基于复杂网络的软件测试路径聚类分析

周宽久,冯金金,兰文辉,迟宗正
ZHOU Kuan-jiu,FENG Jin-jin,LAN Wen-hui,CHI Zong-zheng

大连理工大学 软件学院,辽宁 大连 116620
Software School,Dalian University of Technology,Dalian,Liaoning 116620,China
E-mail:fjj.dut@163.com

ZHOU Kuan-jiu,FENG Jin-jin,LAN Wen-hui,et al.Cluster analysis of software testing paths based on complex networks.Computer Engineering and Applications,2010,46(31):72-76.

Abstract: A complex networks-based algorithm that can get fewer software testing paths is put forward to reduce the number of test data,and to improve test efficiency.Firstly the method of calculating the nodes' weight is advanced.This method is based on the weighted complex networks model of software system.Then a similarity matrix is created according to the uncertainty during software running.Finally,a matrix decomposition algorithm is issued to convert the matrix to obtain the partition of testing paths.After cluster analysis of the seven Linux files,the test paths are on average 17.46% saves.It is shown that this method is simple,rapid and effective from both theoretical analysis and experiment.
Key words: complex networks;software testing;cluster analysis;similarity matrix

摘 要:

Linux

7 17.46%
关键词:
DOI:10.3778/j.issn.1002-8331.2010.31.020 **文章编号:**1002-8331(2010)31-0072-05 **文献标识码:A** **中图分类号:**TP311

1 引言

随着软件系统越来越复杂,如何用尽量少的测试用例覆盖尽量多的测试路径成为当务之急。

一方面,传统测试方法存在着这样或那样的缺陷[1-2]。等价类测试注意到数据依赖关系和函数本身,使用这些技术时要更多考虑判断技巧。决策表技术要求测试人员既考虑数据又考虑逻辑依赖关系。而所有功能性测试方法都有一定的局限:存在未测试的漏洞和冗余测试。结构性测试的主要问题是测试时的工作量太大。另外,这些测试方法很少考虑不同的测试节点会具有不同的重要性。

另一方面,引入基于复杂网络的聚类算法对于解决大型软件系统的测试问题大有帮助。当前,复杂网络以其特殊的性质逐渐成为研究的热点,主要集中在网络拓扑特性与模型、复杂网络上的传播行为、相继故障、搜索算法和复杂网络的同步控制等。而该理论也逐渐应用于越来越多的领域,如社会关系网、生态网络、电力网络、Internet、交通网等[3-4]。随着复杂网络研究的深入,人们发现大型软件系统中也存在复杂网络特征[5-6]。同时,为了使得不同类中的数据尽可能相异,同一类中的数据尽可能相似,人们通常把一组组个体按照相似性归并成若干类别,即聚类[7]。复杂网络聚类方法的研究对分析复杂网络的拓扑结构、理解复杂网络的功能、发现复杂网络中隐藏的规律及预测复杂网络的行为具有十分重要意义和广泛的应用前景[8-10]。

本文通过对软件系统建立加权复杂网络模型[11-12],发现软件系统的关键节点,得到关键测试路径,进而对软件测试路径进行聚类分析,压缩测试路径,减少测试数据,达到节约测试时间,提高测试效率的目的。把软件执行过程看做加权复杂网络,因为软件在运行过程中具有不确定性,在运行中发现的错误很可能是在测试阶段没能检测出来的错误[11]。因此将模糊集理论用于软件的复杂网络模型聚类当中,用较少的测试用例就可以达到较高的覆盖率。

基金项目:大连市信息产业局IT专项基金(IT Special Foundation of Dalian Information Industries Bureau under Grant No.DL20080243)。
作者简介:周宽久(1966-),男,博士,教授,CCF会员,主要研究方向为嵌入式系统工程、软件测试等;冯金金(1986-),女,硕士生,主要研究方向为嵌入式系统工程、软件测试;兰文辉(1983-),男,硕士生,主要研究方向为嵌入式系统工程、软件测试;迟宗正(1984-),男,硕士,主要研究方向为嵌入式软件功耗优化。
收稿日期:2010-03-23 **修回日期:**2010-06-21

