基于证据推理面向生命周期的可信软件评估

傅仰耿[1],巩晓婷[2],吴英杰[1]

(1. 福州大学数学与计算机科学学院,福建 福州 350108)
(2. 福州大学公共管理学院,福建 福州 350108)

[摘要] 可信软件评估是目前学术界研究的一个新的热点和难点,为解决可信软件评估过程中未能有效处理区间不确定评价信息的集结、可信准则体系动态生成算法效率不够高的问题,本文提出了一种基于证据推理面向软件生命周期的可信软件评估方法,给出了可信准则体系的动态生成的两个改进算法,并通过使用区间置信度的证据推理方法进行可信准则评价信息的集结. 最后通过实验和算例验证了该方法的优越性与有效性.

[关键词] 可信软件评估,区间置信度,证据推理,多准则决策分析
[中图分类号]TP311 [文献标志码]A [文章编号]1001-4616(2013)01-0133-09

An Evidential Reasoning Approach for Trustworthy Software Evaluation Under Systems Development Life Cycle

Fu Yanggeng[1], Gong Xiaoting[2], Wu Yingjie[1]

(1. College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China)
(2. School of Public Administration, Fuzhou University, Fuzhou 350108, China)

Abstract: The trustworthy software assessment is a new hot and hard issue in recent research. In order to solve the problems, such as, aggregating information from different criteria under interval uncertainty and constructing dynamical evaluation model for trustworthy criteria more efficiently, we propose an evidential reasoning approach for trustworthy software evaluation under systems development life cycle by giving two improved algorithms for constructing dynamical evaluation model of trustworthy criteria and aggregating evaluation information in the form of interval belief degrees via evidential reasoning algorithm for trustworthy criteria. Finally, superiority and validity of the approach is verified through experiments and a numerical example.

Key words: trustworthy software evaluation, evidential reasoning, interval belief degrees, MCDA

近年来,随着信息科学与技术的迅猛发展,信息产业达到了空前的繁荣,并迅速渗透到社会生活的各个领域,已成为社会生产力发展和人类文明进步的强大动力. 软件是信息产业的核心部分,随着信息化程度的不断深入,人们对软件功能的需求逐步加大,使得软件系统越来越庞大,变得难以控制,系统缺陷和漏洞层出不穷,国内外由于软件缺陷、故障和失效而导致的重大灾难、事故时有发生,各式各类的软件系统总是不以人们期望的方式工作,直接或间接地给用户带来严重损失[1]. 因此,如何评估软件的可信性,确保其在生存期缺陷尽可能少并保障其安全可靠运行,具有很现实的应用前景.

目前,学术界倾向于认为,如果一个软件的行为总是与预期一致,则该软件是可信的. 对于软件可信性的评估,前人已经取得了一系列有意义的研究结果,同时也存在着一定的不足,如:文献[2]提出了一种基于 Trusted Platform Module 的软件可信证据收集机制,并在基于 Linux Security Module 实现了软件原型,该方法具有良好的可扩展性,但无法直接应用于其他不同类型的软件,即通用性存在不足;文献[3]提出了一种基于效用和证据理论的可信软件评估方法,通过基于开放的可信准则数据库和可信准则树生成算法构造评估准则,可以较为灵活地实现评估准则树的动态构建,但是其核心算法是两个递归算法,在评估准则较多的情况下需要付出较大的时间和空间代价,有待进一步优化;文献[4]研究了互联网上开源软件可

收稿日期:2013-01-05.
基金项目:国家杰出青年科学基金(70925004)、福建省教育厅科技项目(JA10035).
通讯联系人:傅仰耿,博士研究生,讲师,研究方向:不确定多准则决策分析、数据安全与隐私保护. E-mail:fu@fzu.edu.cn

信证据的自动化获取方法,并提出了一个面向互联网的开源软件自动化评估证据框架,但仅适用于各类文档齐全的开源软件;文献[5]通过在软件开发阶段植入检查点的方法对软件可信性进行风险监测和风险评估,然而关于检查点选择还有待进一步深入研究;文献[6]提出了一种基于需求动态演化的软件可信性评估方法,该方法能够根据用户需求自主配置评估准则体系,但需要事先知道软件的运行状态.

综上所述,软件可信性评估是一个重要而且复杂的科学问题,目前的解决方案在一定程度上解决了软件可信性度量问题.然而由于开放、动态、多变的互联网环境给软件的开发和运行带来了很多不确定的因素,单一的评价体系已无法满足软件的可信性需求,取而代之的是行为可信、环境可信和使用可信等不同层次的可信要求,如何从整体上度量、获得并保证可信性仍然是非常困难的问题[1].要解决这个问题就亟需一种能够处理不确定、不精确、不完整、模糊信息的分析评价模型.鉴于此,本文拟采用基于证据推理的不确定多准则决策方法[7](以下简称证据推理方法)来融合可信软件评估中多个方面的不确定评价信息,从软件生命周期的角度来建模、度量软件的可信性.

