基于故障关联的多故障概率诊断方法

徐俊洁 陈 荣
(大连海事大学信息科学技术学院 大连 116026)

摘 要 故障软件诊断的必要性在于真实世界中的软件几乎都会包含一个以上的故障。与单故障不同,多个故障的传播及其关联导致软件诊断更复杂,不确定性更高,概率推理因而被用于适应多故障程序的特殊性。提出了一种新的基于变形概率图FCG及其推理的软件诊断方法。相比于BARINEL方法和经典的贝叶斯网,FCG的特别之处在于采用了无向图上候选故障及其关联关系的贝叶斯推理和Noisy-or推理,而候选故障及其关联可以从程序语句间的控制依赖关系和数据依赖关系中创建。从西门子套件到更大的space,grep程序的实验,无论是在处理单故障还是处理多故障的情况下,实验结果都证明了FCG的有效性,其诊断效果比LOUPE,Ochiai,Tarantula甚至BARINEL方法都准确。

关键词 多故障,故障关联与不确定性,概率推理,控制和数据依赖关系,变形概率图
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.027

Probabilistic Diagnosis Approach to Diagnosing Multiple-fault Programs with Fault Correlation

XU Jun-jie CHEN Rong
(College of Information Science and Technology, Dalian Maritime University, Dalian 116026, China)

Abstract Diagnosing multiple-fault software is necessary because almost all real-world software contains more than one fault. Unlike single-fault, the propagation and correlation of multiple faults in software lead to more complexity and great uncertainty, and probabilistic reasoning is thus applied to accommodate such uniqueness. This paper proposed a new probabilistic reasoning method to diagnose multiple-fault software by using variant probabilistic graphs FCG and their inference. Distinguished from the BARINEL method and the classical Bayesian network, FCG features Bayesian and Noisy-or inference from undirected graph consisting of candidate faults and their correlation which can be set up from statement-level control and data dependencies. Experiments were conducted on programs ranging from Siemens suite to larger ones like space and grep. The experimental results validate the effectiveness of the present approach in handling programs no matter with single fault and with multiple faults, and especially it is more accurate than competitors such as LOUPE, Tarantula, Ochiai and even BARINEL.

Keywords Multiple-fault, Fault correlation and uncertainty, Probabilistic reasoning, Control and data dependencies, Variant probabilistic graph

1 引言

近年来,自动化软件调试研究展示了在各种情况下诊断真实软件错误的能力,其中基于测试的故障定位技术(Test-Based Fault Localization, TBFL)更是由于能用较低的代价获得较高的单故障诊断效率而成为当前研究的热点之一。然而TBFL方法因对执行覆盖信息敏感而难以用于多故障程序诊断,Nicholas等人的多故障程序定位的实验研究表明TBFL方法对于"one-fault-at-one-time"模式有效,但对于"many-fault-at-one-time"模式的效果并不好[7]。对于多故障程序,执行错误语句的成功测试用例普遍存在,故障传播及其关联导致测试用例执行信息更加复杂和多变,这些不确定性都严重影响了诊断效果。

针对软件诊断的不确定性问题,近年来一些研究者把目光转向了基于概率图模型的故障诊断。最早出现的有PPDG[3]和EFG[4]等概率图模型,不过它们都假设程序中只存在一个故障。针对多故障程序,Abreu基于人工智能领域的一致性诊断思想,综合运用TBFL的Ochiai度量[5]和贝叶斯推理,提出了多故障程序诊断的BARINEL方法[6]。其做法是找到失败运行导致的冲突,通过计算冲突集合的碰集得到诊断候选,再使用标准贝叶斯定理计算诊断候选的可疑度。从理论上讲,PPDG和EFG使用的都是标准的贝叶斯网,它们仍属于浅层次的概率推理。相比之下,BARINEL的一致性诊断推理方式更复杂,不过由于模型化时仅仅考虑语句的

到稿日期:2015-11-30 返修日期:2016-03-10 本文受国家自然科学基金(61175056)资助。
徐俊洁(1985—),女,博士生,主要研究方向为故障诊断、软件工程,E-mail:connyadmin@163.com;陈 荣(1969—),男,博士,教授,主要研究方向为故障诊断、智慧环境及行为识别、规划调度、互联网与移动计算等。

执行路径信息，其诊断性能仍然存在改进的空间。

实际软件测试中，很多缺陷/故障是相互作用的，它们之间往往存在某种关联关系。包晓安等人[11]使用马尔科夫链测试模型验证了运用缺陷之间的关联信息选择测试用例集合可以得到更好的测试结果。本文研究的诊断模型也重点考虑可能的故障关联关系，为此提出一个变形的概率图模型——故障关联图(Fault Correlation Graph，FCG)，通过该模型上的深层概率推理机制择优定位多个故障。本文的 FCG 是通过程序运行时得到的语句间的控制依赖关系和数据依赖关系来发现故障语句之间的可能关联关系，进而衍生出一种只包含两层结点的无向概率图模型。本文的主要贡献如下：

(1)本文的 FCG 是一种区别于标准贝叶斯网的概率图模型，它是由语句间的控制依赖和数据依赖关系衍生而来的关联关系，用来描述故障语句之间的相互作用而非单向影响。

(2)本文提出的概率推理算法是一种把 Noisy-or 推理和标准贝叶斯推理相结合的深层次推理技术，在计算多故障程序的语句可疑度时更加精确。

(3)实验验证了无论在单故障情况还是多故障情况下本文提出的多故障诊断技术的故障诊断精确度都比 Tarantula[1]，Ochiai[5]，LOUPE[8]甚至 BARINEL[6]更高。

