基于频率差异积分的故障定位算法研究

郑 炜, 李知隆, 靳如一

(西北工业大学 软件与微电子学院, 陕西 西安 710082)

摘 要:软件故障定位是软件测试实践中最重要的活动之一。目前,利用人工方式通过调试器进行故障定位仍然占据主导地位,故障定位能力常常依赖于人员的知识和经验,调试效率低,调试支持工具相对较弱,对程序员要求高。因此,无论在软件开发阶段还是软件投入使用以后,软件调试和故障定位的自动化技术越来越成为一个有重要价值的研究课题。研究了多种成熟的基于谓词的统计学错误定位技术,分析了典型的参数化和非参数化故障定位统计模型,提出了一种新的基于谓词统计的故障定位算法:频率差异积分算法(Frequency Difference Integration,简称为 FDI),突破了现有算法的部分限制;最后对 FDI 算法进行了验证,并和已有算法进行了对比,验证了其有效性和高效性。借助该研究成果,可以有效提高故障定位的准确率和覆盖率,这对于推动程序自动化调试技术、缩短软件开发周期、降低软件的维护成本具有一定的意义和实用价值。

关 键 词:算法,计算机软件,计算复杂度,计算效率,故障检测,数学模型,概率密度函数,软件工程,软件可靠性,软件测试,统计;故障定位,谓词统计

中图分类号:TP311.5 **文献标识码**:A **文章编号**:1000-2758(2013)03-0435-05

随着软件规模越来越大,再加之当前软件开发和运行环境的开放性、动态性、多变性,使得软件产品在推出时就含有很多已知或未知的缺陷,很多时候不以人们期望的方式工作,经常发生故障、失效,给人们带来了各种损失[1]。软件故障严重影响软件的生产和质量,并且影响在日益加剧。

目前,利用人工方式通过调试器进行故障定位仍然占据主导地位,故障定位能力常常依赖于人员的知识和经验。特别是对于大型、复杂的、并发的软件系统而言,如何制定调试策略、进行故障诊断均很有技巧性的要求,而当前的软件调试支持与故障诊断分析技术,仍然处于较原始的状态,调试主要以手工方式为主进行,调试效率低,调试支持工具相对较弱,对程序员要求高。因此,研究软件调试和故障定位的自动化技术支持,无论在软件开发阶段还是软件投入使用以后,故障定位问题日益成为一个有重要价值的研究课题。

1 FDI 算法原理:利用真值率频率分布密度函数包络面积量化差异

伊利诺伊大学香槟分校的 ChaoLiu 提出错误定位算法 SOBER,建立了分别在正确和错误的运行中对谓词的评估方法模型:如果一个谓词在错误和正确的运行中的评估结果相差很大,那么就认为它是与错误相关的,并且以通过对模式差异的有原则量化来测量程序谓词的错误相关度[3]。SOBER 在处理收集到的数据时,假设数值的分布为正态分布,采用了正态分布拟合的方法对收集到的数据进行处理。

同 SOBER[2,3] 一样,我们对某个程序的一次运行过程中某一个谓词 P 多次赋值的记录采用真值率来描述:

1)在程序的一次运行中,假设谓词 P 被记录为真的次数为 n_t,被记录为假的次数为 n_f,则将

收稿日期:2012-09-04

作者简介:郑 炜(1975—),西北工业大学副教授,主要从事软件测试、验证的研究。

$$\pi(P) = \frac{n_t}{n_t + n_f} \tag{1}$$

称为谓词 P 在这次执行中的真值(频) 率。

由极大似然法知,在样本足够多的情况下,P 的真值频率无限逼近 P 取值为真的概率,因此 $\pi(P)$ 在一定程度上反映了谓词 P 在一次执行中取值为真的概率。

2) 假设对于一个程序,所有的测试用例集合为 T,其中 T_f 表示失败的测试用例集合,T_p 表示成功的测试用例集合,则用 $f(X \mid \theta_p)$ 和 $f(X \mid \theta_f)$ 分别表示 T_f 和 T_p 的真值率的概率密度函数。

