{F_first}
12. for all F∈P
13. if (F⋂RightSet!=∅)
14. down(F)
15. //将包含无缺陷基本块的解移至序列尾端，优先检查其他解
16. cin≫erkey
17. //若测试用例全通过，置 1，结束循环
18. Return FaultSet
```

与其他定位方法相比，本文的缺陷定位策略利用了已经检查过的定位基本块的信息，动态地调整检查的次序。该算法在最坏情况下需要检查所有的解，时间复杂度为 $T(n)=O(n^2)$；最好情况为最优解是正确解，时间复杂度为 $T(n)=O(n)$。因此，缺陷定位策略算法的时间复杂度为 $T(n)=O(n^2)$。

3　实验

本文对西门子套件中的程序进行了实验，包括 print_tokens，print_tokens2，tot_info，replace 等。通过对上述程序进行缺陷注入，控制代码中的缺陷为 1 个、2 个或 2 个以上。根据定位基本块的划分，对需定位的函数进行插桩，获取测试用例的路径信息。

本文采用的评价指标为 $EXAM_F$ 和 $EXAM_L$，前者表示检查到第一个缺陷时已检查代码占总代码的百分比，后者表示检查出所有缺陷时已检查代码占总代码的百分比。

实验的相关参数如表 2 所列，其中 N 表示遗传算法的迭代次数，N_p 表示种群的大小，N_{select} 表示选择算子选择的个体数，P_c 表示基因交叉的概率，P_m 表示基因发生变异的概率。

表 2　实验参数

Table 2　Experiment parameters

N	N_p	N_{select}	P_c	P_m	k
200	200	80	0.7	0.05	5

对单缺陷的代码进行缺陷定位实验，结果如图 2 所示。横坐标表示 Tarantula，Ochiai，GAMFL[9]，GAHIT 4 种缺陷定位方法，纵坐标表示检查出缺陷位置时的检测代码占总代码的比例，图中数据为多个实验对象结果的平均值。实验结果表明，这 4 种方法在单缺陷的情况下效果接近，定位都比较准确。

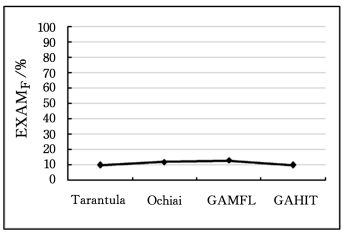

图 2　单缺陷定位结果

Fig. 2　Experimental result of single fault localization

对多缺陷的代码进行缺陷定位实验，结果如图 3、图 4 所示。图 3 为检查出第一个缺陷时的检查代码比，图 4 为检查出所有缺陷时的检查代码比，图中数据为多个程序定位结果的平均值。实验结果表明，4 种方法在发现第一个缺陷的效率方面相差不大；但在检查出所有缺陷方面，本文方法明显优于 Tarantula 和 Ochiai，比 GAMFL 略有提升。在算法效率方面，因为定位基本块的划分，染色体编码的长度缩短了；另外，定位基本块内一个或多个语句有错的情况均包含于同一染色体个体，减少了这部分个体的重复生成，与 GAMFL 方法相比，减少了迭代次数，提高了搜索效率。

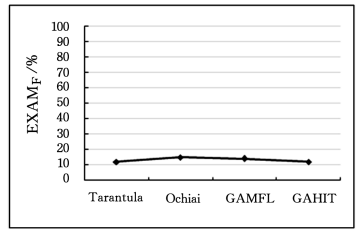

图 3　多缺陷情况下 $EXAM_F$ 指标的比较结果

Fig. 3　Comparison of $EXAM_F$ under condition of multiple faults

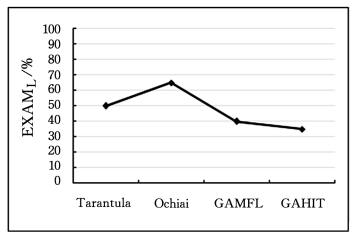

图 4　多缺陷情况下 $EXAM_L$ 指标的比较结果

Fig. 4　Comparison of $EXAM_L$ under condition of multiple faults

4 种定位方法的平均时间消耗如表 3 所列。其中，Tarantula 和 Ochiai 方法只计算了各个语句的可疑度，时间较短，但是这两种方法在多缺陷的程序中定位效果不佳，人为检查缺陷位置的时间较长。与 GAMFL 相比，GAHIT 算法的运行时间略微缩短，因此在算法耗时相近的条件下，本文所提方法的代码检查比较小，能够提高人为检查缺陷阶段的效率。

表 3　执行时间的比较

Table 3　Comparison of execution time

（单位：s）

实验程序	Tarantula	Ochiai	GAMFL	GAHIT
print_tokens	2.01	1.98	11.12	10.23
print_tokens2	1.99	1.80	10.88	9.73
tot_info	0.98	1.05	7.25	8.03
replace	2.95	3.01	18.85	15.75