2　诊断模型

在实际测试多故障程序的过程中，由于故障之间存在关联关系，当某条程序语句是观测失效点时(即该语句见证了故障)，它往往不是引发程序失效的根本原因(即实际故障的语句)。因此需要从该语句出发进行关联分析，推断实际故障的位置或引发故障的原因。为此，在所提的诊断模型中，每条语句 s_i 都应有两个相关的变量 R_i 和 A_i，分别用于刻画该语句的可能状态：R_i 称为实际故障的结点，表示 s_i 为实际故障语句的隐状态变量；A_i 称为可观测的结点，表示 s_i 是可观测的语句。

定义 1　多故障程序的可观测结点的状态分为两种：观测失效和观测正常。

当一条程序语句检测到故障时(观测失效)，需要找到引发该故障的实际故障语句，症结可能是该语句本身，也可能是实际故障语句通过其它相干语句而引起的。这种观测失效的语句和相干语句之间的关系就是故障关联关系。与依赖关系不同，故障关联关系是相互的，不是有向的。而贝叶斯网络是一个有向图，更适合用来描述依赖关系。因此本文使用无向图来建立如下诊断模型。

定义 2　故障关联图 FCG 可以用一个三元组$\langle G'(V,E), P, OBS_p\rangle$表示，其中：

(1)$G'(V,E)$是一个两层结构的无向图，其中，$V(G')=A(G')\cup R(G')$是结点的集合，$R(G')$是位于顶层的结点的集合，$A(G')$是位于底层的结点的集合，$E(G')$表示从 $R(G')$到 $A(G')$的无向边的集合；

(2)P 是 FCG 上的概率分布；

(3)OBS_p 是 FCG 上可观测的结点的观测值集合 $O(G')$。

图 1 中，$R(G')=\{R_1,R_2,\cdots,R_m\}$是所有实际故障的结点集合，$A(G')=\{A_1,A_2,\cdots,A_m\}$是所有可观测的结点的集合，$E(G')$表示实际故障的结点和可观测的相干结点之间的故障关联关系。FCG 模型中，每一个结点除了可以表示程序语句，也可以用其他程序实体来代替，例如语句块、函数/方法、类等。对于不同的程序实体，FCG 描述不同程序实体间的故障关联关系。本文的研究是将语句作为程序实体。

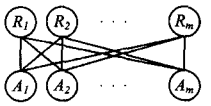

图 1　一个关于 FCG 模型的例子

图 2 描述了如何使用 FCG 诊断多故障程序，具体包括：建模过程以及推理过程。

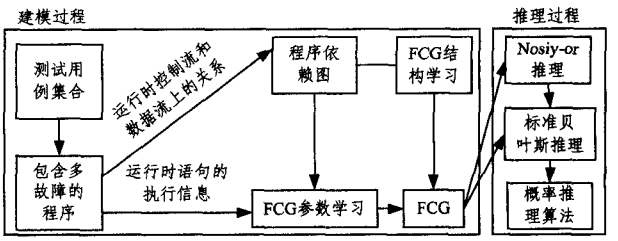

图 2　FCG 诊断多故障程序的大体思路

建模过程分为 4 步：1)使用测试用例集合运行多故障程序；2)根据运行时得到的控制流和数据流上的依赖关系得到程序依赖图；3)利用程序依赖图确立故障语句间的关联关系集合，即故障关联关系，实现 FCG 的结构学习；4)收集程序运行测试用例时的路径执行的语句覆盖信息，利用程序依赖图实现 FCG 上的参数学习。

推理过程包括两个阶段：首先在 FCG 上实现 Noisy-or 推理，然后在此基础上使用标准贝叶斯推理以找到最有可能引发程序失效的实际故障的语句。

(1)Noisy-or 推理

根据 Noisy-or 模型[10]的定义，本文假设 FCG 模型的底层和顶层结点之间是满足 Noisy-or 模型的。在 FCG 模型上，Noisy-or 推理的过程是使用式(3)来实现的。其中，$P(R_i)$表示实际故障结点 R_i 的先验概率；$P(A_j)$表示可观测结点 A_j 的概率；$CS(A_j)$表示与 A_j 存在故障关联的实际故障结点的集合，也即 A_j 的关联关系集合。

定义 3(关联关系集合，Correlation Set)　对于一条语句 s_i(一个可观测结点 A_i)，它的关联关系集合是与它存在故障关联的语句(实际故障结点)的集合，用 $CS(s_i)$($CS(A_i)$)来表示，其中，$CS(s_i)$中的语句和 $CS(A_i)$中的实际故障结点是一一对应的关系。

$$P(A_j)=1-\prod_{R_i\in CS(A_j)}(1-P(R_i)) \tag{1}$$

(2)标准贝叶斯推理

在 Noisy-or 推理的基础上，标准贝叶斯推理利用 FCG 模型上的概率分布，包括先验概率 $P(R_i)$和条件概率 $P(A_j|R_i)$，由式(2)实现。

$$P(R_i|A_j)=\frac{P(A_j|R_i)*P(R_i)}{P(A_j)} \tag{2}$$

接下来讨论如何实现建模和推理。

3 建模过程

作为一种无向网络，FCG 的建模过程分为 FCG 结构学习和 FCG 参数学习两部分。由 FCG 的定义可知，构建 $G'(V,E)$ 即为 FCG 结构学习的过程，计算概率分布 P 和观测值 $OBSp$ 即为 FCG 参数学习的过程。下面将详细进行介绍。