1 证据推理方法

将证据理论[8]用于多准则决策是由 Yang 和 Singh 在 1994 年首先提出来的,他们提出了一种基于证据理论的证据推理方法,可用于解决准则值不完整的多准则决策问题[7].为进一步建模复杂决策问题的不确定信息,Yang 在 2001 年又提出了一种用于建模不完整和不精确信息的通用决策模型[9],进一步完善了证据推理方法.然而该模型在某些情况仍有可能导致不合理的结果,Yang 和 Xu 在 2002 年对该模型进行了进一步的完善,他们深入研究了证据推理方法的基础特性,并提出了准则权重规范化和基本概率分配(Basic Probability Assignments,BPA)的新方法[10],进一步发展原有的证据推理方法,这些最终成为证据推理方法的理论基础.

在此基础上,文献[11]得到了一个证据推理的解析算法,显著减少了准则集结过程中的合成计算量.为处理准则评价值的置信度不确定性和准则评价值为区间数的情形,文献[12]扩展了证据推理方法,使用证据推理方法的解析公式同时合成所有的证据,构建了两对非线性优化模型,分别用于估计合成置信度上下限和计算方案期望效用值的最大和最小值,同时提出了区间数到区间置信度的等价转换方法.

总之,证据推理方法是证据理论与决策理论相结合的产物,随着区间计算、模糊数学、效用理论和概率论等相关理论的引入,该方法已达到空前的完善和发展.其具备一些明显的优点,如:能处理具有定量和定性准则的混合决策问题;能建模信息的不完整、不精确和无知;可用来建立规则库和作预测;能较好地处理具有模糊性及不确定性信息的合成问题等.因此,可处理可信软件评估中不确定、不精确、不完整、模糊的评价信息.

2 面向生命周期的可信软件评估

要提高软件的构造质量,必须从软件的需求开始植入相应的评价与反馈机制,这是近年来软件质量研究所取得的共识.同样,要保证软件的可信性,必须对软件的整个生命周期进行相应的评价与监控,以下将从软件生命周期的角度,介绍软件可信性评估的准则体系构建及评价信息的集结方法.

2.1 评估准则体系的构建

软件开发的过程一般包括:需求收集与分析、设计、编码、测试和部署运行等不同的阶段.为保证软件的质量,在文献[4,13]的基础上,本文按照软件生命周期分阶段归纳了可信评价的基本准则,如表 1 所示,作为收集相应的可信证据的基本框架,可在此基础上进一步构建更加符合可信需求的评价模型,以对不同需求的可信软件进行评价.

表 1 软件生命周期的可信准则

Table 1 Trustworthy criteria for systems development life cycle

软件生命周期	可信准则
需求收集与分析	需求描述规范程度、需求分析人员的能力等级、需求变更频率、需求评审结论、需求阶段的评审缺陷密度和缺陷清除率等
设计	设计人员的能力等级、设计阶段的需求变更数、设计评审结论、设计阶段的评审缺陷密度和缺陷清除率
编码	编码人员的能力等级、编码阶段的需求变更数、单元测试强度、代码行数、代码注释率、McCabe 圈复杂度、非条件跳转语句数、扇入和扇出、最大嵌套层数、代码可维护性
测试	测试人员的能力等级、测试工具支持的有效性、测试缺陷趋势、每千行代码的错误数和缺陷数、文档页数
部署运行	部署过程不合格项趋势

以上对软件生命周期每个不同的阶段均可建立相应的评价准则,这些准则有些是定性的(如:人员的能力等级),有些是定量的(如:非条件跳转语句数),不管是定量还是定性的准则,都要获取相应的事实或者评价信息. 对于评估准则的构建,可事先建立准则数据库,然后再根据不同的可信需求进行筛选更新,并动态生成评估准则体系. 文献[3]给出了一种准则体系的动态构建方法,其核心算法描述如下:

算法 1 原始递归算法

```
NIT FNode=TSE;
Insert(FNode){
  if(FNode.isBaseNode)break;
  for(i=0;i<INDB.length;i++){
    if(INDB[i].fatherID==FNode.ID){
      FNode.child[j]=INDB[i].ID;
      Insert(INDB[i]);
      j++;
    }
  }
}
```

该算法通过比对结点的 ID 递归搜索当前结点的子结点,对于每个结点均要遍历 INDB 数组,故时间复杂度为 $O(N*N)$,而空间复杂度为 $0(N)$. 由此可知,在准则数量较多的情况下,效率将大打折扣. 为改进上述算法的性能,本文提出了以下准则体系构建的改进递归算法,描述如下:

算法 2 改进递归算法

```
NIT FNode=TSE;
int Insert(FNode,index){
  if(FNode.isBaseNode)return 0;
  int reduce=0;
  for(i=0;i<INDB.length;i++){
    Node=INDB[i];
    if(Node.fatherID==FNode.ID){
      FNode.child[j]=Node.ID;
      if(i<=index)reduce+=1;
      INDB.remove(Node);
      int s=Insert(Node,i--);
      reduce+=s;
      i-=s;
      j++;
      newINDB[k++]=Node;
    }
    }
  return reduce;
}
```

在原始递归算法中,每次递归都需要遍历 INDB 数组,但由建树的过程可知,INDB 数组的每个元素都只会被访问一次,故在改进的算法中针对该点,将每次访问后的元素从 INDB 中移除,进而降低了时间的复杂度,尽管最坏情况下仍为 $O(N*N)$,但其实际性能已有较大的提高,这一点在后面的实验中将得到验证. 改进版的递归算法中,由于结点元素逐个从 INDB 数组中移除,故需要另外开辟一个等长度的数组 newINDB 进行储存元素,但空间复杂度维持在 $0(N)$. 为进一步提高该算法的性能,本文又提出了以下非递归的改进算法,描述如下:

算法 3 非递归算法

```
NIT FNode=TSE;
Insert(FNode){
```

```
    INDB[INDB.length]=FNode;
    FNode.fatherID=INDB.length+1;
    for(i=0;i<INDB.length;i++){
      if(! map.containsKey(INDB[i].ID))map.put(INDB[i].ID,i);
    }
    for(i=0;i<INDB.length;i++){
      INDB[i].ID=map.get(INDB[i].ID);
      if(map.containsKey(node.fatherID))INDB[i].fatherID=map.get(INDB[i].fatherID);
    }
    for(i=0;i<INDB.length;i++){
      if(INDB[i].fatherID==INDB.length+1)continue;
      INDB[INDB[i].fatherID].child[j]=INDB[i].ID;
      j++;
    }
  }
```

将结点的 ID 与数组的下标进行匹配,从而方便利用结点的 ID 直接取得所需的结点,且每个结点在建树过程中只会访问一次,所以在建树中时间复杂度为 $O(N)$. 但对于每个结点的编号未必从 0 开始且连续,故需利用哈希将所有结点的 ID 映射成从 0 开始且连续的新的结点 ID,同时更新结点中 fatherID,该处哈希的过程中时间复杂度为 $O(N\log N)$,所以总的时间复杂度为 $O(N\log N+N)$,而在空间复杂度上,算法中均维持在原有空间大小的基础上,故为 $0(N)$. 由此可见,改进的非递归算法将在性能上取得较大的提升.

由上可知,评价准则体系所获取的评价信息可能是定量的也可能是定性的,如何对于不同的评价信息进行集结是一个核心问题,文献[9]给出了一种基于规则和效用的转换方法,可用于定量与定性评价信息的转换. 此外,评价信息还可是区间或是模糊的,以区间为例,以下简要介绍使用区间置信度的证据推理方法用于集结不同准则评估信息的具体实现.

2.2　评估信息的集结方法

在前人工作的基础上,本文将一种使用区间置信度的证据推理方法用于处理准则等级置信度为区间的可信软件评估问题,集结区间不确定的多个准则的评价信息.

首先,根据软件生命周期来建立可信软件的评估准则体系,由于该体系是动态变化的,故以下仅给出大致的框架,如图 1 所示.

其次,在构建面向软件生命周期的可信评估准则体系之后,根据不同的可信需求可确立相应的评价模型,并获取不同阶段的可信证据. 最后即可进行准则评估信息的集结. 以下给出的具体算法步骤,该算法是针对两层结构的评价模型,对于多层结构的评价模型,则可采用自底向上两两合并的方式.

假设有一个两层结构的评价模型,准则 y 下辖 M 个基本准则,且每个准则的评价等级相同,如图 2 所示.

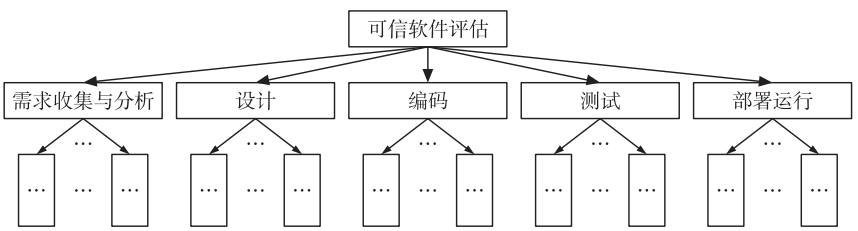

图 1　可信软件的评估准则体系

Fig. 1　Evaluation criteria system for trustworthy software

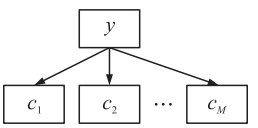

图 2　两层结构的评价模型

Fig. 2　A two-level hierarchy for evaluation

则可信软件评估的证据推理方法可描述如下[10,14]:

(1)定义与描述问题

假设对 L 个待评估的软件 $a_l, l=1,\cdots,L$,定义了 M 个可信准则 $c_i, i=1,\cdots,M$,及 N 个不同的等级 H_n, $n=1,\cdots,N$,则有软件集合 $A=\{a_1,\cdots,a_L\}$,准则集合 $C=\{c_1,\cdots,c_M\}$,等级集合 $H=\{H_1,\cdots,H_N\}$. 那么,可信软件评估问题可以使用一个置信结构决策矩阵来表示:

$$\boldsymbol{S}(c_i(a_l))=\{(H_n,\beta_{n,i}(a_l)),n=1,\cdots,N\},\quad i=1,\cdots,M;\quad l=1,\cdots,L. \tag{1}$$

其中 $\beta_{n,i}(a_l)$ 表示在软件 a_l 中准则 c_i 被评估为等级 H_n 的置信度,满足:

$$\beta_{n,i}(a_l)\geqslant 0 \text{ 且 } \sum_{n=1}^{N}\beta_{n,i}(a_l)\leqslant 1. \tag{2}$$

每个准则需要规范化的权重,若用 w_i 表示准则 c_i 的权重,则有 $W=\{w_1,\cdots,w_M\}$,满足:

$$0\leqslant w_i\leqslant 1 \text{ 和 } \sum_{i=1}^{M} w_i=1. \tag{3}$$

(2)确定每个准则基本概率分配(BPA)

令 $m_{n,i}$ 为基本可信度,表示软件 a_l 中准则 y 的基本准则 c_i 被评估为等级 H_n 的支持程度,满足:

$$m_{n,i}=w_i\beta_{n,i}(a_l), \tag{4}$$

令 $m_{H,i}$ 表示在准则 $c_i(i=1,\cdots,M)$ 未分配给任意等级的剩余可信度,满足:

$$m_{H,i}=1-\sum_{n=1}^{N} m_{n,i}=1-w_i\sum_{n=1}^{N}\beta_{n,i}(a_l),\quad (i=1,\cdots,M). \tag{5}$$

将 $m_{H,i}$ 拆分为两个部分,$\bar{m}_{H,i}$ 和 $\tilde{m}_{H,i}$ 分别表示由权重引起的不确定和由评价信息不完整引起的不确定,满足:

$$\bar{m}_{H,i}=1-w_i, \tag{6}$$

$$\tilde{m}_{H,i}=w_i\left(1-\sum_{n=1}^{L}\beta_{n,i}\right), \tag{7}$$

$$m_{H,i}=\bar{m}_{H,i}+\tilde{m}_{H,i}. \tag{8}$$

(3)迭代合成每个软件的整体概率分配

令 $m_{n,I(1)}=m_{n,1}(n=1,\cdots,N)$,$m_{H,I(1)}=m_{H,1}$,$\bar{m}_{H,I(1)}=\bar{m}_{H,1}$,$\tilde{m}_{H,I(1)}=\tilde{m}_{H,1}$ 则:

$\{H_n\}$:

$$m_{n,I(i+1)}=K_{I(i+1)}\left[m_{n,I(i)}m_{n,i+1}+m_{H,I(i)}m_{n,i+1}+m_{n,I(i)}m_{H,i+1}\right],\quad n=1,2,\cdots,N. \tag{9}$$

$\{H\}$:

$$m_{H,I(i)}=\tilde{m}_{H,I(i)}+\bar{m}_{H,I(i)}, \tag{10}$$

$$\tilde{m}_{H,I(i+1)}=K_{I(i+1)}\left[\tilde{m}_{H,I(i)}\tilde{m}_{H,i+1}+\bar{m}_{H,I(i)}\tilde{m}_{H,i+1}+\tilde{m}_{H,I(i)}\bar{m}_{H,i+1}\right], \tag{11}$$

$$\bar{m}_{H,I(i+1)}=K_{I(i+1)}\left[\bar{m}_{H,I(i)}\bar{m}_{H,i+1}\right], \tag{12}$$

$$K_{I(i+1)}=\left[1-\sum_{t=1}^{N}\sum_{\substack{j=1\\ j\neq t}}^{N} m_{t,I(i)}m_{j,i+1}\right]^{-1}, \tag{13}$$

其中 $i=\{1,2,\cdots,M-1\}$,$m_{n,I(i)}$ 表示前 i 个准则合成的基本可信度. 以上经过 $M-1$ 次的迭代即可求得每个软件关于准则 y 的可信度.

