基于贝叶斯与多故障推理的Web服务诊断

贾志淳 邢 星
(渤海大学信息科学与技术学院 锦州 121013)

摘 要 Web服务组合进程规模的不断增加和复杂性的不断提高都有可能导致服务的运行失败,因此很难确保服务的可靠性。针对传统的基于模型的方法无法处理服务应用系统中状态和行为故障的不确定性问题,提出了一种基于贝叶斯与多故障逻辑推理的诊断方法。该方法使用历史数据构建服务执行矩阵,通过多故障逻辑推理技术获取候选诊断解并利用贝叶斯公式计算候选解的后验概率并排序,最终获得一个最优的诊断结果。相比于传统基于模型的Web服务诊断方法,该方法不仅可以同时定位多个故障,而且能够随着历史数据的增加不断优化诊断结果。实验证明该方法具有很好的诊断效果。

关键词 Web服务,故障诊断,逻辑推理,贝叶斯规则
中图法分类号 TP391.5 **文献标识码** A

Diagnosis of Web Service Based on Bayes and Multi-faults Reasoning

JIA Zhi-chun XING Xing
(School of Information Science and Technology, Bohai University, Jinzhou 121013, China)

Abstract The increasing size and complexity of web service composition process has led to an amplified number of potential failures and makes it harder to ensure service reliability. To localize the faults of undefined service behaviors in service process, we proposed a diagnosis method based on Bayes and multi-faults logic reasoning. On the basis of modeling the service execution matrix by historical data, we combined multi-faults logic reasoning technique with the statistical technique to obtain the diagnosis candidates. By applying Bayes' formula to compute the fault probability of each diagnosis candidate, our diagnostic method is able to obtain an asymptotically optimal diagnosis with increasing historical data. Experiments were conducted to evaluate the effectiveness of the method in diagnosing the web services with multi-faults.

Keywords Web service, Fault diagnosis, Logic reasoning, Bayes rule

1 引言

服务中组件行为数目非常大时,传统的基于模型诊断方法产生的候选诊断的数目会呈指数增长,导致方法的诊断效率及诊断质量低下,难以广泛应用于实际中[1]。为了解决基于模型的Web服务故障诊断方法在这方面的不足,研究人员提出了很多方法来减少诊断的服务组件数目,进而提高诊断效率。一些研究人员提出了分层诊断的方法来解决基于模型诊断的计算复杂性问题[2],然而这种方法也会受服务组件规模的影响,当组件数目增加时诊断的粒度也会相应变大,诊断的精确性和诊断时间都会受到影响。还有一些研究人员提出通过构建服务故障模型[3,4]来提高诊断效率,这种方法在模型中描述了服务部分已知的异常运行状态,可以有效定位这些已知的故障。然而,Web服务运行在一个动态、多变的网络环境当中,在运行过程中服务需要动态发现并绑定外部服务,使得设计者们在设计与部署阶段无法预见所有可能的变化[5],于是故障具有很强的不可预知性,因此很难构建一个完整的故障模型,模型的不完备性限制了此种方法的诊断能力。

虽然已经有一些研究者对服务故障诊断问题进行了细致、深入的研究[6-13],但是这些研究多采用传统的基于模型的诊断,这些方法很难通过形式化的方式来描述系统的不确定状态和行为,对于故障诊断过程中的输入、服务执行以及输出之间存在的不确定性关系不能进行处理[5]。现有针对具有非确定性服务系统的诊断一般采用基于历史数据的概率分析方法,如Dai等人[14]提出了一个使用历史故障数据构建模糊异常矩阵的基于故障传播度的诊断方法,该方法利用概率描述每个行为与异常行为的关系,通过计算获得每个行为发生故障的可疑度,继而定位服务故障;Han等人[15]利用已有日志文件通过计算日志间的结构相似性来构建一个贝叶斯学习网络,然后通过计算待诊断日志与贝叶斯网络的相似度来分类日志,确定日志文件的故障类型。然而这类方法同样有它自身的缺点,即无法深入动态系统的本质解释故障发生的根本原因并对系统进行准实时诊断。

本文提出一种基于贝叶斯与多故障逻辑推理的诊断方

到稿日期:2013-09-01 返修日期:2013-11-12
贾志淳(1982—),女,博士,副教授,主要研究方向为Web服务、服务诊断、云计算等,E-mail:zhichun.jia@gmail.com;邢 星(1982—),男,博士,副教授,主要研究方向为社交网络挖掘、社会计算、推荐系统等。