3.1 FCG 结构学习

对 FCG 结构的学习就是寻找故障关联的过程，也即确定实际故障结点和可观测结点之间的关联关系的过程，本文借助动态程序依赖图(Program Dependence Graph，PDG)来发现这些关联关系。

PDG 是由程序运行测试用例时语句间的控制流和数据流上产生的依赖关系建立的，描述了程序语句间的结构和行为[7]。PDG 可以表示为一个有向图 $G(V,E)$，其中，$V(G)$ 是 PDG 中结点的集合，每个结点表示一条语句；$E(G)$ 是 PDG 中有向边的集合，$E(G)$ 中每一条边表示一个语句间的依赖关系。根据程序运行时控制流和数据流上的语句间的依赖关系，PDG 分为控制依赖图($G_c(V,E)$)和数据依赖图($G_d(V,E)$)，其中 $E(G_c)$ 和 $E(G_d)$ 分别表示语句间的控制依赖关系和数据依赖关系的边的集合。

PDG 包括控制依赖图和数据依赖图，在 FCG 结构学习的过程中，使用控制依赖图(数据依赖图)来获得控制流(数据流)上语句的关联关系集合，生成基于控制依赖的故障关联图，简称 Control-FCG($G_c'(V,E)$)(或基于数据依赖的故障关联图，简称 Data-FCG($G_d'(V,E)$))。其中，与 $E(G_d')$ 关于数据流的情况类似，$E(G_c')$ 表示 $A(G_c')$ 和 $R(G_c')$ 之间的无向边的集合，即可观测的结点和实际故障的结点之间在控制流上的关联关系。

在 PDG 中，任意结点 i 对应语句 s_i 的故障关联关系用该结点的父结点(直接前驱结点)、孩子结点(直接后继结点)和该结点本身组成的集合来描述。

算法 1 描述了应用 PDG 中结点的关联关系集合构建 FCG 模型的过程，它的输入是 $G(V,E)$，输出是 $G'(V,E)$。如果 $G_c(V,E)$ 作为算法 1 的输入，输出便是 $G_c'(V,E)$。同样地，如果 $G_d(V,E)$ 作为算法 1 的输入，输出则是 $G_d'(V,E)$。

算法 1 生成 FCG 模型

输入：G(V,E)

输出：G′(V,E)

1. FOR each i∈V(G) do
2. A(G′) ← A_i;
3. R(G′) ← R_i;
4. IF s_j∈CS(s_i) then
5. E(G′) ←⟨R_j,A_i⟩;
6. RETURN G′(V,E);

以输入 $G_d(V,E)$ 为例对算法 1 进行说明：

1)在步骤 1—步骤 3 中，对 $V(G_d)$ 中每一个结点 i 对应的语句 s_i 生成一个 A_i 结点和 R_i 结点；

2)在步骤 4—步骤 6 中，对 $A(Gd')$ 中每一个 A_i，在 G_d 中计算语句 s_i 的关联关系集合 $CS(s_i)$，对 $CS(s_i)$ 中每一个语句对应的结点 R_j 生成一条从 R_j 到 A_i 的无向边。

3.2 FCG 参数学习

3.2.1 计算 FCG 的观测值集合 $O(G')$

由定义 1 可知，对于一个可观测结点，它或者是观测失效的，或者是观测正常的。本文中，每个可观测结点 A_i 存在两种观测值：0(观测正常)和 1(观测失效)，分别用 $status(A_i)=0$ 和 $status(A_i)=1$ 来表示。观测值集合 $O(G')=F(G')\cup H(G')$，其中，$F(G')=\{A_i|status(A_i)=1,A_i\in G'\}$，$F(G')=\{A_i|status(A_i)=0,A_i\in G'\}$，并且满足 $F(G')\cap H(G')=\emptyset$。当 $status(A_i)=0$ 时，与 s_i 存在故障关联关系的语句被怀疑为实际故障的可能性会降低；当 $status(A_i)=1$ 时，与 s_i 存在故障关联关系的语句被怀疑为实际故障的可能性会增大。Control-FCG(Data-FCG)中 A_i 的观测值与控制依赖图(数据依赖图)中结点 i 的观测值以及语句 s_i 的观测值是一致的，即 $status(A_i)=status(i)=status(s_i)$。

接下来讨论如何确定可观测结点的观测值。很显然，在程序中返回语句可以用来计算可观测结点的观测值。但是，一些返回语句不能在控制依赖图和数据依赖图中同时出现，而本文需要利用 PDG 来确定 FCG 的观测值集合。因此，在 PDG 中引入“虚拟结点”的概念，对每一个运行结果与预期不一致的测试用例的返回语句 s_i 生成一个对应的“虚拟结点”$\_ret_i$，具体思路如下：

Step1 首先判断在控制依赖图或数据依赖图中是否存在与返回语句 s_i 对应的结点 i，若存在，则直接生成一条从结点 i 到 $\_ret_i$ 的边，否则执行 Step2；

Step2 根据执行信息，从 s_i 回溯语句的执行路径，直到在控制依赖图或数据依赖图中找到与 s_i 对应的结点 j，生成一条从结点 j 到 $\_ret_i$ 的边。

