# 基于增强径向函数神经网络的错误定位方法*

张 柯, 张德平, 汪 帅
(南京航空航天大学 计算机科学与技术学院, 南京 210016)

**摘 要**: 结合径向基函数神经网络与正交实验设计理论,提出了一种增强径向基函数神经网络错误定位算法。根据选择的测试用例执行得到源程序的语句覆盖信息和执行结果;通过神经网络计算出每条语句的可疑度值,并通过正交实验设计方法自适应调整神经网络中的参数值;最后按照可疑度值由高到低的顺序逐条检查程序的可疑语句进行错误定位。通过实验对所提出方法与径向基函数神经网络算法以及反向传播神经网络算法进行比较分析,结果表明,基于增强径向基函数神经网络算法具有更精确的错误定位效果和更显著的定位效率。

**关键词**: 错误定位; 程序调试; 径向基神经网络; 正交实验设计; 软件测试
**中图分类号**: TP311.5 **文献标志码**: A **文章编号**: 1001-3695(2015)03-0781-05
doi:10.3969/j.issn.1001-3695.2015.03.032

## Fault localization based on enhanced radial basis function network

ZHANG Ke, ZHANG De-ping, WANG Shuai
(*College of Computer Science & Technology, Nanjing University of Aeronautics & Astronautics, Nanjing* 210016, *China*)

**Abstract**: Combination of radial basis function neural network and orthogonal experiment design theory, this paper proposed an enhanced radial basis function neural network fault location algorithms. First, according to the selected test cases executed, it got the source code statement coverage information and results of the implementation. Then, by neural networks to calculate the suspicious value of each statement, and through the orthogonal experimental design adaptive adjustment neural network parameter values. Finally, in accordance with suspicious values in descending order by one checker error located suspicious statement. Through the test the proposed method and radial basis function neural network algorithm and back-propagation neural network algorithm for comparative analysis, results show that the enhanced radial basis function neural network algorithm has a more precise positioning error localization effects and more significant efficiency.

**Key words**: fault localization; program debugging; radial basis function network; orthogonal experimental design (OED); software testing

## 0 引言

软件开发和维护是非常复杂且容易出错的过程,据统计,在软件开发和维护阶段大约 50% ~70% 左右的工作量要花费在查找软件错误上[1]。而查找软件错误的主要手段是调试,这是整个调试任务中最为困难的部分[2]。随着现代计算机软件系统的规模日益庞大,功能越来越复杂,导致软件维护难度也越来越大,如何更加高效地定位软件中的错误已成为社会广泛关注的焦点之一。

传统的错误定位方法主要是采用人工的方法,利用调试工具设置断点,跟踪程序的执行,并检查程序运行时的状态值是否与预期值相同来进行错误定位。显然,这种方法对调试人员的编程维护能力要求非常高,且对程序中所有的执行语句无差别对待,费时费力,效率低下。随着软件业突飞猛进的发展,软件规模日益增大,复杂性也越来越高,这种单纯利用人工查找错误的方法变得越来越困难,迫切需要引入自动化方法来定位程序中的错误,以提高整个软件调试过程的效率。

现有的错误定位方法主要是采用具有良好覆盖的测试用例集或测试套件,得出程序的覆盖信息以及执行结果,并通过错误定位模型计算出可能出错的语句集合。因此,一个高效的错误定位方法的核心问题在于出错语句可疑度计算的建模。按照使用程序执行信息的差异和是否修改程序运行时的状态,可以将自动化错误定位方法分为基于程序切片的方法、基于程序行为特征对比的方法和基于程序状态修改的方法三类。

基于程序切片的方法[3,4]使用程序中的数据依赖、控制依赖等关系给出可能出错的语句集合,这个集合中不仅包含了出错语句,还包含了帮助程序员理解的程序调试时的上下文。这种方法求得的出错语句集合往往很大,且集合中的待检测语句没有优先级,虽然命中率较高,但是搜索范围也较大。

基于程序行为特征对比的方法主要是利用程序执行时的统计信息[5]。Program Dice[6,7]方法首先通过覆盖率或是随机选择两个测试用例(一个通过一个失败),然后求得失败测试用例的切片和用过测试用例的切片差集 Dice,在 Dice 中搜索程序的错误。该方法的不足之处在于对于错误位置的假定,因此它不能定位到同时存在于失败用例和通过用例的切片中的错误,而且仅选择一个通过测试用例作比较有较大的随机性。