法，利用历史数据对系统的不确定状态和行为进行建模，将概率描述信息引入到基于模型的诊断，通过概率分析与多故障推理计算得出诊断结果。该方法与传统的基于模型的诊断方法相比计算复杂度降低，并且利用贝叶斯更新规则该方法能够随着历史数据的增加不断更新诊断结果，达到准实时诊断的目的。

2 服务执行矩阵

服务执行矩阵为服务诊断提供了一个特殊的分析服务动态行为的方式，它不断从服务的执行中收集历史数据，更新诊断的信息。服务执行矩阵是由一组数字和不同行为的标记组成的，矩阵中的每一行表示一条执行轨迹，而矩阵中的前 m 列表示服务中的 m 个行为，每个行为都通过一个标记来表示该行为是否存在于执行轨迹当中，也就是在执行轨迹当中该行为是否被执行；第 $m+1$ 列表示每条执行轨迹执行失败的次数，第 $m+2$ 列表示每条执行轨迹执行成功的次数，可参见图 1。

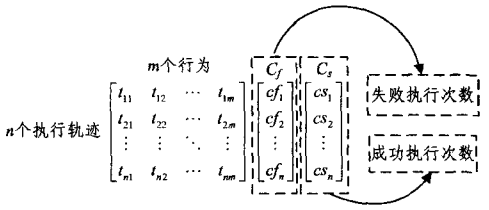

图 1 服务执行矩阵

定义 1 一个服务执行矩阵

$SEM_{n(m+2)}=[Track_{nm}, C_f, C_s]$

其中，n 表示历史数据中执行轨迹的个数；m 表示历史数据中行为的个数；$Track_{nm}=\{t_{ij}\mid 1\leqslant i\leqslant n, 1\leqslant j\leqslant m\}$，$t_{ij}$ 表示行为 j 在第 i 条执行轨迹中是否被执行，如果行为 j 在第 i 条执行轨迹中被执行，那么 $t_{ij}=1$，否则 $t_{ij}=0$；$C_f=\{cf_i\mid 1\leqslant i\leqslant n\}$，$cf_i$ 表示第 i 条执行轨迹失败执行的次数；$C_s=\{cs_i\mid 1\leqslant i\leqslant n\}$，$cs_i$ 表示第 i 条执行轨迹成功执行的次数。

算法 1 给出了构建基本服务执行矩阵的方法，这里假设历史数据是已知的，并且数据格式规定如下：

• 历史数据 $HD=\{hd(i)\mid 1\leqslant i\leqslant l\}$，$l$ 表示 HD 中的执行路径个数；

• $hd(i)=\{r, b(j)\mid 1\leqslant j\leqslant v\}$ 表示第 i 条执行路径，r 表示第 i 条执行路径的执行结果，$r=f$ 表示第 i 条执行路径执行失败，$r=s$ 表示第 i 条执行路径执行成功，$b(j)$ 表示第 i 条执行路径中的第 j 个行为的标号，v 表示 i 条执行路径执行行为的个数。

算法 1 *getSEM(hd)*

```
输入:历史数据 hd
输出:服务执行矩阵 SEM
1. m=hd 中行为个数;
2.   SEM=zero[1,m+2];
3. for i=1:hd. length
4.     track=zero[1,m+2];
5.     for j=1:hd(i). length
6.       track(1,hd(i),b(j))=1;
7.     end for
8.     for k=1 : SEM. length
9.       if track[1,m]=SEM[k,m];
10.        if hd(i). r=f,SEM(k,m+1)++;
11.        else SEM(k,m+2)++;
12.        break;
13.      end if
14.    end for
15.    if k=SEM. length+1;
16.      if hd(i). r=f,track(1,m+1)=1;
17.      else track(1,m+2)=1;
18.      SEM=SEM∪track;
19.    end if
20. end for
21. return SEM.
```

在算法 1 中，$zero[1,m+2]$ 表示构建一个单行 $m+2$ 列且值全为零的矩阵；$SEM[k,m]$ 表示 SEM 矩阵中第 k 行的前 m 个元素。

3 多故障诊断

基于服务执行矩阵的贝叶斯诊断方法利用多故障诊断推理技术，提出一种求解最小故障行为集的方法，将求得的多个最小故障行为集作为诊断候选，引入贝叶斯公式计算每一个诊断候选的故障概率，最终选取概率最大的诊断候选作为故障集。

3.1 相关定义

1987 年，Reiter 等人[16]提出了基于第一原理的多故障诊断方法，在其诊断理论中给出了一种形式化诊断方法，是一种使用深层知识的形式化诊断推理。该方法利用系统的结构和行为等信息，根据实际观测的系统行为与系统描述中应该具有的正确行为之间的不一致，来判断系统存在的故障行为。如果实际观测与系统描述相冲突，就说明系统遭遇了一个诊断问题，也就是说，一个或多个系统组件发生了异常；这些与系统描述相冲突的观测就是一个冲突集合。