执行以上两个步骤后，PDG 变成了带“虚拟结点”的 PDG。本文假设“虚拟结点”的父结点的观测值为 1，除“虚拟结点”外的叶子结点的观测值为 0。

命题 1 如果在带“虚拟结点”的 PDG 中存在一个节点 j 满足 $j\in parent(\_ret_i)$，那么 $status(j)=1$。

证明：在带“虚拟结点”的 PDG 中，对于 $\_ret_i$，存在两种情况：1)存在与 $\_ret_i$ 对应的结点 i；2)不存在结点 i。对于情况 1)，由于结点 i 和 $\_ret_i$ 表示同一条语句 s_i，满足 $status(i)=status(\_ret_i)$，若返回语句 s_i 观测到 $status(s_i)=1$，那么结点 i 的观测值即为 1。对于情况 2)，由于 PDG 中不存在结点 i，需根据执行信息从 s_i 回溯语句的执行路径，直到找到距语句 s_i 最近的语句 s_j，而且满足语句 s_j 在 PDG 中存在对应的结点 j。由于语句 s_i 不在 PDG 中，因此用距 s_i 最近的语句 s_j 来代替，满足 $status(s_j)=status(s_i)$。根据 $status(s_i)=status(\_ret_i)=1$，可以得到 $status(s_j)=1$。

3.2.2 计算 FCG 上的概率分布

首先，计算 FCG 顶层的实际故障结点的先验概率分布，用一个概率分布向量 $V=(P(R_1),P(R_2),\cdots,P(R_m))$ 来表示，其中 $P(R_i)$ 表示实际故障结点 R_i 的先验概率，也即语句

s_i 实际上是故障语句的初始概率，$1 \leqslant i \leqslant m$。在式(3)中，$N_f$ 表示失败测试用例的总个数(如果测试用例的实际输出与预期的结果不一致，那么它就是失败测试用例，否则就是成功测试用例)。如果语句 s_i 在测试用例 t_j 中被执行，且 t_j 是一条失败测试用例，那么$((e_j=1)\cap(a_{ij}=1))=1$，否则等于 0。同样地，如果语句 s_i 在测试用例 t_j 中被执行，且 t_j 是一条成功测试用例，那么$((e_j=0)\cap(a_{ij}=1))=1$，否则等于 0。

$$P(R_i)=\frac{\sum_{j=1}^{N}((e_j=1)\cap(a_{ij}=1))}{\sqrt{(\sum_{j=1}^{N}((a_{ij}=1)\cap(e_j=0))+\sum_{j=1}^{N}((e_j=1)\cap(a_{ij}=1)))*N_f}} \tag{3}$$

从实际故障结点 R_i 到结点 A_j 的条件概率分布 $P(A_j|R_i)$由式(4)计算，其中，N 是测试用例的总个数，N_f 是错误运行的测试用例个数，N_p 是正确运行的测试用例个数。

$$P(A_j|R_i)=\begin{cases}\dfrac{N^3-N_f}{N^3}, & status(A_j)=1\\ \dfrac{N_p}{N^3}, & status(A_j)=0\end{cases} \tag{4}$$

4　推理过程

FCG 模型上的推理过程分为两个阶段，第一个阶段是在 FCG 模型上进行 Noisy-or 推理，目的是根据实际故障结点的先验概率计算可观测结点的概率。第二个阶段是根据可观测结点的概率、实际故障结点的先验概率以及从实际故障结点到可观测结点的条件概率，利用标准贝叶斯推理，计算实际故障结点的后验概率 $P(R_i|A_j)$。当完成全部的推理后，就得到一个更新后验概率后的 FCG 顶层的结点概率分布 V。

由于 FCG 模型包括 Control-FCG 和 Data-FCG，因此在推理过程中需要在两个子模型中分别实现计算后验概率的推理过程。算法 2 描述了 FCG 算法的具体过程，它的输入是 FCG，输出是向量 V'。在算法 2 中，V_c (V_d)是 Control-FCG (Data-PCE)上顶层结点的先验概率分布 V，由式(1)初始化。根据 Control-FCG 和 Data-FCG 上的观测值集合，分别更新 V_c 和 V_d。最终，由函数 $max_R(V_c,V_d)$比较 V_c 和 V_d 中每一个 $P(R_i)$，并把较大的 $P(R_i)$赋值给向量 V'。

算法 2　概率推理算法

输入：FCG

输出：概率向量 V′

1. 初始化 V_c 和 V_d；
2. FOR each $A_j\in(O(G'))$ do
3. 　$P(A_j)$←由式(3)计算；
4. FOR each $A_j\in O(G_d')\cup O(G_c')$ do
5. 　WHILE $\exists R_i|((R_i,A_j)\in E(G_d'))$ do
6. 　$V_d(R_i)$←由式(4)更新 V_d；
7. 　WHILE $\exists R_i|((R_i,A_j)\in E(G_c'))$ do
8. 　$V_c(R_i)$←由式(4)更新 V_c；
9. V'←max_R(V_c, V_d)；
10. RETURN V′；

4.1　关于推理过程的小例子

图 3 描述了一段包含 12 条语句的 3 个整数求平均值的多故障程序 mid()。它的语句 s_2 和 s_4 是有故障的，正确表达应该是 $m=z(s_2)$和 $m=y(s_4)$。当使用给定的 5 条测试用例执行 mid()程序时，如果一条测试用例 t_i 执行了一条语句 s_j，则在执行信息矩阵中记为 $a_{ij}=1$，否则 $a_{ij}=0$。其中 t_4 和 t_5 是失败测试用例($e_4=e_5=1$)；t_1，t_2 和 t_3 是成功测试用例($e_1=e_2=e_3=0$)。

mid()程序

```
int mid() {
int x,y,z,m;
s1:read(x,y,z);
s2:m=z−1;
s3:if(y<z)
s4:  if(x<y+2)
s5:    m=y;
s6:  else if(x<z)
s7:    m=z;
s8:else if (x>y)
s9:  m=y;
s10:else if(x>z)
s11:  m=x;
s12:print(m); }
```