假如图1为某一程序中某一谓词 P_d 的 $f(X \mid \theta_p)$ 和 $f(X \mid \theta_f)$ 在同一坐标系中的函数图像。

图中,横坐标表示真值率 $\pi(P)$,纵坐标表示某真值率占有的比例,即真值率为该值的概率。

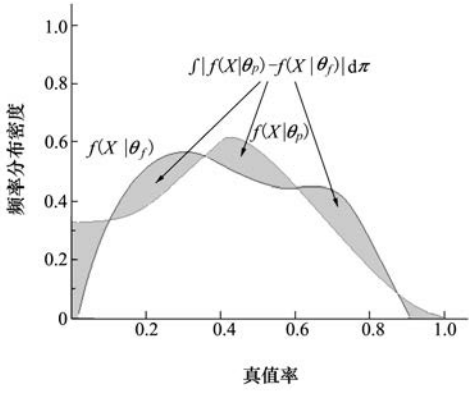

图1　$f(X \mid \theta_p)$ 和 $f(X \mid \theta_f)$ 包络面积

在这里我们无法知道 $f(X \mid \theta_p)$ 和 $f(X \mid \theta_f)$ 符合何种已知的分布,因此无法像 SBOER 那样以参数化的方式描述。但是我们可采用这两者相差的部分的面积

$$S(P) = \int_0^1 | f(X \mid \theta_p) - f(X \mid \theta_f) | \, d\pi(P) \tag{2}$$

作为衡量二者差异的关键值。

2　FDI 算法的具体策略和步骤

因为 $f(X \mid \theta_p)$ 和 $f(X \mid \theta_f)$ 不符合任何已知的分布模型,无法知道 $f(X \mid \theta_p)$ 和 $f(X \mid \theta_f)$ 的函数表达式,因此本算法中用“化整为零,分段计算” 的方式代替直接构造解析表达式,如图2所示。

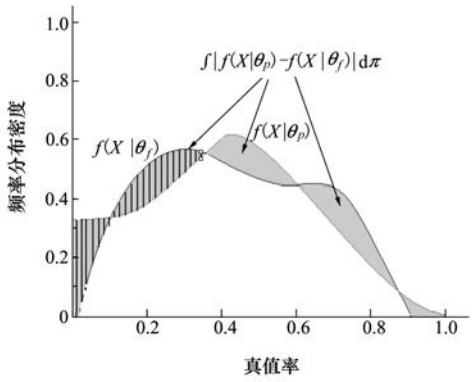

图2　对面积化整为零

假设在总共 X 次的程序执行中,谓词 P 得到了 Y 个真值率记录(因为可能有的执行从不经过谓词 P,故有 $X >= Y$。

谓词 P 的真值率 $\pi(P)$ 的取值范围为闭区间 $[0, 1.0]$,因此可以将该区间分成 N 段单独处理:例如取 $N = 10\ 000$, 我们就得到了 $[0,0.000\ 1)$, $[0.000\ 1,0.000\ 2)$, $[0.000\ 2,0.000\ 3)$, $[0.000\ 3, 0.004)\cdots[0.999\ 9,1.0]$ 这 10 000 个子区间。假设对于每个子区间 $[a,b]$ $(0 <= a < b <= 1)$ 上,在成功运行和失败运行的样本中,$\pi(P)$ 落在该区间范围内的样本数分别为 Z_p 和 Z_f,则相应的分布密度可分别表示为

$$F_p(a,b) = \frac{Z_p}{Y} \quad F_f(a,b) = \frac{Z_f}{Y} \tag{3}$$

于是在该子区间上,2 个函数的曲线围成的等效面积就为

$$\left| \frac{Z_f}{Y} - \frac{Z_p}{Y} \right| \tag{4}$$

而在区间[0,1]上,2 个函数围成的总面积就为

$$\sum_{i=1}^{10\,000} \left| \frac{Z_{fi}}{Y} - \frac{Z_{pi}}{Y} \right| \tag{5}$$

显然,当子区间的个数越多,每个子区间的长度就越小,误差也就越小。理论上,当 N 接近正无穷时,误差为0,二者完全相等,既有

$$\int_0^1 | f(X \mid \theta_p) - f(X \mid \theta_f) | \, d\pi(P) = \lim_{N \to \infty} \left(\sum_{i=1}^{N} \left| \frac{Z_{fi}}{Y_f} - \frac{Z_{pi}}{Y_p} \right| \right) \tag{6}$$

从而

$$S(P) = \lim_{N \to \infty} \left(\sum_{i=1}^{N} \left| \frac{Z_{fi}}{Y_f} - \frac{Z_{pi}}{Y_p} \right| \right) \tag{7}$$