基于程序状态修改的方法首先对比通过测试用例与失败测试用例的运行状态,并动态修改程序状态,然后观察修改之后的测试结果,并依此找出对程序中的测试结果有影响的语

**收稿日期**: 2013-11-13; **修回日期**: 2013-12-26 **基金项目**: 中央高校基本科研业务费专项资金资助项目(NS2012072)

**作者简介**:张柯(1984-), 男, 硕士研究生,主要研究方向为软件测试与软件可靠性建模; 张德平(1973-), 男, 讲师,博士,主要研究方向为软件测试与软件可靠性建模(depingzhang@nuaa.edu.cn); 汪帅 (1987-), 男, 硕士研究生,主要研究方向为软件测试与软件可靠性建模.

句,该类方法典型的有 Delta Debugging[8~10] 和 Predicate Switching[11],其特点是复杂度低,易于实现,且有效性高。

最近,Wong 等人[12~14] 提出了一种径向基函数神经网络(radial basis function network , RBFN)定位算法,该方法是把覆盖信息当做神经网络的神经元,中间层引入径向基函数,然后通过训练神经网络的方法来找出错误语句,并已证明该方法相较于之前的错误定位方法在效率上有所提升。然而,该方法的不足之处在于径向基函数中存在许多参数,而这些参数的设定对最终的定位精确度影响很大。因此,本文在此基础上结合正交实验设计(orthogonal experimental design, OED)与 RBFN,提出一种新的错误定位算法 ERBFN,利用 OED 方法调整 RBFN 中的参数,使 RBFN 具有最佳的错误定位效果。

## 1 正交实验设计

正交实验设计是研究多因素多水平的一种实验设计方法,它根据正交性从整个实验中挑选出部分有代表性的点进行实验,这些有代表性的点具有均匀分散、齐整可比的特点,利用正交表进行科学的安排与分析。其主要优点是能在众多的实验方案中挑选出代表性强的少数几个实验方案,并且通过这些少数实验方案的结果分析,推断出最优方案,是一种高效、快速、经济的实验设计方法。

正交表是一种特殊的表格,是正交实验设计的基本工具,记为 $L_M(Q^N)$。其中,$L$ 是正交表的代号,$M$ 是正交表的行数,代表需要做的实验次数,$Q$ 代表各因素的水平数,$N$ 是正交表的列数,代表最多能安排的因素个数。如

$$L_4(2^3)=\begin{bmatrix}1&1&1\\1&2&2\\2&1&2\\2&2&1\end{bmatrix} \tag{1}$$

式中:$L_4(2^3)$ 表示有三个因素,每个因素的水平数为 2,一共要做四次实验[15]。除了定义之外,正交表矩阵还有两个特点:a)正交表任意一列中,不同的数字出现的次数相等;b)正交表中任意两列,把同行的两个数字看成有序数对时,所有可能的数对出现的次数相同。

定义 $f_m$ 表示第 $m(1\leqslant m\leqslant M)$ 个实验的结果,$S_{nq}$ 表示第 $n(1\leqslant n\leqslant N)$ 个因素的第 $q(1\leqslant q\leqslant Q)$ 个水平的实验结果平均值,如式(2)所示。

$$S_{nq}=\frac{\sum_{m=1}^{M} f_m\times z_{mnq}}{\sum_{m=1}^{M} z_{mnq}} \tag{2}$$

其中:当取第 $m$ 个实验中的第 $n$ 个因素的第 $q$ 个水平时,$z_{mnq}$ 取值为 1,否则取 0。

下面举例说明 OED 的实现过程:求函数 $f(x_1, x_2, x_3)=100x_1-10x_2-x_3$ 的最大值,其中,$x_1$、$x_2$、$x_3$ 的取值分别为 $x_1\in\{1, 2\}$,$x_2\in\{3, 4\}$,$x_3\in\{5, 6\}$。一共有三个因素,每个因素两个水平,一共有 $2^3=8$ 种取值情况要讨论。然而,通过引入正交实验设计,只需要计算四种取值情况,即可得到最后的结果。

a)根据 OED 原则,构建一个 $L_4(2^3)$ 的正交表;b)分别计算 $S_{nq}(n=1, 2, 3;\ q=1, 2)$ 的值,如 $S_{21}=(f_1+f_3)/2=114.5$;c)通过对比每个因素的 $S_{nq}$ 值确定最佳水平,如在因素 $x_1$ 这一列中,$S_{11}<S_{12}$,就取 $x_1=2$;d)可以得出最佳取值为 $(x_1, x_2, x_3)=(2, 3, 5)$,即 $\mathrm{MAX}\ f(x_1, x_2, x_3)=f(2, 3, 5)=165$。具体运算过程与结果如表 1 所示。