结束语　本文提出一种基于频谱信息并结合碰集和遗传算法的缺陷定位方法，通过定位基本块的划分，精简了覆盖矩阵；提出采用求解失败用例覆盖信息碰集的方法，猜测缺陷组合，优化了初始种群的生成；给出了新的适应度函数，提高了算法的整体执行效率；最后给出在遗传算法结果中搜索缺陷的策略，提高了缺陷定位的准确性。本文结果表明，GAHIT 方法能够自动地对单缺陷和多缺陷的情况分别计算可疑度值，在单缺陷情况下，定位效果与 Tarantula 一致，在多缺陷的情况下，优于其他方法。

本文方法仅利用了测试用例的覆盖信息，未来可以结合控制依赖和数据依赖等信息来进一步提高缺陷定位的精确性，同时在适应度值函数上进行进一步的验证和改进。另外，本文方法的通用性需要在大型项目上进行进一步验证。

参 考 文 献

[1] LIU X,LIU Y,LI Z,et al. Fault Classification Oriented Spectrum Based Fault Localization[C]// IEEE,Computer Software and Applications Conference. IEEE Computer Society,2017:256-261.

[2] GETZNER E,HOFER B,WOTAWA F. Improving Spectrum-Based Fault Localization for Spreadsheet Debugging[C]// IEEE International Conference on Software Quality,Reliability and Security. IEEE,2017.

[3] CHEN X,JU X L,WEN W Z,et al. Review of dynamic fault localization approaches based on program spectrum[J]. Journal of Software,2015,26(2):390-412. (in Chinese)
陈翔,鞠小林,文万志,等. 基于程序频谱的动态缺陷定位方法研究[J]. 软件学报,2015,26(2):390-412.

[4] KIM J,PARK J,LEE E. A New Spectrum-based Fault Localization With the Technique of Test Case Optimization[J]. Journal of Information Science & Engineering,2016,32(1):177-196.

[5] SHU T,YE T,DING Z,et al. Fault localization based on statement frequency[J]. Information Sciences,2016,360(C):43-56.

[6] ZHANG X,WANG Z,ZHANG W,et al. Spectrum-Based Fault Localization Method with Test Case Reduction[C]// IEEE,Computer Software and Applications Conference. IEEE Computer Society,2015:548-549.

[7] ZHANG P,MAO X,LEI Y,et al. Fault localization based on dynamic slicing via JSlice for Java programs[C]// IEEE International Conference on Software Engineering and Service Science. IEEE,2014:565-568.

[8] GONG D,SU X,WANG T,et al. State dependency probabilistic model for fault localization[J]. Information & Software Technology,2015,57(1):430-445.

[9] CAO H L,JIANG S J,JU X L,et al. Fault Localization Based on Dynamic Slicing and Association Analysis[J]. Chinese Journal of Computers,2015,38(11):2188-2202. (in Chinese)
曹鹤玲,姜淑娟,鞠小林,等. 基于动态切片和关联分析的错误定位方法[J]. 计算机学报,2015,38(11):2188-2202.