(4)计算每个软件合成的整体信任度

令 β_n 表示在软件 a_l 中准则 y 被评估为 H_n 的置信度,那么:

$\{H\}$:

$$\beta_H=\frac{\tilde{m}_{H,I(M)}}{1-\bar{m}_{H,I(M)}}, \tag{14}$$

$\{H_n\}$:

$$\beta_n=\frac{m_{n,I(M)}}{1-\bar{m}_{H,I(M)}},\quad n=1,2,\cdots,N. \tag{15}$$

软件 a_l 的全部评估信息可以表示成:

$$\boldsymbol{S}(y(a_l))=\{(H_n,\beta_n(a_l)),n=1,\cdots,N\}. \tag{16}$$

(5)计算每个软件的期望效用值和效用区间

令 $u(H_n)$ 表示等级 H_n 的效用,可记为:$u_1,\cdots,u_N$,假设 $u(H_1)\leqslant u(H_2)\leqslant\cdots\leqslant u(H_N)$,即:$u_1\leqslant u_2\leqslant\cdots\leqslant u_N$. 如果全部评估信息是完整的,那么软件 a_l 对准则 y 的期望效用值可以通过式子 $u(a_l)=\sum_{n=1}^{N}\beta_n(a_l)u(H_n)$ 计算;如果评估信息并非完整,那么最大、最小、平均效用值可以通过以下式子计算:

$$u_{\max}(a_l)=\sum_{n=1}^{N-1}\beta_n(a_l)u(H_n)+(\beta_N(a_l)+\beta_H(a_l))u(H_N),\tag{17}$$

$$u_{\min}(a_l)=(\beta_1(a_l)+\beta_H(a_l))u(H_1)+\sum_{n=2}^{N}\beta_n(a_l)u(H_n),\tag{18}$$

$$u_{\text{avg}}(a_l)=\frac{u_{\max}(a_l)+u_{\min}(a_l)}{2}.\tag{19}$$

如果全部评估信息是完整的,那么 $\beta_H(a_l)=0$,$u(\boldsymbol{S}(y(a_l)))=u_{\max}(a_l)=u_{\min}(a_l)=u_{\text{avg}}(a_l)$.

上述算法假设所有准则的评价等级一致,但现实往往并非如此,有的准则是定量准则,有的则是定性准则,即使同是定量或者定性,其评价等级也不尽相同. 对于准则评价等级不一致的情况,则需要配置准则转化规则,可以通过基于规则的信息转化技术[9]来解决.

针对上述算法迭代计算量大、不便于对模型参数进行优化设计、以及输入输出关系为"黑箱"等的局限性,文献[11]推导和证明了证据推理方法的解析公式,解决了该方法输入输出关系上的"黑箱"问题. 该解析算法主要改进了迭代合成过程,能够将 M 个准则一次合成,其计算公式如下:

$$k=\Big[\sum_{n=1}^{N}\prod_{i=1}^{M}(m_{n,i}+\bar{m}_{H,i}+\tilde{m}_{H,i})-(N-1)\prod_{i=1}^{M}(\bar{m}_{H,i}+\tilde{m}_{H,i})\Big]^{-1},\tag{20}$$

$$\bar{m}_H=k\Big[\sum_{i=1}^{M}\bar{m}_{H,i}\Big],\tag{21}$$

$$\tilde{m}_H=k\Big[\prod_{i=1}^{M}(\bar{m}_{H,i}+\tilde{m}_{H,i})-\prod_{i=1}^{M}\bar{m}_{H,i}\Big],\tag{22}$$

$$m_n=k\Big[\prod_{i=1}^{M}(m_{n,i}+\bar{m}_{H,i}+\tilde{m}_{H,i})-\prod_{i=1}^{M}(\bar{m}_{H,i}+\tilde{m}_{H,i})\Big],\tag{23}$$

$$\beta_H(a_l)=\frac{\tilde{m}_H}{1-\bar{m}_H},\quad n=1,2,\cdots,N;\quad l=1,2,\cdots,L,\tag{24}$$

$$\beta_n(a_l)=\frac{m_n}{1-\bar{m}_H},\quad n=1,2,\cdots,N;\quad l=1,2,\cdots,L.\tag{25}$$

为了处理准则评价值的置信度为区间不确定的情形,在上述可信软件评估的证据推理方法的基础上,引入文献[12]的扩展方法:

(1)将评价值的置信度扩展为区间,即 $\beta_{n,i}\in[\beta_{n,i}^-,\beta_{n,i}^+]$,准则评价值变为区间置信结构:

$$\boldsymbol{S}(c_i(a_l))=\{(H_n,[\beta_{n,i}^-(a_l),\beta_{n,i}^+(a_l)]),\quad n=1,\cdots,N\}.\tag{26}$$

