# 一种基于二分图故障检测模型的软件故障定位方法研究

王耀宣[1] 叶俊民[1] 陈静汝[1] 欧中红[2]
(华中师范大学计算机科学系 武汉 430079)[1] (中国重工集团709研究所 武汉 430074)[2]

**摘 要** 软件故障诊断过程中代价最昂贵和最耗时的活动之一就是软件故障定位。为了辅助测试人员进行软件故障的定位,在设计分层思想的指导下,在分析软件及其各个模块以及模块中代码之间复杂关系的基础上,通过分析历史数据中软件故障与现象之间的对应关系,建立了基于拓扑图的软件故障传播模型,使得软件故障诊断人员能够利用该拓扑图模型描述具体的软件故障现象。通过该拓扑图模型,软件故障传播模型可转换成更容易进行问题求解的基于二分图的故障检测模型。然后针对该模型设计了基于贪心策略的算法,该算法解决了基于二分图故障检测模型的最小覆盖求解问题,这一问题的求解结果描述了软件故障原因假设集合,通过故障原因与软件模块关系分析可找出与该故障原因对应的相应模块,从而实现故障定位。实验表明,本研究方案能够有效处理软件故障定位问题。

**关键词** 软件故障诊断,故障定位,分层模型,二分图,最小覆盖
**中图法分类号** TP3-0 **文献标识码** A

## Software Fault Location Method Based on Fault Detection Model of Bipartitie Graphs

WANG Yao-xuan[1] YE Jun-min[1] CHEN Jing-ru[1] OU Zhong-hong[2]
(School of Computer Science, Central China Normal University, Wuhan 430079, China)[1]
(The Chinese Heavy Industry Group 709 Institute, Wuhan 430074, China)[2]

**Abstract** In software fault diagnosis process, the most costly and time-consuming is software fault location. In order to help tester locate software fault, with guidance of layering design thought, based on the complex relationship between software and its various modules and codes, this paper proposed the software failure propagation model based on topological graph through analysis of the historical data of the corresponding relationships between software fault and its phenomenon, making it possible to use the topology graph model to describe the software fault phenomenon. Through the topological graph model, the software fault propagation model can be converted into an easier fault detection model based on the bipartitie graphs. Then, an algorithm is designed based on greedy strategy according to this model. This algorithm solves the problem of the minimum coverage solution based on the bipartitie graphs. The result of this solution describes the set of assumed reasons for software faults, finds the corresponding module to the fault through the analysis of the relationship between the fault and the software modules, and thus to achieve fault location. Experiments show that this method of software fault location is effective.

**Keywords** Software fault detection, Software fault location, Layering model, Bipartitie graph, Minimum coverage

Web应用引起的故障原因具有多样性,涉及到终端系统、服务器系统、网络、硬件和软件等方方面面。本文主要关注Web应用的软件系统方面随着Web应用软件系统规模的日益扩大,Web应用中出现故障的可能性也随之增大。因此,如何维持Web应用的服务质量,保证Web应用的高可靠性和高可用性,已经成为一个迫切需要解决的问题。提高Web应用的一个主要手段就是在Web应用发生故障时,采用相应措施避免宕机或性能降级,以便维护人员能够尽快修复故障并恢复系统的正常工作状态,这是一个非常值得研究的热点问题。

根据故障和告警之间的关系,故障诊断技术主要有确定性故障诊断和非确定(概率)故障诊断两种技术,前者常见的技术包括专家系统、基于规则的系统和基于案例的系统,后者常见的技术包括图论技术、神经网络和不确定性推理等。由于基于神经网络技术的故障诊断方法面对的模型需要经历长时间的训练才能贴近实际模型,而不确定性推理方法需要大量样本信息且假设条件过于苛刻,因此采用图论技术是一种可行的不确定性故障诊断技术。采用图论技术进行故障诊断的思路是,首先对故障管理场景进行建模,然后在该故障模型的基础上设计故障定位算法。

在软件故障诊断的诸多研究课题中,最核心的研究内容之一就是软件故障定位问题。目前关于故障场景建模和故障定位算法的研究工作主要有:Bouloutas等人采用上下文无关语法描述故障关联场景[1];Katzela等人采用依赖图建立故障