[10] LAMRAOUI S M,NAKAJIMA S. A Formula-Based Approach for Automatic Fault Localization of Imperative Programs[M]// Formal Methods and Software Engineering. Springer International Publishing,2014:88-98.

[11] WANG Z,FAN X Y,ZOU Y G,et al. Genetic algorithm based multiple faults localization technique[J]. Journal of Software,2016,27(4):879-900. (in Chinese)
王赞,樊向宇,邹雨果,等. 一种基于遗传算法的多缺陷定位方法[J]. 软件学报,2016,27(4):879-900.

[12] SOUZA H A,CHAIM M L,KON F. Spectrum-based Software Fault Localization: A Survey of Techniques, Advances, and Challenges[J]. arXiv:1607.04347.

[13] MORENOCENTENO E,KARP R M. The Implicit Hitting Set Approach to Solve Combinatorial Optimization Problems with an Application to Multigenome Alignment[J]. Operations Research,2013,61(2):453-468.

[14] KRAMER O. Genetic Algorithm Essentials[M]. West Berlin and Heidelberg:Springer International Publishing,2017.

[15] ZENG R,ZANG R,DING L,et al. Fault Diagnosis of Diesel Engine Based on Genetic Algorithms and Dempster-Shafer Fusion Theory[C]// Chinese Control And Decision Conference. IEEE,2017.

(上接第 201 页)

[7] LI C H,LU Z D. An image digital watermarking based on support vector machine[J]. Journal of Image and Graphics,2016,11(9):1322-1326. (in Chinese)
李春花,卢正鼎. 一种基于支持向量机的图像数字水印算法[J]. 中国图像图像学报,2006,11(9):1322-1326.

[8] THANKI R,BORRA S,DWIVEDI V,et al. A RONI Based Visible Watermarking Approach for Medical Image Authentication [J]. Journal of Medical Systems,2017,41(9):143.

[9] AGARWAL H,SEN D,RAMAN B. Visible watermarking based on importance and just noticeable distortion of image regions[J]. Multimedia Tools & Applications,2016,75(13):7605-7629.

[10] GUO J,SHI P F. Visible watermarking technique based on luminance and texture features[J]. Infrared and Laser Engineering,2003,32(1):92-95. (in Chinese).
郭捷,施鹏飞. 基于亮度和纹理特征的可见水印技术[J]. 红外与激光工程,2003,32(1):92-95.

[11] XU C,LU Y,ZHOU Y. An automatic visible watermark removal technique using image inpainting algorithms[C]// International Conference on Systems and Informatics. 2017:1152-1157.

[12] RANGEL E K,FRAGOSO N E,CRUZ R C. Adaptive removable visible watermarking technique using dual watermarking for digital color images[J]. Multimedia Tools & Applications,2018,77(11):1-28.

[13] DU R,WANG H,ZHAO T T. A new lossless recoverable colour visible watermarking technology[J]. Computer Applications and Software,2015(10):213-215. (in Chinese)
杜染,王晅,赵婷婷. 一种新的可无损恢复的彩色可见水印技术[J]. 计算机应用与软件,2015(10):213-215.

[14] XIN P Z,SHUO Z W,GUO R F. Reversible Visible Watermarking with Lossless Data Embedding Based on Difference Value Shift[J]. Intelligent Automation & Soft Computing,2011,17(2):233-243.

[15] HAN M T,CHANG L W. Secure reversible visible image watermarking with authentication[J]. Signal Processing: Image Communication,2010,25(1):10-17.

[16] LI H,LIU G,DAI Y. Copyright Protecting Using The Secure Visible Removable Watermarking In JPEG Compression[J]. International Journal of Digital Content Technology & Its Applications,2010,4(4):34-42.

[17] CHEN L,LIU H W,DENG X H. A digital image reversible watermarking algorithm based on integer wavelet transform[J]. Computer Applications and Software,2016,33(4):286-291. (in Chinese)
陈亮,刘惠文,邓小鸿. 基于整数小波变换的数字图像可逆水印算法[J]. 计算机应用与软件,2016,33(4):286-291.