测试用例(x,y,z)

$t_1=(4,2,6)$, $t_2=(5,4,3)$, $t_3=(3,5,1)$, $t_4=(5,3,4)$, $t_5=(1,5,2)$

语句的执行信息

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	e_i
t_1	1	1	1	1	1	0	0	0	0	0	0	1	0
t_2	1	1	1	0	0	0	0	1	1	0	0	1	0
t_3	1	1	1	0	0	0	0	1	0	1	1	1	0
t_4	1	1	1	1	1	0	0	0	0	0	0	1	1
t_5	1	1	1	0	0	0	0	1	0	1	0	1	1

图 3　软件诊断问题的一个例子

在 mid()中，控制依赖关系是指语句 s_4 和 s_8 是否执行由 s_3 的谓词'if'的判断结果决定，用(s_3,s_4)和(s_3,s_8)表示。语句 s_2 使用的变量 z 是在 s_1 语句中被赋值的，且从 s_1 到 s_2 没有其他的语句对 z 的值进行修改，所以(s_1,s_2)就是一个数据依赖关系。mid()程序共有 8 个语句间的控制依赖关系以及 15 个数据依赖关系。根据这些依赖关系，图 4 示出了相应的控制依赖图(G_c)和数据依赖图(G_d)。其中，结点 i 对应语句 s_i，每条有向边代表一个依赖关系，例如控制依赖图中的边 8→9对应控制依赖关系(s_8,s_9)。

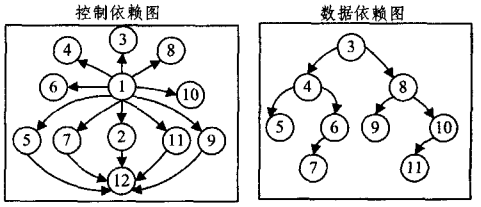

图 4　程序 mid()的程序依赖图

在程序依赖图中，对每个结点 i 计算它的关联关系集合 $CS(i)$，例如，在控制依赖图中 $CS(5)=\{4,5\}$。根据这些关联关系建立 FCG 模型，如图 5 所示。控制依赖图结点 5 对应 Control-FCG 中的 A_5 和 R_5。由于 $CS(5)=\{4,5\}$，在 Control-FCG 中存在两条无向边，分别连接结点 R_4 和 R_5 到结点 A_5。

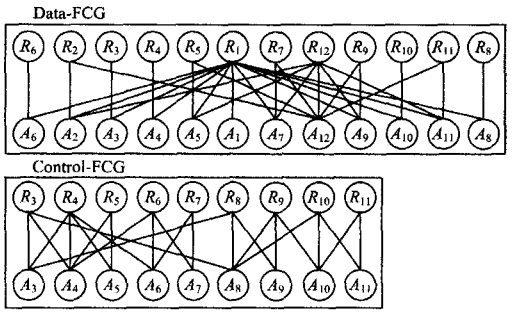

图 5 程序 mid() 的故障关联图

接下来，利用带“虚拟结点”的 PDG(见图 6)计算 FCG 上的观测值和概率分布。在图 6 中，满足 $status(i)=0$ 的集合为{7,9,11}，满足 $status(i)=1$ 的集合为{5,10}。相应地，在 Control-FCG 中，观测值集合为：$H(G_c')=\{A_7,A_9,A_{11}\}$，$F(G_c')=\{A_5,A_{10}\}$。同理可以得到 Data-FCG 中观测值集合，$H(G_d')=\{A_3,A_4,A_6,A_8,A_{10}\}$，$F(G_d')=\{A_{12}\}$。Control-FCG 和 Data-FCG 具有相同的概率分布，具体结果如表 1 所列。

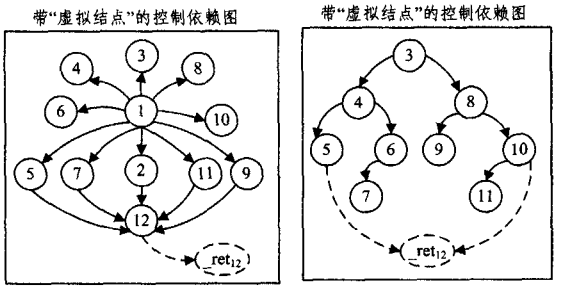

图 6 程序 mid()的带“虚拟结点”的 PDG

表 1 Control-FCG 的推理过程

参数学习	概率分布	$V_c=(0.63,0.63,0.63,0.50,0.50,0.00,0.00,0.41,0.00,0.50,0.00,0.63)$ $status(A_j)=0$ 时，$P(A_j\|R_i)=0.02$ $status(A_j)=1$ 时，$P(A_j\|R_i)=0.98$
	观测值	$H(G_c')=\{A_7,A_9,A_{11}\}$，$F(G_c')=\{A_5,A_{10}\}$
概率推理	A_9	$V_c=(0.58,0.58,0.58,0.41,0.50,0.00,0.00,\mathbf{0.02},\mathbf{0.00},0.50,0.00,0.58)$
	A_5	$V_c=(0.58,0.58,0.58,\mathbf{0.98},\mathbf{0.98},0.00,0.00,0.02,0.00,0.50,0.00,0.58)$
	A_7	$V_c=(0.58,0.58,0.58,0.98,0.98,\mathbf{0.00},\mathbf{0.00},0.02,0.00,0.50,0.00,0.58)$
	A_{10}	$V_c=(0.58,0.58,0.58,0.98,0.98,0.00,0.00,\mathbf{0.04},0.00,\mathbf{0.96},0.00,0.58)$
	A_{11}	$V_c=(0.58,0.58,0.58,0.98,0.98,0.00,0.00,0.04,0.00,\mathbf{0.04},\mathbf{0.00},0.58)$