到稿日期:2012-08-16 返修日期:2012-11-12 本文受湖北省自然科学基金面向项目(2010CDB04001),武汉大学计算机软件工程国家重点实验室开放基金项目(SKLSE20080705),华中师范大学基本科研业务基金项目(CCNU11A02007)资助。
王耀宣(1989—),硕士生,主要研究方向为软件故障诊断,E-mail:jmye@mail.ccnu.edu.cn;叶俊民 男,博士,教授,主要研究方向为可信软件工程;欧中红 博士,研究员,主要研究方向为软件故障诊断与容错技术。

传播模型，并在此基础上提出一种基于分治法的故障定位算法[2]；Steinder 等人采用贝叶斯网络作为不确定性故障传播模型，并在此基础上设计了故障定位算法及其改进算法[3]；Cynthia 等人使用贝叶斯网络对故障进行预测[4]；Yemini 等人采用二分图建模故障管理场景，使用密码本方法进行故障定位[5]；Fischer 等人[6]提出了一个分布式故障诊断方案；Khanna 等人[7]提出了一个基于分布式监测系统。国内的研究也取得了很多成果，如北京邮电大学网络与交换国家重点实验室的程时端等人在多窗口故障诊断、概率和噪声环境下提出基于主动探针的 Internet 服务故障管理、动态环境下的互联网服务故障诊断算法，并对多域服务环境下的分布式故障诊断算法等进行了深入研究，取得了一系列的成果[8-11]。在上述研究工作基础上，一些实用性工具也开发出来了，如 Shrink[12]等。但是上述研究工作或者需要了解被检测系统的详细结构，或者相关对象之间的依赖关系难以观察到，这使得研究工作中的故障模型要么构造难度大，要么故障定位算法的复杂度高，这使得相关研究难以满足工程实践的需要。

## 1 问题分析与解决策略描述

为了控制软件设计的复杂性，设计者通常采用层次方式进行设计活动，而软件设计的体系结构中，一种最常见的结构就是软件的层次结构，这一结论常常用软件层次结构图来表示，该图中的构件与连接件主要实例之一可以是模块和模块之间的调用关系。每个模块通常由很多函数组成，而构成函数的成分主要就是代码段和代码行。与这一软件设计模型相对应的软件故障传播模型或者软件故障模型也是分层次的。在针对该分层软件故障模型实施一次软件测试之后会发现很多故障或者错误，我们需要解决的问题就是，如何以有效且快速的方式定位故障的出现位置，即首先将软件故障定位到某个模块，再由这个模块定位到某个函数，最后进一步定位到代码段或者代码行。

一个软件由很多模块构成，不同模块之间的互相调用是多对多的关系。一个模块由很多函数构成，不同的函数之间的互相调用也是多对多的关系。函数与代码之间大多是一对多的关系，但也存在多对多的关系，比如全局变量会影响所有调用该变量的函数。因此，软件的故障传播模型错综复杂，一行代码出错可能导致函数运行出错，进而导致多个模块结果出错，出现多个故障现象。本文研究的内容仅限于单一的故障原因导致单个或多个故障现象的情况，不考虑多个故障原因同时发生导致单个或多个故障现象的情况。

针对分层故障传播模型的故障定位的研究思路是，将分层故障传播模型划分成数个相邻两层情况来处理，因此，本文将以该故障模型中的任意相邻两层为例进行讨论。也就是说，对于每一次具体的故障定位而言，仅仅涉及到故障传播模型中的任意两层，并可将这两层用一个二分图来建模和表示，我们将这一简化后的模型称为基于二分图的故障检测模型，于是将基于该二分图故障检测模型的故障定位问题转化为求图的最小覆盖问题来解决。这样的做法简化了问题求解，从而简化了软件故障诊断活动和诊断过程。