若是有效评价则应满足:

$$\sum_{n=1}^{N}\beta_{n,i}(a_l)\leqslant 1,\beta_{n,i}(a_l)\in[\beta_{n,i}^-(a_l),\beta_{n,i}^+(a_l)].\tag{27}$$

其中信任度 $\beta_{H,i}(a_l)$ 代表的是所有等级集合 H 的置信度,也就是由评价信息不完整引起的未知部分,表示为 $[\beta_{H,i}^-,\beta_{H,i}^+]$,其计算公式如下:

$$\beta_{H,i}^-(a_l)=\max\Big(0,1-\sum_{n=1}^{N}\beta_{n,i}^+(a_l)\Big),\tag{28}$$

$$\beta_{H,i}^+(a_l)=1-\sum_{n=1}^{N}\beta_{n,i}^-(a_l).\tag{29}$$

(2)在用区间表示置信度的基础上,*BPA* 定义如下:

$$m_{n,i}\in[m_{n,i}^-,m_{n,i}^+]=[w_i\beta_{n,i}^-(a_l),w_i\beta_{n,i}^+(a_l)],\tag{30}$$

$$\bar{m}_{H,i}=1-w_i,\tag{31}$$

$$\tilde{m}_{n,i}\in[\tilde{m}_{H,i}^-,\tilde{m}_{H,i}^+]=[w_i\beta_{H,i}^-(a_l),w_i\beta_{H,i}^+(a_l)],\tag{32}$$

满足以下约束:

$$m_{n,i}^-\leqslant m_{n,i}\leqslant m_{n,i}^+,\tag{33}$$

$$\tilde{m}_{H,i}^-\leqslant\tilde{m}_{H,i}\leqslant\tilde{m}_{H,i}^+,\tag{34}$$

$$\sum_{n=1}^{N}m_{n,i}+\bar{m}_{H,i}+\tilde{m}_{H,i}=1,\quad i=1,\cdots,M;n=1,\cdots,N.\tag{35}$$

(3)使用式(20)~(23)合成所有的可信准则评价信息,并构建以下非线性优化模型,用于估计合成置信度上下限:

$$\begin{aligned}&\text{Max/Min}\quad \beta_n(a_l)=\frac{m_n}{1-\bar{m}_H},\quad \beta_H(a_l)=\frac{\tilde{m}_n}{1-\bar{m}_H},\\&\text{s.t.}\quad \text{式}(20)\sim(23),\\&\qquad \text{式}(33)\sim(35).\end{aligned}\tag{36}$$

(4)构建以下最优化模型求解每个软件的最大与最小效用值:

$$\begin{aligned}&\text{Max}\quad u_{\max}(a_l)=\sum_{n=1}^{N-1}u(H_n)\beta_n(a_l)+u(H_N)(\beta_N(a_l)+\beta_H(a_l)),\\&\text{s.t.}\quad \text{式}(20)\sim(23),\\&\qquad \text{式}(33)\sim(35).\end{aligned}\tag{37}$$

$$\begin{aligned}&\text{Min}\quad u_{\min}(a_l)=u(H_1)(\beta_1(a_l)+\beta_H(a_l))+\sum_{n=2}^{N}u(H_n)\beta_n(a_l)),\\&\text{s.t.}\quad \text{式}(20)\sim(23),\\&\qquad \text{式}(33)\sim(35).\end{aligned}\tag{38}$$

3 实验结果及算例

3.1 准则树构建算法实验

我们在一台 2G 内存、2.7GHz 处理器、装有 32bit Window 7 的 PC 机上对以上所实现的算法进行了测试对比. 实验数据依据给定的树高随机生成,每个结点分支数的范围为 1~6. 得到如下准则树高为 7 和 8 的两组数据,见表 2 和表 3,图 3 和图 4 是表 2 和表 3 依据准则个数变化的趋势图. 从表中可以看出,随着评价准则的增多,3 种算法的效率有明显的区别,本文所提出的两种算法明显优于原始算法,这一点在图 3 和图 4 中可以得到印证.