定义 2 对于一个 Web 服务诊断系统 $DS=(WS, SB, Obs)$，一个冲突集 $CS=\{b_1, b_2, \cdots, b_k\}$，其中：$WS$ 是一个服务的描述；SB 是服务的行为集合，并且 $\{b_1, b_2, \cdots, b_k\}\subseteq SB$；$Obs$ 是服务执行的观察集合；$WS\cup Obs\cup\{\neg ab(b_1), \neg ab(b_2), \cdots, \neg ab(b_k)\}$ 是不一致的，也就是说 $\{b_1, b_2, \cdots, b_k\}$ 必定包含故障行为才导致观测与系统描述冲突，$ab(b_i)$ 表示行为 b_i 是一个故障行为。

如果一个 Web 服务中包含冲突，那么需要知道冲突集中的哪些行为能够解释实际观测与系统描述之间的这个冲突，而从冲突集中找出的最小故障行为集就是一组能够解释这个冲突的行为集合，也称为诊断候选。

定义 3 对于一个 Web 服务诊断系统 $DS=(WS, SB, Obs)$ 存在一个冲突集 CS，CS 的一个最小故障行为集是一个行为集合 $mfbs$，并且：

• $\forall c_i\in CS, c_i\cap mfbs\neq\emptyset$，$c_i$ 是 CS 中的一个冲突序列；

• 当且仅当 $mfbs\subseteq mfbs'$，则有 $\forall c_i\in CS, mfbs'\cap c_i\neq\emptyset$。

从定义 3 可以看出，如果 $mfbs$ 是冲突集 CS 的一个最小故障行为集，那么 $mfbs$ 与 CS 中的任何冲突序列都至少存在一个相同的行为，且这些相同的行为能够解释冲突序列发生的冲突。

需要强调的是，对于一个冲突集来说，冲突集中并不一定

只包含一组最小故障行为集，可能有多个最小冲突集都能用于解释该冲突的发生。

3.2 求解最小故障行为集

为了获得能够解释实际观察与服务描述不一致的最小故障行为集，本文提出了一种求解最小故障行为集的方法，该方法改进了 Abreu[17] 提出的用于求解程序故障诊断中的最小碰集的方法，使其适用于求解故障服务的最小故障行为集。

首先定义一个相似性系数 $behSim$[17]，该相似性系数用于表示行为与异常观测之间的相似程度。

$$behSim(i)=\frac{n_{f1}(i)}{\sqrt{(n_{f1}(i)+n_{s1}(i))(n_{f1}(i)+n_{f0}(i))}} \quad (1)$$

其中，

$$n_{f1}(i)=\sum_{j=1}^{n}(SEM(j,i)\cdot SEM(j,m+1))$$

$$n_{s1}(i)=\sum_{j=1}^{n}(SEM(j,i)\cdot SEM(j,m+2))$$

$$n_{f0}(i)=\sum_{j=1}^{n}(SEM(j,m+1)\wedge(SEM(j,i)=0))$$

这里，$n_{f1}(i)$表示行为 i 在失败执行中出现的次数，$n_{s1}(i)$表示行为 i 在成功执行中出现的次数，$n_{f0}(i)$表示行为 i 没有出现在失败执行中的次数。

然后根据相似性系数排序所有行为，并从系数最大(也就是与异常观测最相关)的行为开始查找，看该行为是否能覆盖所有失败执行，如果能则认为其是一个最小故障行为集，并将其从行为集合和失败执行中删掉；当检查完所有行为之后，如果行为集合不为空，并且不满足停止条件，那么就迭代寻找能够覆盖所有失败执行的行为组合并验证该组合是否满足最小故障行为集定义，如果该组合是一个最小故障行为集就将该组合放入解集当中，直到行为集合为空或满足停止条件。具体的求解最小行为故障集的方法见算法 2。