表 1 也列出了在 Control-FCG 上更新概率向量 V_c 的具体过程，在 Data-FCG 上更新 V_d 的过程与此类似。以 $H(G_c')$ 中 A_9 为例，首先通过 Noisy-or 推理计算 A_9 可能被观测为故障的初始概率 $P(A_9)=1-(1-P(R_8))*(1-P(R_9))=0.41$，由于 $status(A_9)=0$，因此 $P(A_9|R_8)=P(A_9|R_9)=0.02$。接下来通过标准贝叶斯推理计算 R_8 和 R_9 的后验概率。$P(R_8|A_9)=P(A_9|R_8)*P(R_8)/P(A_9)=0.02$，因此，$V_c$ 中的 $P(R_8)$由 0.41 更新为 0.02，更新 $P(R_9)$后变为0.00。以此类推，更新 V_c。

按照表 1 的过程更新 V_d，可以得到 $V_d=(0.00,0.66,0.01,0.02,0.53,0.00,0.00,0.02,0.00,0.02,0.00,0.66)$，由函数 $max_R(V_c,V_d)$计算最后的可疑度排序 $V=(0.58,0.66,0.58,0.98,0.98,0.00,0.04,0.00,0.04,0.00,0.66)$。显而易见，$P(R_4)=P(R_5)=0.98$ 是最大值，由此可知故障发生在控制依赖关系(s_4，s_5)上。此外，排在第二位的 $P(R_2)=P(R_{12})=0.66$，可以看出在数据依赖关系(s_2，s_{12})上发生了故障。

5 实验分析

采用普遍使用的西门子程序以及较大的 space 和 grep 程序(http://sir.unl.edu/content/sir.html)作为实验对象。程序的执行信息、控制依赖关系、数据依赖关系通过插桩工具 WET(http://wet.cs.ucr.edu/index.html)来获得。在奔腾 4 (2.80GHz)、内存 1.0GB 的电脑上运行红帽 9.0 系统，使用 Linux C++开发平台实现 FCG 方法。

5.1 实验设计

表 2 中“#1”代表程序单故障版本，“#2”和“#3”对应的列分别给出了对“#1”中单故障版本随机组合得到的包含 2 个故障和 3 个故障的版本数。“#测试用例”对应的列给出了每个程序的测试用例个数。

表 2 实验用到的具体数据

程序名	#1	#2	#3	#行号	#测试用例
replace	32	31	28	507	5542
schedule	9	10	9	397	2650
schedule2	10	9	8	299	2710
tcas	41	30	31	174	1608
print_tokens	7	5	5	539	4130
print_tokens2	10	10	10	489	4115
tot_info	23	15	16	398	1052
space	38	3	3	9564	4333
grep	12	4	4	12689	470

5.2 实验度量方法

采用不需要检查的代码占所有代码的百分比($Score$)来衡量算法的精确性。对于多故障程序，$Score$ 包括在“one-fault-at-one-time”模式下找到一个故障语句时的 $Score^{one}$ 以及在“many-fault-at-one-time”模式下找到全部故障语句时的 $Score^{all}$。

5.3 实验结果分析

分别在 1-故障、2-故障和 3-故障情况下，使用 LOUPE，Ochiai，Tarantula，BARINEL，FCG 算法计算所有程序的故障版本。考虑到测试用例对实验结果的影响，本文使用 3 种情况的测试用例，分别随机抽取给定的测试用例集合的 10%，50%以及 100%来进行实验。

(1)182 个包含单故障版本的实验结果

本次实验使用了表 2 中列出的 182 个单故障版本。图 7

中，横坐标表示诊断精度 *Score*，纵坐标表示达到 *Score* 的故障版本占所有故障版本的百分比。其中 *Score*=100 表示找到故障语句时需要检查的语句数小于 1%，*Score*=90 表示检查语句的百分比在 90%～99%之间，其他分数以此类推。例如，当 *Score*=80 时，在使用 10%的测试用例的情况下，FCG 有 75.2%(136 个)的版本可以在检查少于 20%的代码时找到故障。如图 7 所示，FCG 在任何测试用例下都保持最高的精确度，特别是在使用 10%的测试用例时，其效果比其他方法更好。本次实验证明了 FCG 在单故障情况下对测试用例的敏感性是最低的。

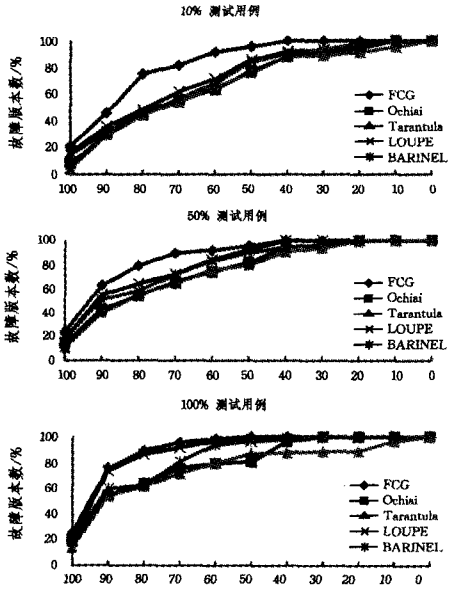

图 7　单故障的实验结果

(2)117 个包含两个故障的版本的实验结果

本次实验使用了表 2 中 117 个包含两个故障的版本，图 8(a)3 个图对比了找到一个故障语句时的 $Score^{one}$，图 8(b)是找到全部故障语句时的 $Score^{all}$。由图 8 可以看出，各种方法的 $Score^{one}$ 的差别不是很大，FCG 仅以很小的优势胜出，但是对于 $Score^{all}$，FCG 的优势明显，这说明 FCG 处理多故障的能力比其他方法更好。