表 2 3 个算法的效率比较(准则树高为 7)

Table 2 The comparison of performance for three algorithms (The criteria tree height is 7)

结点数	原始递归算法	递归改进算法	非递归算法
1 095	0.011 410 426	0.006 591 474	0.002 092 710
3 305	0.112 471 055	0.059 391 213	0.001 949 969
5 846	0.324 056 191	0.184 896 120	0.003 089 448
6 672	0.436 577 106	0.255 363 393	0.004 046 590
7 806	0.628 341 871	0.392 928 880	0.024 566 488
9 182	0.873 797 523	0.477 211 656	0.005 012 531
11 596	1.282 162 887	0.746 496 141	0.022 038 224
12 378	1.705 881 634	0.819 098 165	0.006 779 676
14 253	2.285 110 084	1.178 428 810	0.024 444 279
16 174	4.496 632 160	1.393 054 511	0.009 086 985
17 437	5.347 288 642	2.510 917 434	0.010 528 075

表 3 3 个算法的效率比较(准则树高为 8)

Table 3 The comparison of performance for three algorithms (The criteria tree height is 8)

结点数	原始递归算法	递归改进算法	非递归算法
2 213	0.056 414 686	0.039 021 876	0.006 235 112
6 835	0.516 447 643	0.337 656 116	0.003 738 623
10 730	1.659 890 412	0.817 977 262	0.006 026 379
12 566	2.894 570 638	1.086 837 926	0.008 048 696
16 144	3.406 790 822	2.104 708 328	0.010 720 188
20 076	5.258 699 704	2.209 035 364	0.030 823 108
24 207	8.716 031 350	5.544 859 809	0.014 753 579
33 128	15.875 455 168	10.682 439 960	0.024 023 879
40 437	26.120 180 009	14.820 661 953	0.024 377 796
49 956	31.687 447 114	20.172 945 529	0.042 969 722
63 372	65.576 579 443	26.503 183 549	0.096 513 471

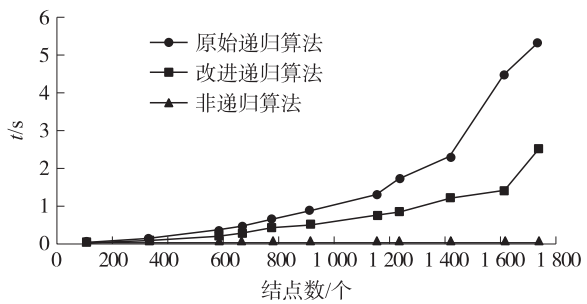

图 3 3 个算法的效率比较(准则树高为 7)

Fig. 3 The comparison of performance for three algorithms (The criteria tree height is 7)

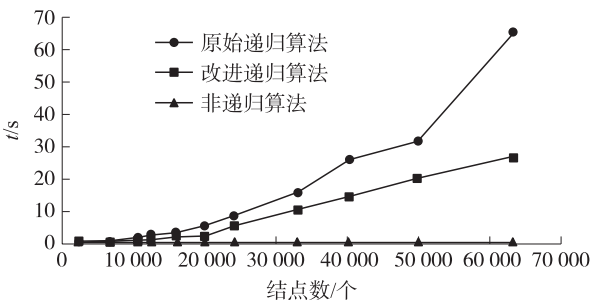

图 4 3 个算法的效率比较(准则树高为 8)

Fig. 4 The comparison of performance for three algorithms (The criteria tree height is 8)

以上实验表明,本节所提出的两种改进算法的效率均优于原始的递归算法,这与之前的理论分析结果相同.

3.2　可信准则评价信息集结算例

对软件进行可信性的评估是一个极其复杂的不确定过程,简单起见,本节仅给出如何运用基于区间置信度的证据推理方法来进行可信准则评价信息集结的一个算例. 假设软件生命周期的某个阶段有 5 个可信准则,每个准则的评价等级均为:{Poor(P),Average(A),Good(G),Very good(V),Excellent(E)},即 $H=\{H_1,H_2,H_3,H_4,H_5\}=\{P,A,G,V,E\}$,同时假设这 5 个可信准则的权重均相同,即 $W=(0.2,0.2,0.2,0.2,0.2)$,经过评估后得到如下 5 种软件(S1,S2,S3,S4,S5)在 5 个可信准则下的评价信息的区间置信度分布,如表 4 所示.

表 4　对 5 种软件进行可信性的评估得到的置信度分布

Table 4　Distributed assessment information using interval belief degrees for the five kinds of software

可信准则	S1	S2	S3	S4	S5
1	{(P,[0.2,0.4]),(A,[0.6,0.7])}	Unknown	{(G,[0.2,0.4]),(V,[0.5,0.7])}	{(A,[0.4,0.5]),(G,[0.3,0.6])}	{(G,[0.6,0.8)}
2	{(G,[0.3,0.5]),(V,[0.5,0.6])}	{(A,[0.6,0.7])}	{(V,[0.5,0.6]),(E,[0.3,0.6])}	{(V,[0.1,0.3]),(E,[0.6,0.8])}	{(G,[0.2,0.3]),(V,[0.6,0.7])}
3	{(G,[0.3,0.4]),(V,[0.6,0.7])}	{(A,[0.3,0.5]),(G,[0.5,0.6])}	{(G,[0.4,0.6])}	Unknown	{(G,[0.1,0.2]),(V,[0.8,0.9])}
4	{(V,[0.2,0.4]),(E,[0.6,0.7])}	{(G,[0.7,0.8])}	{(G,[0.2,0.5]),(V,[0.4,0.6)}	{(V,[0.4,0.5]),(E,[0.5,0.6])}	{(A,[0.5,0.8])}
5	{(G,[0.3,0.4]),(V,[0.6,0.8])}	{(V,[0.5,0.7])}	{(G,[0.3,0.5]),(V,[0.5,0.6])}	{(V,[0.1,0.3]),(E,[0.7,0.8])}	{(A,[0.3,0.4]),(G,[0.5,0.7])}