现以从故障模块定位到故障函数的过程为例进行说明。在软件开发、测试及使用过程中会产生很多历史数据，分析这些历史数据可以得出故障模块与故障函数之间的对应关系及其相应比例。某个模块出现故障可能由某个故障函数导致，同时也可能由别的一些故障模块导致此模块无法正确运行。因此，由这些故障模块和故障函数可以构成一个复杂的故障传播模型的拓扑图，如图 1 所示。对图 1 中的拓扑图而言，我们将入度为 0 的节点称为故障函数，将其它节点称为故障模块。为使其具有一般性，我们将入度为 0 的节点统称为故障原因，将其它节点统称为故障现象。在图 1 中，虚线框外所示的节点即为故障原因，虚线框内所示的节点即为故障现象。将虚线框外入度为 0 的节点与其它节点分别转化到二分图的两个集合，故障传播模型的拓扑图转化后的二分图模型如图 2 所示。

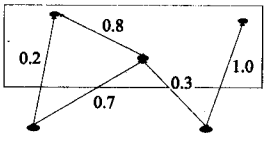


图 1 一个复杂的故障传播模型的拓扑图

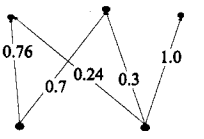


图 2 简化后的基于二分图的故障检测模型

这样，在分层的理论下，我们可以通过对被诊断软件进行建模来得到其故障传播模型的拓扑图，接下来进行故障定位处理的步骤是：1)将该拓扑图转化为基于二分图的故障检测模型；2)对该故障检测模型进行最小覆盖，实现故障的定位。下面从这两个方面进行详细阐述。

## 2 构建基于二分图的故障检测模型

为了定位某一软件故障，可以建立整个故障传播模型的拓扑图，该拓扑图可定义如下。

**定义 1**(故障传播模型的拓扑图) 该图为一个多元组 $TG=(N,P_{N,N})$，其中 $N$ 为节点集合，节点代表故障现象和故障原因，$P_{N,N}$ 为概率矩阵，表示某个节点由其他节点引起的概率。

在整个拓扑图中，设 $N$ 集合的元素个数为 $t$。由于故障原因是故障传播的根源，因此表征故障原因的节点入度为 0，表征故障现象的节点的入度表示引起该现象的概率值，所以其入度和一定为 1。$P_{N,N}$ 的值由专家分配或从历史数据估算得到。在整个软件生命周期中，从软件原型的诞生开始就包含了软件故障传播的记录数据，包括软件开发过程中的单元测试、软件测试过程中的详细测试、软件维护中的日志文件。通过这些记录文件中的故障原因与故障集合的对应比例，即可以建立当前软件故障传播的拓扑图模型。在软件生命周期中，随着记录文件的变化，拓扑图模型也会逐渐变化。在软件生命周期的初始阶段，由于记录数据比较少，误差会比较大。记录数据越多，模型的准确率就越高。但软件故障检测往往是针对软件使用和维护过程，其记录数据已经很大，从而保证了比较高的准确率。

故障传播模型的拓扑图转化为基于二分图的故障检测模型，该模型的定义如下。

**定义 2**(基于二分图的故障检测模型) 该模型为多元组 $BG=(U,D,P_{U,D})$，其中 $D$ 为故障原因；$U$ 为故障现象；$P_{U,D}$ 为故障原因与故障现象之间的关系矩阵，表示 $U_i$ 由 $D_j$ 引起的概率。假设故障现象 $U$ 集合的节点个数为 $m$，故障原因 $D$ 集合的节点个数为 $n$。

该二分图模型的构建来源于对上述复杂拓扑图的转化。将该拓扑图中入度为 0 的所有节点构成的集合对应到二分图中的故障原因 $D$ 集合中，将其他节点集合对应到二分图的故

障现象 $U$ 集合中，$u_i \in U$ 由 $d_j \in D$ 引起的概率 $P_{U,D}$ 可定义为所有起点为 $d_j$、终点为 $u_i$ 的路径的权值之和。根据概率论中的乘法定理，$P(AB)=P(A|B)P(B)$，则每条路径的权值的计算方法是这条路径所有连线上的权值的乘积。拓扑图中所有入度不为 0 的节点，其入度和均为 1。由于所有的出度均可以分散到其全部的入度中去，因此转化之后的二分图的故障现象 $U$ 集合中所有节点的入度和也为 1。