表 1 OED 运算过程与结果

| 实验 | $A(x_1)$ | $B(x_2)$ | $C(x_3)$ | 运算结果 |
|---|---|---|---|---|
| $C_1$ | 1 (1) | 1 (3) | 1 (5) | $f_1=65$ |
| $C_2$ | 1 (1) | 2 (4) | 2 (6) | $f_2=54$ |
| $C_3$ | 2 (2) | 1 (3) | 2 (6) | $f_3=164$ |
| $C_4$ | 2 (2) | 2 (4) | 1 (5) | $f_4=155$ |
| | | 因素分析 | | 水平 |
| $S_{j1}$ | 59.5 | 114.5 | 110 | 1 |
| $S_{j2}$ | 159.5 | 104.5 | 109 | 2 |
| OED 结果 | 2 (2) | 1 (3) | 1 (5) | $f^*=165$ |

## 2 增强径向基函数神经网络模型(ERBFN)

ERBFN 是一个三层网络结构。第一层为输入层,主要用来接收外部的输入信息;第二层为中间层,每个中间层节点都包含一个径向基函数,通过输入层的输入数据,计算出每个节点的函数值;第三层为输出层,把中间层节点的各个值通过加权和的形式计算出最后的输出结果,如图 1 所示。

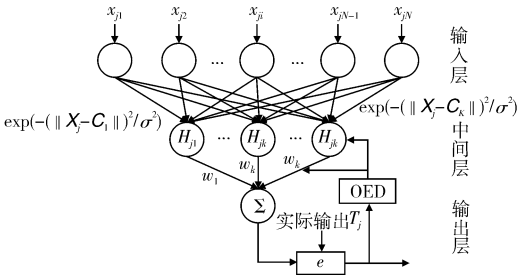


图 1 ERBFN 模型结构

在输入层中,记第 $j$ 个输入向量 $\boldsymbol{X}_j=(x_{j1}, x_{j2}, \cdots, x_{jN})$,$(j=1, 2, \cdots, M)$,那么,可以把输入数据写成如下矩阵形式:

$$\boldsymbol{X}=(x_{ji})_{M\times N}\quad j=1,2,\cdots,M;i=1,2,\cdots,N \tag{3}$$

在中间层中,若中间层节点数为 $K$ 个,记向量 $\boldsymbol{C}_k=(c_{k1}, c_{k2}, \cdots, c_{kN})$ $(k=1,2,\cdots,K)$ 为第 $k$ 个中间层节点的径向基函数中心,$\sigma_k$ 为第 $k$ 个节点的径向基函数的域宽,那么 $\|\boldsymbol{X}_j-\boldsymbol{C}_k\|$ 就表示第 $j$ 个输入向量 $\boldsymbol{X}_j$ 与第 $k$ 个中心 $\boldsymbol{C}_k$ 的 Euclidean 距离,如式(4)所示。

$$\|\boldsymbol{X}_j-\boldsymbol{C}_k\|=\sqrt{\sum_{i=1}^{N}(x_{ji}-c_{ki})} \tag{4}$$

这样便可以得到第 $j$ 个输入向量 $\boldsymbol{X}_j$ 在第 $k$ 个中间层节点的输出结果为

$$H_{jk}=e^{\frac{-(\|\boldsymbol{X}_j-\boldsymbol{C}_k\|)^2}{\sigma_k^2}}\quad k=1,2,\cdots,K \tag{5}$$