假设 $u(H_1)=0$,$u(H_2)=0.4$,$u(H_3)=0.6$,$u(H_4)=0.8$,$u(H_5)=1.0$,根据式(30) ~ (36)可求得 5 个可信准则的综合评价信息的置信度分布,再根据式(37) ~ (38)即可求得 5 种软件的效用值和排序,如表 5 所示. 最终可以得到软件的可信性评价排序为:S4,S3,S1,S5,S2.

表 5　对 5 种软件进行可信性评价的结果

Table 5　The result of trustworthy evaluation for the five kinds of software

期望效用	S1	S2	S3	S4	S5
最大值	0.741 060	0.765 258	0.824 433	0.922 834 0	0.723 592
最小值	0.594 824	0.306 398	0.544 963	0.552 335 0	0.487 326
平均值	0.667 942	0.535 828	0.684 698	0.737 584 5	0.605 459
排序	3	5	2	1	4

4　小结

本文介绍了证据推理方法及其在可信软件评价中的应用,从软件生命周期的角度,介绍了如何对可信软件评价准则进行构建,改进了前人所实现的准则构建算法,使得新的准则构建算法在复杂度上有较大的降低;引入了使用区间置信度的证据推理方法,使得在进行可信准则评价信息集结时,能够处理置信度的区间不确定性;实验与算例表明,本文所实现的可信软件评估方法有效.

尽管目前在可信软件评估上取得了一定程度的进展,但仍然有很多亟待解决的问题,比如如何确定可信软件评估准则的权重,该问题将是我们下一步要重点研究的内容.

[参考文献]

[1]　刘可,单志广,王戟,等. 可信软件基础研究重大研究计划综述[J]. 中国科学基金,2008,22(3):145-151.

[2]　古亮,郭耀,王华,等. 基于 TPM 的运行时软件可信证据收集机制[J]. 软件学报,2010,21(2):373-387.

[3] 杨善林,丁帅,褚伟. 一种基于效用和证据理论的可信软件评估方法[J]. 计算机研究与发展,2009,46(7):1 152-1 159.
[4] 袁霖,王怀民,尹刚,等. 面向互联网的开源软件自动化评估证据框架[J]. 小型微型计算机系统,2011,32(11):2 145-2 151.
[5] 刘玉玲,杜瑞忠,冯建磊,等. 基于软件行为的检查点风险评估信任模型[J]. 西安电子科技大学学报:自然科学版,2012,39(1):179-184.
[6] 丁帅,鲁付俊,杨善林,等. 一种需求驱动的软件可信性评估及演化模型[J]. 计算机研究与发展,2011,48(4):647-655.
[7] Yang J B, Singh M G. An evidential reasoning approach for multiple attribute decision making with uncertainty[J]. IEEE Transactions on Systems, Man and Cybernetics, 1994, 24(1): 1-18.
[8] Shafer G. A Mathematical Theory of Evidence[M]. Princeton: Princeton University Press, 1976.
[9] Yang J B. Rule and utility based evidential reasoning approach for multiple attribute decision analysis under uncertainty[J]. European Journal of Operational Research, 2001, 131(1): 31-61.
[10] Yang J B, Xu D L. On the evidential reasoning algorithm for multiattribute decision analysis under uncertainty[J]. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, 2002, 32(3): 289-304.
[11] Wang Y M, Yang J B, Xu D L. Environmental impact assessment using the evidential reasoning approach[J]. European Journal of Operational Research, 2006, 174(3): 1 885-1 913.
[12] Wang Y M, Yang J B, Xu D L. The evidential reasoning approach for multiple attribute decision analysis using interval belief degrees[J]. European Journal of Operational Research, 2006, 175(1): 35-66.
[13] 刘旭东,郎波,谢冰,等. 软件可信分级规范[EB/OL]. http://www.trustie.org/UserFiles/File/[TRUSTIE-STC]软件可信分级规范[V2.0]-水印.pdf, 2009.5.
[14] Yang J B, Xu D L. Nonlinear information aggregation via evidential reasoning in multiattribute decision analysis under uncertainty[J]. IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans, 2002, 32(3): 376-393.

[责任编辑:陆炳新]