算法 2 *get_mfbs(SEM,l,s)*

输入：服务执行矩阵 SEM，最小故障行为集个数 l，搜索停止标准 s

输出：最小故障行为集集合 Mset

```
1. m=SEM(1).length-2;
2. cf=Σ_{i=1}^{n} SEM(i,m+1);
3. rb=rank(behSim,SEM,m);
4. Mset=∅;sp=0;
5. for i=1:m
6.   if n_f1(i)=cf
7.     Mset=Mset∪{i};sp=sp+1/m;
8.     dm=delete(SEM,i);rb=delete(rb,i);
9.   end if
10. end for
11. while rb≠∅∧Mset.length≤l∧sp≤s
12.   b=rb(1);rb=delete(rb,1);
13.   dm'=delete(dm,j);sp=sp+1/m;
14.   Mset'=get_mfbs(dm',l,s);
15.   while Mset'≠∅
16.     diag=Mset'(1);
17.     Mset'=delete(Mset',1);
18.     diag=diag∪{b};
19.     if subset(Mset,diag)=false
20.       Mset=Mset∪diag;
21.     end if
22.   end while
23. end while
24. return Mset.
```

在算法 2 中，m 表示 SEM 中的行为个数；cf 表示 SEM 中失败执行的次数总和；$rank(behSim,SEM,m)$表示根据式(1)计算 SEM 中 m 个行为的相似性系数并从大到小排序行为，返回一个行为序列；$delete(SEM,i)$表示一个新的服务行为矩阵，该矩阵不包含 SEM 中的第 i 列；$delete(rb,i)$表示将 rb 中的第 i 个元素删除，并返回删除第 i 个元素后的数组；$delete(dm,j)$表示将 dm 中的第 j 列元素删除，并返回删除第 j 列元素后的矩阵；$delete(Mset',1)$表示将 $Mset'$ 中的第 1 个子集删除，并返回删除第 1 个子集后的集合；$subset(Mset,diag)$用于检查 $diag$ 是否包含 $Mset$ 中任何一个子集，如果 $diag$包含 $Mset$ 中任何一个子集，那么返回 true，否则返回 false。该函数用来验证获得的集合是否是一个最小故障行为集。

3.3 贝叶斯诊断

在获得最小故障行为集集合 $Mset=\{mfbs_i\mid 1\leqslant i\leqslant w\}$ 后，还需要计算这些最小故障行为集可能是故障的概率，这里 w 表示集合中最小故障行为集的个数。首先假设行为发生故障的可能都是独立于其它行为的，然后应用贝叶斯概率计算公式计算 $Mset$ 中每个最小故障行为集对于 SEM 中的每个执行序列的后验故障概率，计算公式如下：

$$fP(mfbs_i\mid SEM(j))=\frac{fP(SEM(j)\mid mfbs_i)}{fP(SEM(j))}fP(mfbs_i) \quad (2)$$

其中，

$$fP(SEM(j)\mid mfbs_i)=succ(mfbs_i)^{cs_j}(1-succ(mfbs_i))^{cf_j}$$

这里，$succ(mfbs_i)$表示在所有最小故障行为集 $mfbs_i$ 中的行为的执行中成功执行的概率，即

$$succ(mfbs_i)=\frac{n_{s1}(mfbs_i)}{n_{s1}(mfbs_i)+n_{f1}(mfbs_i)}$$

并且

$$n_{s1}(mfbs_i)=\sum_{j=1}^{n}(cs_j\wedge(\exists t_{jk}\in mfbs_i\wedge t_{jk}=1))$$

$$n_{f1}(mfbs_i)=\sum_{j=1}^{n}(cf_j\wedge(\exists t_{jk}\in mfbs_i\wedge t_{jk}=1))$$

对于式(2)中的 $mfbs_i$ 的先验故障概率 $fP(mfbs_i)$，本文设定为 0.01，因此如果 $mfbs_i$ 中包含 k 个行为，那么 $mfbs_i$ 的先验概率就是

$$fP(mfbs_i)=(0.01)^k(1-0.01)^{m-k}$$

当每次计算得出一个在给定执行序列下的 $mfbs_i$ 的后验概率时，都将其作为下一次计算的先验概率，因此当所有执行序列都被考虑后，$mfbs_i$ 的最终后验概率也就是

$$fP(mfbs_i\mid SEM)=\prod_{j=1}^{n}\frac{fP(SEM(j)\mid mfbs_i)}{fP(SEM(j))}\cdot fP(mfbs_i) \quad (3)$$

这里，对于 SEM 中的所有执行序列，它们的概率都是相同的，因此可以使用一个变量 α 来代替所有的 $fP(SEM(j))$，那么式(3)可以简化为

$$fP(mfbs_i\mid SEM)=\prod_{j=1}^{n}fP(SEM(j)\mid mfbs_i)\frac{fP(mfbs_i)}{\alpha^n}$$

又因为所有最小故障行为集的故障概率之和为 1，即

$$\sum_{i=1}^{w}fP(mfbs_i\mid SEM)=1$$

所以

$$\alpha=\sqrt[n]{\sum_{i=1}^{w}\left(\prod_{j=1}^{n} fP(SEM(j)\mid mfbs_i)\, fP(mfbs_i)\right)}$$