式(5)为 Gaussian 函数[16],中间层节点个数选定可由初始中心个数而定,而初始中心的训练可以通过文献[12 , 17 ~ 19]的方法来确定。

在输出层,记 $w_1, w_2, \cdots, w_K$ 为各个中间层节点的权值,最后的输出结果便可由式(6)确定:

$$y_j=\sum_{k=1}^{K} w_k H_{jk} \tag{6}$$

式(7)表示通过神经网络计算后的结果与预期结果的拟合程度。

$$e=\sqrt{\sum_{j=1}^{M}(T_j-y_j)^2}=\sqrt{\sum_{j=1}^{M}(T_j-\sum_{k=1}^{K}w_k H_{jk})^2}=\sqrt{\sum_{j=1}^{M}(T_j-\sum_{k=1}^{K}w_k e^{\frac{-(\|\boldsymbol{X}_j-\boldsymbol{C}_k\|)^2}{\sigma_k^2}})^2} \tag{7}$$

其中:$T_j$ 表示第 $j$ 个输入向量的预期结果,$y_j$ 表示第 $j$ 个输入向量的实际结果。

接下来将定义 ERBFN 的训练规则:

a)将 $N$ 维输入向量 $X_1$, $X_2$,…,$X_M$ 依次输入神经网络。

b)每输入一个向量 $X_j$,通过式(5)和(6)计算出向量 $X_j$ 的输出结果 $y_j$。

c)将每个输入向量的计算结果与预期结果进行对比,并通过式(7)得出拟合精度值 $e$,$e$ 越小代表拟合精度越高,如果 $e$ 的值达到理想的拟合精度,训练完成,否则继续步骤 d)。

d)按下式更新参数 $w_k$、$C_k$ 以及 $\sigma_k$[18]:

$$w_k(n+1)=w_k(n)-\mu_w\frac{\partial}{\partial w_k(n)}e(n) \tag{8}$$

$$C_k(n+1)=C_k(n)-\mu_c\frac{\partial}{\partial C_k(n)}e(n) \tag{9}$$

$$\sigma_k(n+1)=\sigma_k(n)-\mu_\sigma\frac{\partial}{\partial \sigma_k(n)}e(n) \tag{10}$$

其中:$n$ 表示神经网络训练的代数,$\mu_w$、$\mu_c$、$\mu_\sigma$ 为学习参数,将这三个参数代入到 OED 当中,水平数设置为 2,分别设置为

$$\begin{cases}\mu_{\bullet 1}=\mu_\bullet+\Delta\mu\\ \mu_{\bullet 2}=\mu_\bullet-\Delta\mu\end{cases} \tag{11}$$

其中:$\Delta\mu$ 为学习参数调整值,一般设置为很小的数。这样,通过 OED 可得到下一代拟合精度最佳的参数值,返步骤 a),继续执行。

## 3　基于 ERBFN 的错误定位算法

考虑有一个程序 $P$,$P$ 有 10 条语句、8 个测试用例,以及这些测试用例的覆盖信息和执行结果,如表 2 所示。

表 2　错误定位的程序与输入信息实例

| 语句 | 程序实例 | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ |
|---|---|---|---|---|---|---|---|---|---|
| | function cout(char * s){ | | | | | | | | |
| | int let,dig,other,i; | | | | | | | | |
| $s_1$ | while(c=s[i++]){ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $s_2$ | if('A'<=c&&'Z'>=c) | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| $s_3$ | let+=2; //FAULT | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| $s_4$ | else if('a')<=c&&'z'>=c) | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| $s_5$ | let+=1; | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| $s_6$ | else if('0'<=c&&'9'>=c) | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| $s_7$ | dig+=1; | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| $s_8$ | else if(isprint(c)) | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| $s_9$ | other+=1; | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| $s_{10}$ | print("%d %%",let,dig,other); | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | $R_i$ | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

其中:$t_i(i=1,2,\cdots,8)$ 表示第 $i$ 个测试用例,$s_j(j=1,2,\cdots,10)$ 表示第 $j$ 条语句。覆盖信息中,1 代表测试用例 $t_i$ 在测试过程中执行了语句 $s_j$, 0 则表示没有执行。$R_i$ 表示第 $i$ 个测试用例的执行结果,0 代表执行通过,1 则代表执行失败。

将数据代入到 ERBFN 模型中去,每个测试用例的语句覆盖信息对应神经网络的输入层神经元,以上实例 ERBFN 的输入层节点为 10,共有八组测试用例。数据进入模型后,首先确定中间层节点的个数[12]以及各个中间层节点中心 $C$ 和域宽 $\sigma$ 的初始值[20],然后通过 Gaussian 函数[16]以及各中间层节点的加权和计算出实际的输出结果,再与实例的实际结果 $R$ 作比较,得出拟合精度,利用 OED 方法不断训练与调整里面的参数,直到拟合精度值 $e$ 小于预定的拟合精度,这就代表径向基函数神经网络已经训练完成,所有参数都训练到了最佳值。