此即为 FDI 算法最后要求的结果。

综上所述,FDI 的核心就是采用化整为零的方法,将需计算的面积分成 N 块,对每一块求值,然后将所有的 N 个子块的面积求和,即可求得所要求的总区间[0,1.0]上的总面积,也就是反映谓词和错误相关性的度量值。在一定范围内,N 的值越高,计算精确度也越高。但是,考虑到实际应用当中,真值率的精度是有限的(一般为 double 类型),因此将区间无限缩小将毫无意义。实践结果表明 N 取 1 000 到 10 000 之间效果都很理想。

3　算法实验结果评估

和 CBI、SOBER 以及 MW 为核心的主流算法检测方式一样[4],为了验证算法的效果,本文也采用了 Siemens Suite[5] 所提供的标准被测错误程序和对应的测试用例,对新算法进行了系统的测试。验证结果表明:无论谓词的真值率的概率密度分布函数是否符合任何已知的参数化模型,该算法都能够对其进行有效的错误定位。

对于 SOBER 无法处理的数据分布,FDI 算法同样可以进行有效处理:

在此以 Siemens Suite 中的 1 个错误程序为例:tot_info 套件中的第 1 个错误版本中,一共有 34 个谓词,其中出错的代码对应的谓词为第 16 号谓词($k<0$ L):

```
P1:  if (rdf < =0 || cdf < =0) {
       info = -3.0;
       goto ret3;
       }
       /* 此处省略部分非关键代码 */
P2:  for (i=0; i<r; ++i) {
       double sum =0.0; /* accumulator */
P3:  for (j=0; j<c; ++j)
       {
       long k=x(i,j);
P4:  if (k<0 L)
       {
       info = -2.0;
错误: /* goto ret1; */
       }
       sum += (double)k;
       }
       N += xi[i] = sum;
       }
P5:  if (N < =0.0) {
       info = -1.0;
       goto ret1;
       }
       /* compute column sums */
P6:  for (j=0; j<c; ++j) {
       double sum =0.0; /* accumulator */
P7:  for (i=0; i<r; ++i)
       sum += (double)x(i,j);
       xj[j] = sum;
       }
```

在上述代码中,在谓词 $P4$ 控制的代码段中有 1 处错误:少了 1 行 goto 语句,直接导致程序逻辑出错。谓词 $P4$ 在整个代码中的序号为第 16 位。

表 1 分别列出用 SOBER 和 FDI 算法验证所得出结果。

表 1　SOBER 和 FDI 结果对比

SOBER		FDI	
谓词序号	关键值	谓词序号	关键值
18	62.598 5	16	2
6	42.931 1	19	1.809 69
1	24.009 9	18	1.631 26
20	23.370 2	23	1.482 74
32	23.292 8	21	1.433 19
…	…	…	…
31	-10 000	31	0
16	-10 000	24	0
25	-10 000	25	0
26	-10 000	26	0

从表 1 中可以看出,在用 SOBER 计算出的结果中,第 16 号谓词 P4 在可疑列表排名中排名倒数第 3,计算出的关键值最低,可见 SOBER 已完全失效;而在用 FDI 计算出的结果中,第 16 号谓词排名第 1,计算出的关键值达到了理论上最大的 2.0,故该算法的效果非常令人满意。

再给出一个更加直观的实例。

Siemens Suite 中 replace 套件第 1 个错误版本的第 14 号谓词的真值率在成功和失败的运行中的频数分布如图 3 和图 4 所示。

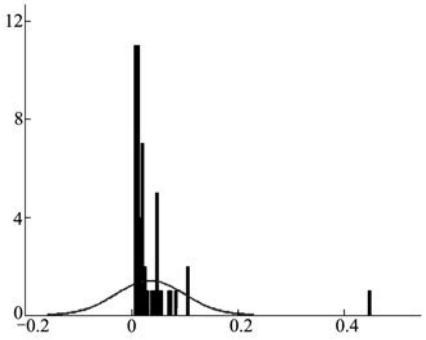

图3　成功运行中谓词 P 的频数分布

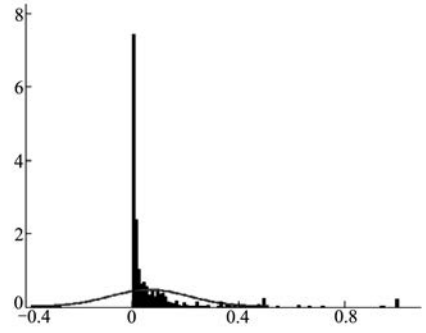

图4　失败运行中谓词 P 的频数分布图

显然,上述数据并不符合正态分布,但是若要按照 SOBER 的算法对数据进行处理,则从图3和图4可知,对上述数据进行正态拟合后,本来差别很大的2组数据,拟合后二者的平均值却非常接近,根据 SOBER 的计算方式,最后计算出的可疑度值非常低,也就是说这种处理方式丢失了大量的数据,算法效果非常差。若采用了新的算法,则根本不受是否服从正态分布的影响,都能计算出二者的面积差,新算法依旧有效。因此,该算法可以解决 SBOER 无法完成的某些错误定位需求。