至此已经求得 *Mset* 中的所有最小故障行为集的故障概率，通过对这些概率进行排序，就能够得到一个具有多个故障行为的诊断解。算法 3 描述了整个服务故障诊断过程。

算法 3 *semDiag*(*HD*)

输入：历史数据 HD

输出：诊断解 d

```
1. SEM=convert(HD);
2. Mset=get_mfbs(SEM,1,s);
3. mid=0;mv=0;
4. for i=1:Mset. length
5.   p(i)=fP(Mset(i),SEM);
6.   if mv<p(i)
7.     mv=p(i);
8.     mid=i;
9.   end if
10. end for
11. d=Mset(mid);
12. return d.
```

在算法 3 中，*convert*(*HD*)表示将历史数据 *HD* 转换为诊断服务所需要的服务执行矩阵；$p(i)=fP(Mset(i),SEM)$ 表示将使用式(3)计算得到的 *Mset* 中的第 i 个最小故障行为集的故障概率赋予 $p(i)$；*mv* 和 *mid* 用于记录最大概率和拥有最大故障概率的最小故障行为集编号。

4 案例研究

本节以图 2 为例说明本文方法的整个诊断过程。如图 2 所示，当一个用户想要在网上订购机票时，用户通过订票服务端口向订票代理服务发送订票请求，订票代理服务首先根据接收到的用户信息调用用户等级查询服务查找用户等级，如普通用户、VIP 用户等。然后订票代理服务再根据用户的等级及用户输入的机票信息(如时间、出发地、目的地等)从 American Airlines 和 Air China 两个航空公司查询相应的机票价格。最后，订票代理服务比较价格并返回最便宜的机票信息给用户。

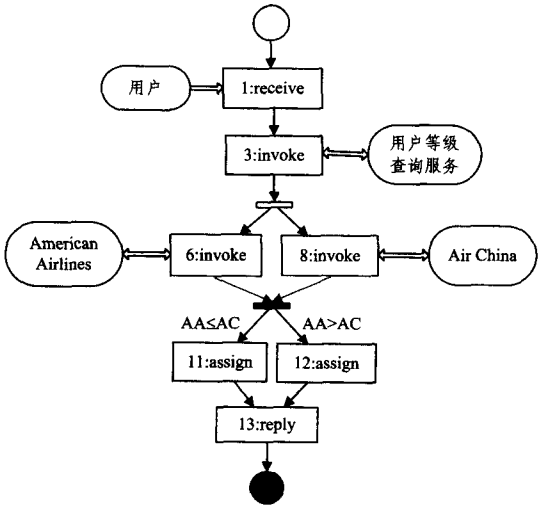

图 2 订票代理服务实例

假定行为 b_1 和 b_{11} 发生了故障，且这两个行为失败执行的概率都是 0.85，然后依据给定的故障概率生成 30 条执行序列，并将其转化为服务执行矩阵，如表 1 所列。

表 1 订票代理服务的服务执行矩阵

	b_1	b_3	b_6	b_8	b_{11}	b_{12}	b_{13}	cf	cs
1	0	1	1	1	0	1	1	0	5
2	1	0	0	0	0	0	1	10	0
3	1	0	0	0	1	0	1	5	0
4	0	1	0	0	1	0	0	10	0

根据得到的服务执行矩阵，通过算法 1 一共可以得到 3 个最小故障行为集，*Mset*={{1,11},{3,13},{11,13}}。然后根据式(3)分别计算这 3 个最小故障行为集的故障概率，最后得到如下结果：