将一个软件分层以后可以构造出数目为层数－1 的拓扑子图模型，在每两层之间建立一个拓扑图模型。然而，在实际软件诊断过程中一般只会显示其中的一部分故障现象，那么对于每个拓扑图来说，只要抽取可能的故障现象与故障原因构成一个拓扑子图即可。但如果只提取检测到的故障现象及其相关的故障原因而不改变其相应的对应比例，会导致表示故障现象的节点入度和不为 1 的情况。所以抽取子图的过程中需要进行特殊的处理，即每次删掉一个节点就需要更新该节点指向的节点的入度。具体算法描述如图 3 所示。

算法 1　子图抽取算法

输入：软件的故障传播拓扑图模型和检测到的故障现象；

输出：只包含检测到的故障现象的子图；

Begin

1. 找出一个未检测到的故障现象节点 i；
2. 找到一个 i 所指向的节点 j，i 指向 j 的线上的权值为 $PN_i,N_j$；
3. 对所有指向节点 j 的节点 k，连线上的值 $PN_k,N_j = PN_k,N_j/(1-PN_i,N_j)$；
4. 删除节点 i 与节点 j 之间的连线；
5. 返回步骤 2，直到找不到节点 j 为止；
6. 返回步骤 1，直到找不到节点 i 为止；

End

图 3　子图抽取算法

现分析该算法的时间复杂度如下。该算法的关键步骤在第 3 步，扫描 $i$、$j$、$k$ 为嵌套的 3 重循环，在最坏情况下 3 者的复杂度都为 $t$，则该算法的复杂度为 $O(t^3)$。

经过子图抽取算法得到的子图中，所有的故障现象都是检测到的故障现象，所以，子图可以直接转化为二分图进行处理。具体算法描述如图 4 所示。

算法 2　将子图转化为二分图算法

输入：前一个过程得到的子图；

输出：一个二分图模型；

Begin

1. 找到所有的入度为 0 的节点，将其放到故障原因 D 集合中；
2. 将其他的节点放到故障现象 U 集合中；
3. 概率矩阵 $P_{U,D}$ 中所有元素初始化为 0；
4. 将故障原因 D 集合中所有节点标记为 false；
5. 在故障原因 D 集合中找到一个标记值为 false 的节点 i；
6. 以 i 为初始节点进行深搜，为每个节点保存一个临时变量 num 值，i 处的 num 值为 1；
7. 每搜过一条线，目标节点 j 的 num 值即更新为上一个节点的 num 值乘以线上的权值，将二分图中 $P_{i,j}$ 的值加上 j 处的 num 值；
8. 深搜遍历完成后，将 i 节点标记值改为 true；
9. 返回步骤 5，直到遍历完故障原因 D 集合；

End

图 4　将子图转化为二分图算法

现分析该算法的时间复杂度如下。该算法的关键步骤为深度优先搜索过程，在最坏情况下，针对故障原因 $D$ 集合的每个元素均要遍历故障现象 $U$ 中所有的元素，所以该算法的时间复杂度为 $O(m*n)$。

## 3　基于二分图的故障检测模型的最小覆盖算法实现故障定位

在得到基于二分图的故障检测模型之后，故障定位问题就转化为在该二分图的故障原因 $D$ 集合中找出错误假设集合来解释故障现象 $U$ 集合中的所有故障现象。为解决这个问题，提出了一个基于贪心策略的算法，该算法对故障原因 $D$ 集合中的所有节点 $D_i$ 按照与之连线的权值和，尽量使用贪心策略，优先选取权值和最大的节点，将其添加到错误假设集合 $Fs$ 中，然后从该二分图中去掉该节点以及与该节点相连的节点，重复这个过程，直到没有故障现象为止。具体算法描述如图 5 所示。

算法 3　实现故障定位的基于二分图故障检测模型的最小覆盖算法

输入：算法 2 所构建的二分图模型；

输出：错误假设集合；

Begin

1. 设错误假设集合 Fs=Ø；
2. 计算故障原因 D 集合中每个节点的出度和 $E_i$：$E_i=\sum_{j=1}^{m} P_{U_j,D_i}$；
3. 找出出度和 E 最大的节点 i，将节点 i 加入错误假设集合 Fs 中；
4. 删除节点 i，并删除所有与节点 i 有连线的节点；
5. 如果故障现象 U 集合不为空，返回步骤 2，直到故障现象 U 集合为空为止；