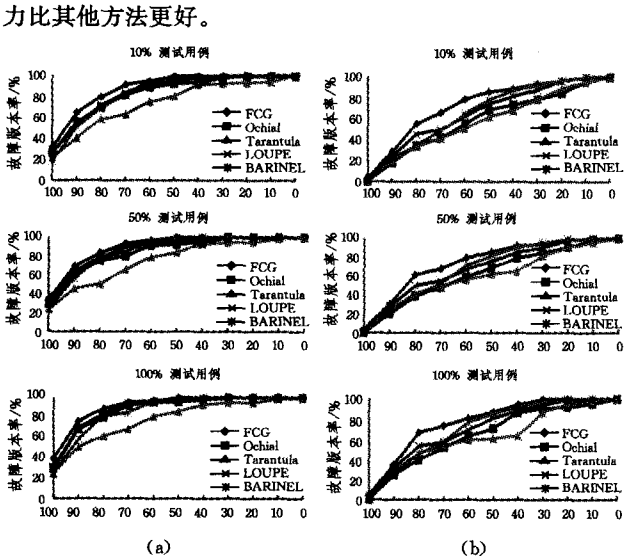

图 8　两个故障的实验结果

(3)118 个包含 3 个故障的程序版本的实验结果

图 9(a)3 个图是找到一个故障的结果，图 9(b)3 个图是找到全部故障的结果，它显示的具体结论与图 8 的结论类似。通过图 7－图 9 可以看出，随着故障个数的增多，找到多故障中的一个故障变得越来越容易，这是由程序具有故障密度决定的。但是随着故障数量的增加，找到所有故障的效率下降，这是由多故障程序的复杂的不确定性决定的。尽管这些不确定性始终难以被完全克服，但是从以上这些实验结果可以看出，考虑了故障关联的 FCG 比 Tarantula，LOUPE，Ochiai 以及 BARINEL 对于处理多故障程序不确定性具有优势。

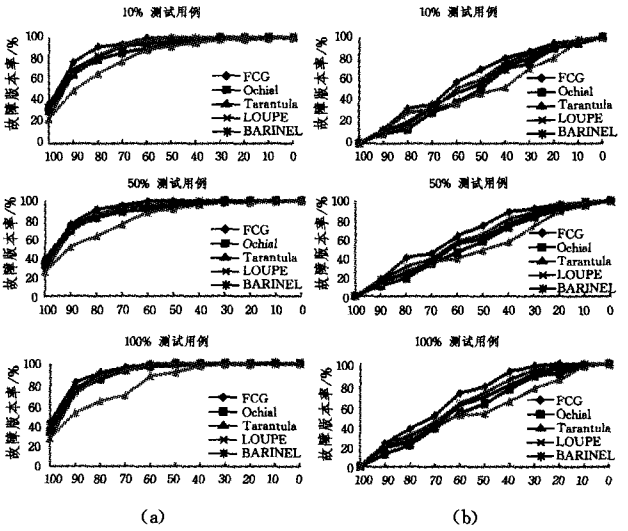

图 9　3 个故障的实验结果

6　相关工作

到目前为止，软件故障诊断研究主要有 TBFL、程序切片、基于模型的软件调试以及概率图模型诊断等方法。TBFL 技术在本质上是一种统计智能方法，根据测试覆盖信息(语句、谓词、信息流等)计算程序语句发生错误的可疑度进而进行故障诊断，例如 Ochiai，Tarantula，LOUPE 等。其中 Loupe[8] 是一种统计控制依赖关系和数据依赖关系的 TBFL 技术。由于 TBFL 方法对测试用例的执行路径敏感，Jones 等人[1] 指出类似 Tarantula 的方法不能够识别初始化语句和主要的程序路径代码。Masri[2] 证明了执行错误语句的成功测试用例普遍存在，并且严重影响 TBFL 技术的精确性。

程序切片经历了从静态切片技术到动态切片技术，再到条件切片技术的方展过程。由于采用程序切片技术能够减小问题语句的搜索空间，一些研究人员也在其它故障定位方法中综合使用程序切片技术，特别是针对面向对象的软件诊断。Wotawa 等人[12] 提出的 JSDiagnosis 工具通过计算动态切片的最小碰集，可以从切片中提取导致错误的变量，并依此得到语句的可疑值。文万志等人[13] 提出了一种基于层次切片的错误定位方法，该方法分别在包层、类层和方法层将测试的包、类和方法删出，以缩小错误存在的范围。

始于 20 世纪 70 年代中期，基于模型的软件调试技术通过建立描述程序行为的模型，对软件故障进行形式化的诊断

推理，完成行为预测、故障定位等任务。针对程序的多故障，Abreu 等人[6]提出了一种基于观测值的模型，其通过计算候选诊断的后验概率来进行软件诊断，候选诊断的计算使用了一种启发式的最小碰集算法。此外，针对单故障软件诊断，已有的基于模型的诊断技术包括：通过插桩源程序和学习执行测试用例时产生的数据信息提出概率程序依赖图[3]；根据程序指令建立的基于叶斯网络的错误流图[4]。这两种方法都没有强调处理多故障的能力。