$fP(\{1,11\}\mid SEM)=0.999974$

$fP(\{3,13\}\mid SEM)=0.000013$

$fP(\{11,13\}\mid SEM)=0.000013$

其中，$\alpha=0.00094$。

根据计算结果可以看出，最小故障行为集{1,11}是诊断解。这个案例说明本文的诊断方法对于故障服务的诊断是有效的。

5 仿真实验

5.1 实验设置

为了评估本文方法的有效性，我们使用 Matlab 实现了一个用于服务故障诊断的仿真实验系统，该系统从 QWS (Quality of Web Service) 数据集[18,19]中收集了 2507 个真实的 Web 服务属性集合，并且给这些服务属性赋予 2507 个行为结点，这些行为结点被分配到不同的服务池中。通过定义一组服务端口集合，并将集合中的服务端口随机分配给每个服务池一个输入端口和一个输出端口，使同一服务池中的所有服务结点都具有相同的输入和输出端口。为了生成所需要的 Web 服务工作流，我们首先给定要生成的 Web 服务的行为个数和其中包含的结构行为个数。根据给定的个数我们使用随机选择方法确定结构行为在工作流中的位置，从选择、并发、循环中随机选择一个作为结构行为的结构类型。接着，我们从服务池中随机选择一个服务结点作为要生成的 Web 服务的初始行为结点，然后检查下一个结点是否是结构行为结点，如果不是则根据当前结点的输出端口从服务池中找到具有与该端口类型一致的输出端口的服务池，再从该服务池中随机选择一个服务结点作为该 Web 服务的下一个行为结点；如果是则将相应结构行为结点插入，并根据结构类型和内部基本行为结点个数从服务池中选择服务结点；而结构结点自身则按照普通结点一样分配结点属性；直至结点数达到给定的服务结点数，整个 Web 服务生成完成。最后，根据已得到的工作流及其行为的属性，我们生成指定个数的工作流执行路径。如果我们并没有特别指出结点失败执行概率，那么按照初始服务属性中的可靠性概率生成结点，否则按照给定失败概率生成执行路径。通过得到的工作流及其执行信息，我们可以使用设定的诊断方法对服务故障进行诊断。

为了评估本文方法的诊断准确性，共产生两组实验数据，每组包含 100 个 Web 服务，第一组中的每个 Web 服务都包含 10 个行为结点，而第二组中的每个 Web 服务都包含 20 个行为结点。对于这两组数据中的每一个 Web 服务，分别随机

选择1个、2个和5个行为作为故障行为，设置这些故障行为的失败执行概率为0.8，并且根据这3组故障行为每个Web服务生成10组分别包含10，20，…，100条执行路径的历史数据。

5.2 实验分析

在图3中使用第一组实验数据比较了不同数量的历史数据信息对包含不同故障行为个数的服务的诊断准确率的影响。图中的 $fn=1$ 表示服务中的故障行为个数是1，$fn=2$ 表示服务中的故障行为个数是2，而 $fn=5$ 表示服务中的故障行为个数是5。可以看出，本文诊断方法的诊断准确率在62%到83%之间，即使在服务中包含多个故障行为时，也能准确地诊断出故障发生位置；本文方法的诊断准确率会随着历史数据中执行路径个数的增加而不断提高，直到到达一个稳定状态。这主要是由于历史数据中执行路径个数越多就意味着在数据中有助于本文方法做出诊断的信息越多，历史数据所覆盖的诊断信息也就越完备。而当历史数据中的执行路径个数达到50条时，诊断的准确率并未随着执行路径个数的增加而增加，这是由于历史数据已经覆盖了较完备的诊断信息，而此时增加的执行路径都是之前信息的重复，因此并不对诊断结果产生影响。

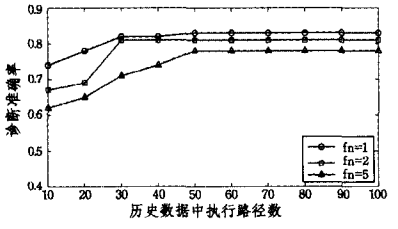

图3 第一组实验数据的诊断准确率比较

图4使用第二组实验数据比较了不同数量的历史数据信息对包含不同故障行为个数的服务的诊断准确率的影响。从图4中可以看出与图3一样的变化规律，即本文方法的诊断准确率会随着历史数据的增加而不断提高，当历史数据到达一定数量后，诊断准确率也会趋于稳定。然而，与第一组实验数据不同的是，第二组实验数据需要更多的历史数据才能使准确率趋于稳定，也就是说，当历史数据中执行路径个数为80时，准确率才保持不变。这主要是因为第二组实验数据中每个Web服务中的行为个数要比第一组实验数据的行为个数多，因此有更多不同的执行路径，也就需要更多的历史数据才能覆盖所有的执行情况。

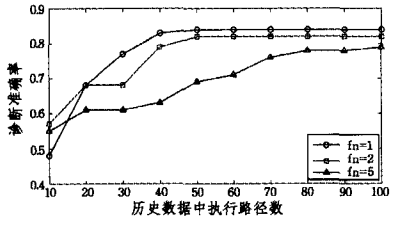

图4 第二组实验数据的诊断准确率比较

从实验中可以看出，基于服务执行矩阵的贝叶斯诊断方法对于包含不同故障行为个数的Web服务都能做出有效的诊断，并且该方法会随着诊断信息的增加而不断优化诊断结果。

结束语 本文提出了一种基于贝叶斯和多故障推理的Web服务诊断方法。该方法利用历史数据作为诊断信息，首先将历史数据转化为服务执行矩阵，应用诊断推理方法分析服务执行矩阵中的行为执行信息，找出可能引起服务故障的最小故障行为集；根据贝叶斯公式计算每个最小故障行为集的故障概率并排序，最后确定诊断解。该方法不仅能够诊断出服务系统中的故障行为，而且随着历史数据的不断增加，该方法的诊断结果也会不断优化。仿真实验的结果显示该方法是有效的，而且诊断准确性随着历史数据的增加而不断提高。