现在,ERBFN 已经非常好地适应了测试用例覆盖信息以及执行结果的数据变化情况。为了使 ERBFN 能够应用到错误定位中来,引入一组虚拟测试用例[12]。设 $v_1$,$v_2$,…,$v_m$ 为一组 $m$ 维的虚拟测试用例向量,其中 $m$ 为程序语句的条数,测试用例向量 $v_j(j=1,2,\cdots,m)$ 只执行一条语句 $s_j$,这样,便构建了一个 $m\times m$ 的方阵,如

$$\begin{bmatrix}v_1\\v_2\\\vdots\\v_m\end{bmatrix}=\begin{bmatrix}1&0&\cdots&0\\0&1&\cdots&0\\\vdots&\vdots&&\vdots\\0&0&0&1\end{bmatrix} \tag{12}$$

最后,以虚拟测试用例向量 $v_1$,$v_2$,…,$v_m$ 为输入,加入到先前已经训练完毕的 ERBFN 当中,得出最终的输出结果 $R_1$, $R_2$, …, $R_m$,这 $m$ 个结果值就是 $m$ 条语句的可疑度值,如 $R_j$ ($j=1$, 2, …, $m$)值越大,就代表语句 $s_j$ 出错的可能性越大。至此,整个 ERBFN 算法的错误定位已经完毕,具体过程如下:

a)输入测试用例数据以及构建 ERBFN 神经网络模型,确定网络中间层节点的个数。

b)对网络中间层的各个节点参数进行初始化,参数包括中心 $C$、域宽 $\sigma$ 和权值 $w$。

c)用 OED 方法来调整最佳的学习参数 $\mu_w$、$\mu_c$、$\mu_\sigma$,以更新神经网络中的参数,直到训练完成。

d)使用一组虚拟测试用例 $v_j(j=1,2,\cdots,m)$ 为输入,计算出最后的输出结果 $R_j(j=1,2,\cdots,m)$。

e)将语句的怀疑度值按从高到低的顺序排序,以方便调试人员根据每条语句的怀疑度逐行检查错误,从而提高错误定位效率。

由表 2 中给出的实例计算出最后输出结果如表 3 所示。根据表 3 所计算出的语句可疑度,对其降序排列:$s_3$、$s_2$、$s_1$、$s_{10}$、$s_4$、$s_5$、$s_6$、$s_7$、$s_8$、$s_9$,通过此实例结果,可以快速高效地找出错误语句 $s_3$。

表 3　实例输出结果

| 语句 | 输出 | 语句 | 输出 | 语句 | 输出 |
|---|---|---|---|---|---|
| $s_1$ | 0.092933 | $s_2$ | 0.533692 | $s_3$ | 0.548684 |
| $s_4$ | 0.004685 | $s_5$ | 0.004292 | $s_6$ | 0.001045 |
| $s_7$ | 0.001045 | $s_8$ | 0.000649 | $s_9$ | 0.000649 |
| $s_{10}$ | 0.092933 | | | | |

## 4　实验结果与分析

本文使用测试数据集 Siemens Suite[21]作为实验的研究对象,其包含了七个程序,每个程序都有正确版本、若干错误版本和测试用例。每个错误版本都包含一个错误,有些错误可能跨越多行。以上所有资源可以从 Software-artifact Infrastructure Repository (SIR)[21]获得,如表 4 所示。

表 4　Siemens Suite

| 程序 | 错误版本数 | 测试用例数 | 可执行语句数 |
|---|---|---|---|
| Print_tokens | 7 | 4 130 | 175 |
| Print_tokens2 | 10 | 4 115 | 178 |
| Replace | 32 | 5 542 | 216 |
| Schedule | 9 | 2 650 | 121 |
| Schedule2 | 10 | 2 710 | 112 |
| Tcas | 41 | 1 608 | 55 |
| Tot_info | 23 | 1 052 | 113 |