结束语 本文提出了一种基于 TBFL 技术和无向网络的概率推理技术。针对多故障程序，在建立 FCG 时使用 PDG 来发现故障关联，并使用 TBFL 技术来计算 FCG 的概率分布，最后通过 FCG 上的深层概率推理计算得到语句的可疑度排序。通过采用不同大小的测试用例进行实验对比分析，验证了 FCG 方法对多故障程序的有效性。本文采用二进制方式来表示语句执行信息，但是带循环语句的程序可能存在执行次数大于 1 的语句，所以在将来的工作中将考虑采用整数代替二进制来表示语句的执行次数。

参 考 文 献

[1] JONES J A, HARROLD M J. Empirical evaluation of the tarantula automatic fault-localization technique[C]// Proceedings of the ACM International Conference on Automated Software Engineering. New York, USA, 2005: 273-282.

[2] MASRI W, ABOU-ASS R, EL-GHALI M, et al. An empirical study of the factors that reduce the effectiveness of coverage-based fault localization [C]// Proceedings of the 2nd International Workshop on Defects in Large Software Systems. Chicago, Illinois, USA, 2009: 1-5.

[3] BAAH G K, PODGURSKI A, HARROLD M J. The Probabilistic Program Dependence Graph and Its Application to Fault Diagnosis [J]. IEEE Transactions on Software Engineering, 2010, 36: 528-545.

[4] FENG M, GUPTA R. Learning universal probabilistic models for fault localization [C]// Proceedings of ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. Toronto, Ontario, Canada, 2010: 81-88.

[5] ABREU R, ZOETEWEIJ P, GOLSTEIJN R, et al. A practical evaluation of spectrum-based fault localization [J]. Journal of Systems and Software, 2009, 82(11): 1780-1792.

[6] ABREU R, ZOETEWEIJ P, VAN Gemund A J C. A New Bayesian Approach to Multiple Intermittent Fault Diagnosis[C]// Proceedings of International Joint Conference on Artifical Intelligence. San Francisco, 2009: 653-658.

[7] DIGIUSEPPE N, JONES J A. On the influence of multiple faults on coverage-based fault localization[C]// Proceedings of the International Symposium on Software Testing and Analysis. ACM, 2011: 210-220.

[8] YU K, LIN M, GAO Q, et al. Locating faults using multiple spectra-specific models[C]// Proceedings of ACM Symposium on Applied Computing. 2011: 1404-1410.

[9] PEARL J. Probabilistic reasoning in intelligent systems: networks of plausible inference [M]. Morgan Kaufmann Publishers, 1988.

[10] KOLLER D, Friedman N. Probabilistic Graphical Models: Principles and Techniques [M]. The MIT Press, 2009.

[11] BAO X A, XIE X M, ZHANG N, et al. Optimized Software Testing Strategy Based on the Defect Correlation Markov Model [J]. Journal of Software, 2015, 26(1): 14-25. (in Chinese)
包晓安，谢晓鸣，张娜，等. 基于缺陷关联度的 Markov 模型软件优化测试策略[J]. 软件学报，2015，26(1)：14-25.

[12] WOTAWA F. Fault Localization Based on Dynamic Slicing and Hitting-Set Computation[C]// The 10th International Conference on Quality Software, 2010. 2010: 161-170.

[13] WEN W Z, LI B X, SUN X B, et al. Technique of Software Fault Localization Based on Hierarchical Slicing Spectrum [J]. Journal of Software, 2013, 24(5): 977-992. (in Chinese)
文万志，李必信，孙小兵，等. 一种基于层次切片谱的软件错误定位技术[J]. 软件学报，2013，24(5)：977-992.

(上接第 99 页)

雷少梅，贾旭杰，于在洋. 基于高斯核函数的短时交通流量预测[J]. 中央民族大学学报(自然科学版)，2013，33(3)：82-87.

[10] XIE Y, ZHAO K, SUN Y, et al. Gaussian Processes for Short-Term Traffic Volume Forecasting[J]. Transportation Research Record: Journal of the Transportation Research Board, 2010, 2165(-1): 69-78.

[11] YU B, WU S H, WANG M H, et al. K nearest neighbor short-term traffic flow forecasting model[J]. Journal of Traffic and Transportation Engineering, 2012, 12(2): 105-111. (in Chinese)
于滨，邬珊华，王明华，等. K-临近短时交通流预测模型[J]. 交通运输工程学报，2012，12(2)：105-111.

[12] WILLIAMS B M, HOEL L A. Modeling and Forecasting Vehicular Traffic Flow as a Seasonal ARIMA Process: Theoretical Basis and Empirical Results[J]. Journal of Transportation Engineering, 2014, 129(6): 664-672.

[13] SUN S, YU G, ZHANG C. Short-term traffic flow forecasting using Sampling Markov Chain method with incomplete data[C]// Intelligent Vehicles Symposium, 2004 IEEE. IEEE, 2004: 437-441.

[14] ZHANG S, ZHAO Z F, DING W L, et al. Urban Road Trip Time Measured Calculation Based on Map- Reduce[J]. Computer and Digital Engineering, 2014, 42(9): 1542-1546. (in Chinese)
张帅，赵卓峰，丁维龙，等. 基于 MapReduce 的城市道路旅行时间实测计算[J]. 计算机与数字工程，2014，42(9)：1542-1546.

[15] 邓聚龙. 灰预测与灰决策(修订版)[M]. 武汉：华中科技大学出版社，2002.