和上文中提到的基于 MW 算法的非参数化错误定位技术相比,FDI 算法最大的优势在于计算成本非常低,不仅算法的时间复杂度很低,而且对计算机硬件要求很低,并且解决了混合编秩带来的大量误差。

4　结　论

FDI 算法成功突破了 SOBER 的一些限制,并且解决了算法的时间复杂度的问题,实际应用价值非常高。

但是本文提出的算法 FDI 仍然有其局限性,如在测试用例不足时,算法的定位效果会下降,这也是其他基于测试用例的错误定位算法共有的问题,因此考虑如何解决这个限制也是重要的需求,对实际应用时降低测试用例制作成本,提高精度有着重要意义。

其次,在少数情况下,谓词真值率的取值为固定的几个值,而非随机分布时,FDI 的理论就不如其他算法那么适合了,如何解决这个问题也有待进一步研究。

参考文献:

[1] Tao Hongwei, Chen Yixiang. A New Metric Model for Trustworthiness of Software's Telecommunication System. Information Science and Application (ICISA), 2011

[2] Jame S Collofello, Scott N Woodfield. Evaluating the Effectiveness of Reliability Assurance Techniques. J Syst Softw, 1989, 9(3): 191-195

[3] Liu Chao, Yan Xifeng, Fei Long. SOBER: Statistical Model-Based Bug Localization. ACM SIGSOFT Software Engineering Notes, 2005, 30(5): 286-295

[4] Fei Long, Lee Kyungwoo, Li Fei, Midkiff Samuel P. Argus: Online Statistieal Bug Deteetion. FASE 2006, LNCS 3922, 2006, 308-323

[5] Hu P, Zhang Z, Chan W K, Tse T H. Quality Software. The Eighth International Conference on OSIC, 2008, 385-395

Proposing an Effective Software Fault Location Algorithm Based on Frequency Difference Integration

Zheng Wei, Li Zhilong, Jin Ruyi
(Department of Software Engineering, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: Locating software faults manually with the help of debugging tools often relies on the knowledge and experience of a programmer, has poor debugging efficiency and high requirements for the programmer. Therefore, the automation of software debugging and fault location is increasingly urgent and important. However, due to the complexity of software faults and complicated processes, the existing fault location methods have strong dependence and weak versatility, being difficult in analyzing the causes of the faults correctly. We analyze the existing non-parametric and parametric statistical fault location algorithms and propose our fault location algorithm for the frequency difference integration (FDI) of predicate statistics, which we believe is more effective and efficient. Finally, to verify the effectiveness and efficiency of our fault location algorithm, we perform experiments on the Siemens suite; the experimental results, given in Figs. 3 and 4 and Table 1, and their analysis show preliminarily that: (1) our fault location algorithm can effectively locate software faults no matter whether the probability density function of the predicate's truth value conforms with any known parametric models; (2) compared with other fault location algorithms, our fault location algorithm reduces computational costs and computational complexity.

Key words: algorithms, computer software, computational complexity, computational efficiency, fault detection, mathematical models, probability density function, software engineering, software reliability, software testing, statistics; fault location, predicates statistics

《机械科学与技术》2013 年第 32 卷第 6 期共发表 31 篇平均每篇有第六版《Ei Thesaurus》主题词 1.94 个

《机械科学与技术》第 32 卷第 6 期共有 31 篇论文,把每篇论文使用的 Ei 主题词数目加起来共得 60 个,平均每篇有 1.94 个 Ei 主题词。

按照所用第六版《Ei Thesaurus》主题词的数目,可把 31 篇分成 5 组:(1)1 篇 5 个;(2)8 篇各 3 个;(3)11 篇各 2 个;(4)9 篇各 1 个;(5)2 篇各 0 个。

5 个的 1 篇见该期第 884 页;3 个的 8 篇分别见该期第 781、791、796、805、814、819、834、932 页。

胡沛泉

2013 年 6 月