Siemens Suite 中一共包含 132 个错误版本,其中有 12 个版本被排除在实验之外:Replace 程序的版本 32、Schedule2 程序的版本 9 在其所在版本所有的测试用例均通过,所以没有失败的测试用例;Print_tokens 程序的版本 4 和版本 6,Tcas 程序的版本 13、版本 14、版本 36、版本 38,Tot_info 程序的版本 6、版本 10、版本 19、版本 21,这些版本的错误不在主程序文件而在头文件中。所以,最后只有 120 个错误版本应用到了实验之中。

整个系统实现需要经过数据采集、程序特征提取、正确版本与错误版本比较、数据整合、错误定位算法导入、可疑语句排序、算法定位效率比较等过程,具体流程如图 2 所示。

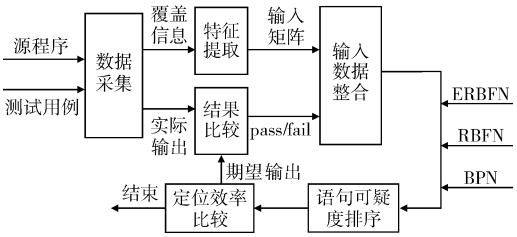


图 2 Siemens Suite 错误定位实验基本流程

a)数据采集。该过程的主要功能是获取测试用例集(测试套件)对应的程序输出以及程序的覆盖信息,为下面进一步提取信息作准备。主要有插桩、编译运行测试用例以及生成覆盖文件等步骤。

b)特征提取。其功能是从已得到的覆盖信息文件中抽象出错误定位算法需要的特征信息,主要分为过滤、分离、提取三个步骤。

c)结果比较。该过程是为了判断测试用例是否通过。对比所有测试用例的实际输出与期望输出,判断两者内容是否相同,如果相同,就输出 pass,否则输出 fail。

d)输入数据整合。该过程的主要功能是将测试套件中所有测试用例的执行结果以及所有提取后的特征值进行合并(由 0 和 1 组成),形成表 2 所示的错误定位算法接口输入数据。

e)运行算法与语句可疑度排序。将输入数据分别代入错误定位算法 ERBFN、RBFN 和 BPN 中,得出这三种算法的语句可疑度排序情况。

f)定位效率比较。按照语句可疑度从高到低的顺序,进行逐行查找错误,并统计所有错误版本的定位情况,形成算法综合效率比较数据。

错误定位算法主要的效率比较标准是在已经定位出错误的前提下,检查尽可能少的语句,这里采用 accuracy 值来描述定位效率[22]:

$$\text{accuracy} = \frac{\text{错误语句的所在可疑度排名}}{\text{执行语句数}} \tag{13}$$

其中:accuracy 表示在找到真正的错误语句之前所需要查找语句的百分比,其值越小,代表错误定位的效率越高。

图 3 比较了 ERBFN、RBFN 和 BPN 算法 accuracy 值的情况,其中,$x$ 轴表示已查找了的语句数占总语句数的百分比,$y$ 轴表示已找出错误的错误版本数占所有错误版本数的百分比。在查找相同语句数量的情况下,能找出错误的错误版本数越多,代表错误定位算法的性能越好。

为进行进一步细节比较,本文引入标准 Imp 百分比[23],Imp 表示在单个程序所有错误版本的错误都被找出的情况下,所花费在查找上的语句数总和占所有错误版本语句数总和的百分比。显然,Imp 百分比越低,代表错误定位算法所消耗的查找时间越少,效率越高。

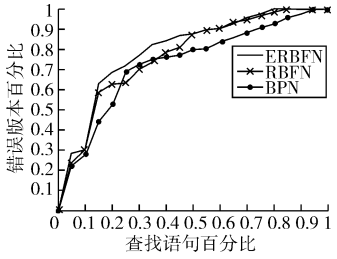


图 3 ERBFN、RBFN 和 BPN 的 accuracy 值比较

错误定位算法的 Imp 百分比比较如图 4 所示。与此同时,还可以从每个错误定位算法节省查找语句开销的最大、最小以及平均值的角度进行性能分析。

从图 5 可以看出,在 Siemens Suite 的七个程序中,ERBFN 算法无论在最好情况、最坏情况还是平均情况下,相比于 RBFN 和 BPN 算法,找到错误并节省需要查找的语句数均最多。