参 考 文 献

[1] Yan Y, Dague P, Pencole Y, et al. A model-based approach for diagnosing faults in web service processes [J]. The International Journal of Web Services Research (JWSR), 2009, 6(1): 87-110

[2] Console L, Picardi C, Dupre D T. A framework for decentralized qualitative model-based diagnosis [C]// Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07). San Francisco, CA, USA, 2007: 286-291

[3] Li Y, Ye L, Dague P, et al. A decentralized model-based diagnosis for BPEL services [C]// Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence (ICTAI'09). Newark, NJ, 2009: 609-616

[4] 范贵生，虞慧群，陈丽琼，等. 基于 petri 网的服务组合故障诊断与处理 [J]. 软件学报，2010，21(2)：231-247

[5] 付晓东. Web服务组合服务质量保障关键问题研究 [D]. 昆明：昆明理工大学，2008

[6] Zhang Jing, Huang Zhen-qiu, Lin K J. A hybrid diagnosis approach for QoS management in service-oriented architecture [C]// 2012 IEEE 19th International Conference on Web Ser-vices (ICWS). Honolulu, HI, 2012: 82-89

[7] Jayashree K, Anand S. Policy based distributed run time fault diagnoser model for web services [C]// 2nd International Conference on Computer Science and Information Technology (CCSIT 2012). Bangalore, India, 2012: 9-16

[8] Farj K, Yuhui C, Speirs N A. A fault injection method for testing dependable web service systems [C]// 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC). Guangdong, China, 2012: 47-55

[9] Fan Gui-sheng, Yu Hui-qun, Chen Li-qiong, et al. A petri net-based Byzantine fault diagnosis method for service composition [C]// 2012 IEEE 36th Annual Computer Software and Applications Conference (COMPSAC). Izmir, 2012: 42-51

[10] Zhu Z, Dou W. QoS-based probabilistic fault-diagnosis method for exception handling [C] // New Horizons in Web-Based Learning: ICWL 2010 Workshops. Berlin, 2011: 227-236

[11] Mayer W, Friedrich G, Stumptner M. Diagnosis of service failures by trace analysis with partial knowledge [J]. Service-Oriented Computing, 2010, 6470: 334-349

[12] Kopp O, Leymann F, Wutke D. Fault handling in the web service stack [J]. Service-Oriented Computing, 2010, 6470: 303-317

[13] Ardissono L, Bocconi S, Console L, et al. Enhancing web service composition by means of diagnosis [C]// Business Process Management Workshops. Milano, Italy, 2008: 468-479

[14] Dai Y, Yang L, Zhang B, et al. Exception diagnosis for composite service based on error propagation degree [C] // 2011 IEEE

International Conference on Services Computing (SCC 2011). Washington, DC, USA, 2011: 160-167

[15] Han X, Shi Z, Niu W, et al. Similarity-based Bayesian learning from semi-structured log files for fault diagnosis of web services [C]// 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT). Toronto, Canada, 2010: 589-596

[16] Reiter R. A theory of diagnosis from first principles [J]. Artificial Intelligence, 1987, 32(1): 57-95

[17] Abreu R, Gemund A J C V. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis [C]// Symposium on Abstraction, Reformulation and Approximation. Lake Arrowhead, CA, USA, 2009: 2-8

[18] Al-Masri E, Mahmoud Q H. Discovering the best web service [C] // 16th International Conference on World Wide Web (WWW). Banff, Alberta, Canada, 2007: 1257-1258

[19] Al-Masri E, Mahmoud Q H. QoS-based discovery and ranking of web services [C]// IEEE 16th International Conference on Computer Communications and Networks (ICCCN). Maui, Hawaii, 2007: 529-534

(上接第 187 页)