图 5　实现故障定位的基于二分图故障检测模型的最小覆盖算法

## 4　算法仿真分析

本文的算法仿真数据基于对某舰载机系统试运行阶段的数据。对该系统设立了上万个监控点，在试运行的大半年时间里搜集了大量数据。将这些数据里面相对独立的小系统的数据提取出来转换成拓扑图模型和相应的真实故障集合的数据，这样就构建了实验所需的仿真场景。仿真场景建立好之后，就按照本文的上述算法进行故障定位仿真。共运行 1000 个仿真场景。

仿真程序是在 2.67GHz Intel core i5 CPU，2G 内存的计算机上运行的。实验结果出来之后，通过对真实故障集合与假设故障集合的对比，来分析基于二分图故障模型的软件故障定位算法。以下部分通过一个简单的例子来阐述实验过程。

仿真实验的输入数据格式如图 6 所示，第一行是一个数字 $n$，表示拓扑图有 $n$ 个节点，接着是一个 $n*n$ 的概率矩阵，最后是一个数字 $m$，表示检测到的故障现象的个数，然后是 $m$ 个故障现象的编号，接着是一个数字 $p$，表示真实的故障原因个数，最后是这 $p$ 个故障原因的编号。图 6 中的数据对应的拓扑图如图 7 所示。

```
7
0 0 0 0.4 0.1 0.3 0
0 0 0 0 0.9 0.7 0.2
0 0 0 0 0 0 0.7
0 0 0 0 0 0 0
0 0 0 0.6 0 0 0
0 0 0 0 0 0 0.1
0 0 0 0 0 0 0
3
4 6 7
1
2
```

图 6　仿真场景的输入数据格式

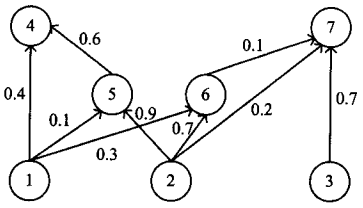


图 7　图 6 中的数据对应的拓扑图

根据图 6 中的拓扑图和故障现象描述，采用子图抽取算法抽取的子图如图 8 和图 9 所示。

6

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0.46 | 0.3 | 0 |
| 0 | 0 | 0 | 0.54 | 0.7 | 0.2 |
| 0 | 0 | 0 | 0 | 0 | 0.7 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0.1 |
| 0 | 0 | 0 | 0 | 0 | 0 |

图 8　拓扑图子图的数据

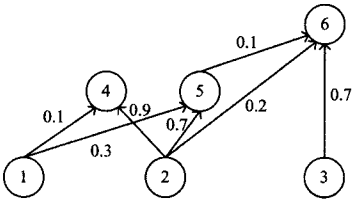


图 9　拓扑图子图

通过拓扑图模型转化成二分图模型的算法将拓扑子图转化为二分图，如图 10 和图 11 所示。然后通过基于贪心的最小覆盖算法求得其故障原因只有一个，即图中的 2 节点，此例中结果也与输入数据一致。

| | | |
|---|---|---|
| 3 | 3 | |
| 0.46 | 0.3 | 0.03 |
| 0.54 | 0.7 | 0.27 |
| 0 | 0 | 0.7 |

图 10　二分图模型的数据

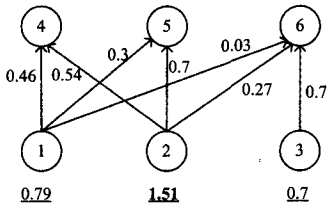


图 11　二分图模型

本文使用了如下 3 个指标来评估本文提出的软件故障定位算法：检测率(Detection Rate，简称 DR)、误测率(False Detection Rate，简称 FDR)、检测率方差(Variance of Detection Rate，简称 VDR)，这 3 个指标各自的定义如下：

$$DR=\frac{|Fr\cap Fs|}{|Fr|} \tag{1}$$

$$FDR=\frac{|Fs-Fr|}{|Fs|} \tag{2}$$

$$VDR=\frac{\sum_{i=1}^{N}(DR_i-\sum_{i=1}^{N}DR_i/N)^2}{N} \tag{3}$$

式(1)－式(3)中，$Fr$ 表示实际的故障集合，$Fs$ 表示用以上方法得到的故障假设集合，$DR_i$ 表示第 $i$ 个仿真场景得到的检测率，$N$ 表示仿真场景的个数。