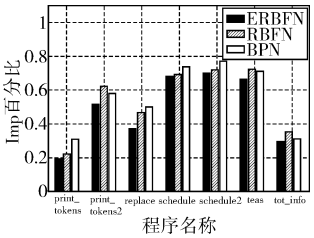


图 4 ERBFN、RBFN 和 BPN 各个程序的 Imp 百分比比较

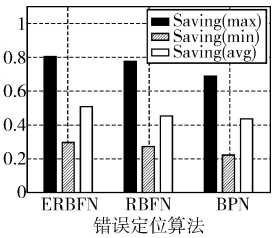


图 5 各算法节省查找语句最大、最小以及平均值对比

## 5 结束语

本文将正交实验设计(OED)与径向基函数神经网络(RBFN)错误定位算法相结合,提出了一种增强径向基函数神经网络(ERBFN)错误定位算法。实验表明,该方法在定位错误的效率和定位效果上较 RBFN 和 BPN 算法具有一定提升。

本文不足之处在于:实验中使用的实验对象 Siemens Suite 中程序的错误版本都是单一错误,没有考虑程序中存在多个错误的情况,也没有考虑多个错误相互之间的关系。这些问题都需要进一步的深入研究。另外,本文目前实现的仅仅是将错误定位算法所得到的结果与源程序关联起来并显示给调试人员,至于修改错误,还是需要调试人员手动去作判断。下一步工作将研究在显示结果的同时允许调试人员动态地修改代码,修改后重新运行所有测试用例,从而进一步提高软件调试效率。

**参考文献:**

[1] COLLOFELLO J S, WOODFIELD S N. Evaluating the effectiveness of reliability assurance techniques [J]. Journal of Systems and Software, 1989, 9(3): 191-195.

[2] BALL T, EICK S G. Software visualization in the large [J]. Computer, 1996, 29(4): 33-43.

[3] LYLE J R, WEISER M. Automatic program bug location by program slicing[C]//Proc of the 2nd International Conference on Computer and Applications. 1987: 877-883.

[4] TIP F. A survey of program slicing techniques[J]. Journal of Programming Languages, 1995, 3(3): 121-189.

[5] REPS T, BALL T, DAS M, *et al*. The use of program profiling for software maintenance with applications to the year 2000 problem

[C]//Proc of the 6th European Software Engineering Conference Held Jointly with the 5th ACM SIGSOFT International Symposium on Foundation of Software Engineering. 1997:432-449.

[6] AGRAWAL H, HORGAN J, LONDON S, *et al*. Fault localization using execution slices and data flow tests [C]//Proc of International Symposium on Software Reliability Engineering. 1995:143-151.

[7] COLLOFELLO J S, COUSINS L. Towards automatic software fault location through decision path analysis [C]//Proc of National Computer Conference. 1987: 539-544.

[8] ZELLER A, HILDEBRANDT R. Simplifying and isolating failure-inducing input [J]. IEEE Trans on Software Engineering, 2002, 28(2): 183-200.

[9] ZELLER A. Isolating cause-effect chains from computer programs [C]//Proc of the 10th ACM SIGSOFT Symposium on Foundations of Software Engineering. 2002:1-10.

[10] CLEVE H, ZELLER A. Locating causes of program failures [C]// Proc of the 27th International Conference on Software Engineering. 2005:342-351.

[11] ZHANG X, GUPTA N, GUPTA T. Locating faults through automated predicate switching [C]//Proc of the 28th International Conference on Software Engineering. 2006: 272-281.

[12] WONG W E, DEBROY V, THURAISINGHAM B, *et al*. RBF neural network-based fault location, Technical Report UTDCS-20-10[R]. 2010.

[13] WONG W E, SHI Yan, QI Y, *et al*. Using an RBF neural network to locate program bugs[C]//Proc of the 19th International Symposium on Software Reliability Engineering. 2008: 27-36.

[14] WONG W E, DEBROY V, GOLDEN R, *et al*. Effective software fault localization using an RBF neural network[J]. IEEE Trans on Reliability, 2012, 61(1): 149-169.

[15] HO S Y, SHO L S, CHEN J H. Intelligent evolutionary algorithms for large parameter optimization problems[J]. IEEE Trans on Evolutionary Computation, 2004, 8(6): 522-541.

[16] HASSOUN M H. Fundamentals of artificial networks[M]. Cambridge, MA: MIT Press, 1995.