迭代过程如表 1 所列，初始时 S 中只有 a。

表 1 时序最短距离迭代过程

迭代	S	u	dist(a,b)	dist(a,c)	dist(a,d)	dist(a,e)	dist(a,f)	dist(a,g)	dist(a,h)	dist(a,i)
初始	{a}	—	5	15	22	∞	∞	∞	∞	∞
1	{a,b}	b	5	15	22	8	∞	∞	∞	∞
2	{a,b,e}	e	5	15	22	8	∞	∞	∞	∞
3	{a,b,e,c}	c	5	15	22	8	20	∞	∞	∞
4	{a,b,e,c,f}	f	5	15	22	8	20	∞	∞	35
5	{a,b,e,c,f,d}	d	5	15	22	8	20	25	∞	35
6	{a,b,e,c,f,d,g}	g	5	15	22	8	20	25	∞	35
7	{a,b,e,c,f,d,g,i}	i	5	15	22	8	20	25	∞	35

4.2 算法结果对比

从表 1 得知，对给定的时序网络图 3，用本文提出的算法得到节点 a 到其他节点的最短时序距离分别为：$dist[b]=5$，$dist[c]=15$，$dist[d]=22$，$dist[e]=8$，$dist[f]=20$，$dist[g]=25$，$dist[h]=\infty$，$dist[i]=35$。文献[13]的算法(即式(1))在将时间窗口 T 取为 35 时，得到节点 a 到其他节点的平均时序距离分别为：$\tau_{ab}=14.66$，$\tau_{ac}=16.52$，$\tau_{ad}=16.52$，$\tau_{ae}=16.523$，$\tau_{af}=\infty$，$\tau_{ag}=\infty$，$\tau_{ah}=\infty$，$\tau_{ai}=\infty$。

通过对比可知，利用式(1)所得的节点间平均时序距离和本文算法得到的最短时序距离是不同的，式(1)得到的是整个时间窗口内的平均值，并且可能会出现本来可达的节点之间平均时序距离为无穷大的情况，比如节点 a 到 f，而本文中提出的算法得到的是精确的结果。

结束语 本文提出的算法是针对更接近现实生活的时序网络的，可用于解决信息从源到目的地所需要的最短时间、疾病从起源地到其他地方需要的最短时间、挑选最快到达一个地点的航班等问题。文章在第 3 节提出最短时序路径算法，第 4 节进行严格证明，并在第 5 节进行实证分析。由此表明，本文提出的算法具有可行性。但是本算法是一种精确算法，计算复杂度和 Dijkstra 算法是一样的，在大规模复杂网络中，寻找节点间的时序最短路径可能会造成时间过长的问题。

参 考 文 献

[1] 荣玮. 基于道路网的最短路径算法与实现[D]. 武汉：武汉理工大学，2005

[2] 涂海丽. 最短路径算法及其应用探讨[J]. 科技广场，2011(9)

[3] 吴莲. 基于城市道路网的遗传最短路径算法研究[D]. 南京：南京理工大学，2010

[4] 于东凯，刘玉树. 基于平面图的最短路径算法的研究[J]. 北京理工大学学报，2003，21(1)

[5] 董俊，黄传河. 改进 Dijkstra 算法在 GIS 导航应用中最短路径搜索研究[J]. 计算机科学，2012，39(10)

[6] 艾菊梅，周书民，陆玲. 基于 WebGIS 公交查询平台与移动增值服务[J]. 微计算机信息，2007(10)

[7] SCHULTESD. Route planning in road networks[D]. Karlsruhe: UniversitätFridericiana, 2008

[8] Fu L. Real-time vehicle routing and scheduling in dynamic and stochastic traffic networks[D]. Edmonton, Alberta: University of Alberta, 1996

[9] Nicosia G, Oriolo G. An approximate A* algorithm and its application to the SCS problem[J]. Theoretical Computer Science, 2003, 290(3): 2021-2029

[10] Bander J L, White C C. A heuristic search algorithm for path determination with learning[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 1998, 28(1): 131-134

[11] 宋青，汪小帆. 最短路径算法加速技术研究综述[J]. 电子科技大学学报，2012(2)

[12] 唐晋韬，王挺，王戟. 适合复杂网络分析的最短路径近似算法[J]. 中国：软件学报，2011，22(10)

[13] Rajk P, Jari S. Path lengths, correlations, and centrality in temporal networks[J]. Finland: Phys. Rev. E 84, 2011: 016105

[14] Dijkstra E W. A note on two problems in connexion with graphs [J]. Numerical Mathematics, 1959, 1(1): 269-271

[15] 严蔚敏，吴伟民. 数据结构[M]. 北京：清华大学出版社，2001，186-190

[16] 张先迪，李正良. 图论及其应用[M]. 北京：高等教育出版社，2005

[17] Petter H, Jari S. Temporal networks[J]. Phys. Rep, 2012, 519: 97-125