对这 1000 个场景的数据进行实验，得到检测率均值 $\overline{DR}=63.56\%$，误测率均值 $\overline{FDR}=12.90\%$。检测率方差 $VDR=0.24$。

## 5　相关工作对比

本文研究工作受到北京邮电大学网络与交换国家重点实验室的黄晓慧[8]研究工作的启发，研究也采用了二分图模型及其相关处理方法，同时我们的实验所用的评估指标与文献[8]一致。我们的研究工作与文献[8]的不同之处在于：(1)研究对象不同，文献[8]的研究对象是 Internet 服务故障，我们的研究对象是软件；(2)研究策略与方法不同，由于网络故障中故障原因与表现出来的故障现象之间的关系相对简单，因此可以直接采用二分图模型来处理。但对软件而言，仅以软件中的函数调用为例，其函数调用关系就十分复杂，因此在软件故障定位中很难直接建立起基于二分图的故障传播模型。我们的做法是首先建立软件故障的拓扑图模型，然后将基于拓扑图的软件故障模型转化成基于二分图的软件故障检测模型，从而达到故障定位和求解故障原因的假设集合的目的。

**结束语**　本文分析了软件测试软件故障定位中存在的困难，提出了将软件测试中常见的拓扑图故障传播模型转化为二分图故障传播模型，提出了以贪心策略为基础的算法来解决二分图故障定位问题，并通过简单的仿真实验对其进行了检验。由于本文在构造拓扑图模型时没有考虑若干个故障原因共同导致某个故障现象的情况，而在实际情况，这种情况是可能存在的，因此，在进一步的研究工作中，我们将考虑这一情况，并对相应的二分图最小覆盖算法进行改进。

## 参 考 文 献


[1] Bouloutas A T, Calo S, Finkel A. Alarm correlation and fault identification in communication networks[J]. IEEE Transactions on Communications, 1994, 42(234): 523-533

[2] Katzela I, Schwartz M. Schemes for Fault Identification in Communication Networks[J]. IEEE/ACM Transactions on Networking, 1995, 3

[3] Steinder M, Sethi A S. Probabilistic Fault Localization in Communication Systems Using Belief Networks [J]. IEEE/ACM Transactions on Networking, 2004, 12(5)

[4] Hood C S, Ji Chuan-yi. Proactive network fault detection[C]// Proceedings of the IEEE INFOCOM. Kobe, Japan, April 1997

[5] Yemini S A, Kliger S, Mozes E, et al. High speed and robust event correlation[J]. Conununications Magazine, 1996, 34(5): 82-90

[6] Fischer W, Xie G, Young J. Cross-domain fault localization: a case for a graph digest approach [C]// IEEE Third Workshop on Internet Network Management (INM'08). Orlando, FL, USA, 2008: 1-6

[7] Khanna G, Cheng M Y, Varadharajan P, et al. Automated rule-based diagnosis through a distributed monitor system [J]. IEEE Transactions on Dependable and Secure Computing, 2007, 4(4): 266-279

[8] 黄晓慧，邹仕洪，褚灵伟，等. Internet 服务故障管理：分层模型和算法[J]. 软件学报，2007，18(10)：2584-2594

[9] 褚灵伟，邹仕洪，程时端，等. 一种动态环境下的互联网服务故障诊断算法[J]. 软件学报，2009，20(9)：2520-2530

[10] Chu LW, Zou S H, Cheng S D, et al. Active probing based Internet service fault management in uncertain and noisy environment [J]. Science in China Series F: Information Sciences, 2008, 51(11): 1857-1870

[11] 褚灵伟，邹仕洪，程时端，等. 多域服务环境下的分布式故障诊断算法[J]. 电子与信息学报，2010，32(4)：836-840

[12] Kandula S, Katabi D, Vasseur J-P. Shrink: A Tool for Failure Diagnosis in IP Networks[C]// Sigcomm2005 MineNet Workshop. 2005