[17] DANG J, WANG Y, ZHAO S. Face recognition based on radial basis function neural networks using subtractive clustering algorithm[C]// Proc of the 6th World Congress on Intelligent Control and Automation. 2006: 10294-10297.

[18] LIN G F, CHEN L H. Time series forecasting by combining the radial basis function network and the self-organizing map[J]. Hydrological Processes, 2005, 19(10): 1925-1937.

[19] WAN C, HARRINGTON P B. Self-configuring radial basis function neural networks for chemical pattern recognition[J]. Journal of Chemical Information and Modeling, 1999, 39(6):1049-1056.

[20] MOODY J, DARKEN C J. Learning with localized receptive fields [C]//Proc of Connectionist Models Summer School. 1988: 133-142.

[21] SIR[EB/OL]. http://sir.unl.edu/portal/index.php.

[22] LEI Yan, MAO Xiao-guang, DAI Zi-ying, *et al*. Effective statistical fault localization using program slices[C]//Proc of the 36th IEEE International Conference on Computer Software and Applications. 2012: 1-10.

[23] DEBROY V, WONG W E, XU X, *et al*. A grouping-based strategy to improve the effectiveness of fault localization techniques[C]//Proc of the 10th International Conference on Quality Software. 2010: 13-22.

(上接第 770 页)微博主题发现的速度,实现了快速海量微博主题发现。

今后的研究工作主要集中在:a)优化 PBTM 的内存占用量问题;b)探索 MPI 主机数量和 Coherence 值的关系,从而找出在某个 MPI 主机数量情况下最合适的吉布斯采样迭代次数;c)尝试将 PBTM 运用到海量的短文本分类上。

**参考文献:**

[1] DEERWESTER S C, DUMAIS S T, LANDAUER T K, *et al*. Indexing by latent semantic analysis[J]. Journal of the American Society for Information Science, 1990, 41(6): 391-407.

[2] HOFMANN T. Probabilistic latent semantic indexing[C]//Proc of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 1999: 50-57.

[3] BLEI D M, NG A Y, JORDAN M I. Latent Dirichlet allocation[J]. The Journal of Machine Learning Research, 2003, 3(4-5): 993-1022.

[4] HEINRICH G. Parameter estimation for text analysis[R]. 2005.

[5] RAMAGE D, HALL D, NALLAPATI R, *et al*. Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora [C]//Proc of Conference on Empirical Methods in Natural Language Processing. 2009: 248-256.

[6] 张晨逸, 孙建伶, 丁轶群. 基于 MB-LDA 模型的微博主题挖掘[J]. 计算机研究与发展, 2011, 48(10): 1795-1802.

[7] AJSUMAIT L, BARBARÁ D, DOMENICONI C. On-line LDA: adaptive topic models for mining text streams with applications to topic detection and tracking[C]//Proc of the 8th IEEE International Conference on Data Mining. 2008: 3-12.

[8] WANG Xue-rui, McCALLUM A, WEI Xing. Topical *n*-grams: phrase and topic discovery, with an application to information retrieval [C]//Proc of the 7th IEEE International Conference on Data Mining. 2007: 697-702.

[9] LIU Zhi-yuan, HUANG Wen-yi, ZHENG Ya-bin, *et al*. Automatic keyphrase extraction via topic decomposition[C]//Proc of Conference on Empirical Methods in Natural Language Processing. 2010: 366-376.

[10] FENG Yan-song, LAPATA M. Topic models for image annotation and text illustration[C]//Proc of Annual Conference of the North American Chapter of the Association for Computational Linguistics. 2010: 831-839.

[11] SMYTH P, WELLING M, ASUNCION A U. Asynchronous distributed learning of topic models[C]//Advances in Neural Information Processing Systems. 2008: 81-88.

[12] YAN Xiao-hui, GUO Jia-feng, LAN Yan-yan, *et al*. A biterm topic model for short texts[C]//Proc of the 22nd International World Wide Web Conferences. 2013: 1445-1456.

[13] 张华平, 刘群. 计算所汉语词法分析系统 ICTCLAS[EB/OL]. 2002(2010-08-25). http://www.nlp.org.cn/project/project.php.

[14] MIMNO D, WALLACH H M, TALLEY E, *et al*. Optimizing semantic coherence in topic models[C]//Proc of Conference on Empirical Methods in Natural Language Processing. 2011: 262-272.