2　软件系统复杂网络建模

自从Watts和Strogatz发现大量真实网络都具有小世界效应[5]后,复杂网络理论应用到越来越多的领域。Valverde在其2002年发表的文章中分析了JDK1.2和Ubi Soft ProRally 2002的统计特性,结果表明:这两个软件系统的结构都有很明显的“小世界”和“无标度”特性[12]。Myers、Valverde和Moura等在2003年发表的文章中分析了6个开放源代码的软件系统,除了发现软件有像网络同样具有“小世界”和“无标度”特性外,还发现出度大的节点通常入度较小,入度大的节点出度较小[12-14]。随后,研究人员在不同层次上对开源软件进行了分析,从包、类、方法的角度进一步验证以上结论。国内在此方面的研究工作刚刚开始,但是也已经进行了一些实证分析,如闫栋和韩明畅等在2006年对Java编写的一些软件系统在类级进行分析,发现了其中的“小世界”和“无标度”特征[15-16]。

2.1　复杂网络模型证明

在Linux系统上,使用GCC对GNU的开源源码包进行重新编译,并且加入一些调试信息,从而能够在软件执行过程中自动收集软件的执行路径,得到函数之间的调用关系。此处以tar、gedit、emacs程序为例。

2.1.1　小世界效应

把软件执行过程看作是无权复杂网络时,它具有明显的小世界效应。根据加权复杂网络性质的定义以及统计公式,实验得到如下数据,如表1所示。

表1　加权复杂网络统计信息表

程序	Node	Edge	S	L	C
Tar	327	699	90.656	0.429	0.003
gedit	623	1435	23.332	0.592	0.004
emacs	783	1627	28.178	0.613	0.003

从表中可以看出,平均最短距离和聚类系数明显变小。同随机网络比较,软件执行过程的加权复杂网络具有小世界效应。

2.1.2　无标度效应

对于同一个实验程序,选取测试结果的入点强度和出点强度来说明软件执行路径的加权复杂网络的点强度分布特性,图1显示了入点强度和出点强度的分布情况。

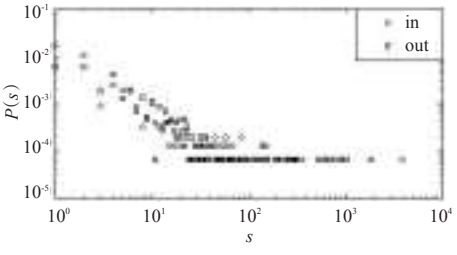

图1　入点强度和出点强度分布图

从图中可以看出,无论是入点强度还是出点强度,软件执行路径的加权复杂网络都具有明显的复杂网络特性。

在分析了软件执行路径的加权复杂网络点强度分布以后,也分析了权重的分布,图2显示了权重的分布。

从权重的分布图不难看出,软件执行路径的加权复杂网络的权重同样也具有无标度特性。

从对点强度、权重的分析来看,软件执行路径的加权复杂网络同样具有无标度特性,这是在以前的研究中不曾有的。

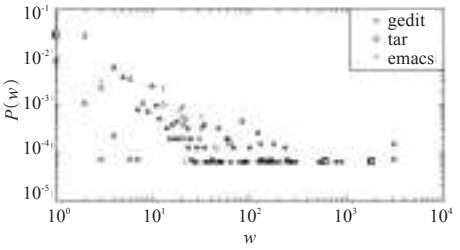

图2　加权复杂网络的权重分布

2.2　建模算法

迄今为止,学者们还没有给复杂网络一个精确的概念。学界公认的复杂网络的重要特征有度分布、平均最短路径、聚集度与聚集系数。现实世界中,网络中的边权值经常是不一样的,甚至会影响整个系统的性能。加权复杂网络能够比较完整地表达复杂网络的结构。在此,定义加权复杂网络节点中的加权度、加权聚集度和加权聚集系数。

本文1.1节再次证明了大型软件系统内部符合复杂网络模型。而基于网络系统的聚类算法研究理论基础比较扎实,这对于实现减少测试用例的目标大有帮助。因此设计了一套软件系统复杂网络模型的建立方法,主要是针对面向结构的软件。根据软件的执行过程,将执行中的每个函数看成节点,节点之间添加有向加权边,形成有向加权复杂网络。两个节点之间的距离越近,权值越大,最终建立边权值。具体算法描述为:

```
void setWeightEdge(){
    //抽取软件系统所有函数节点;
    记录整个单词名称,word;
    if(前单词为返回类型&&后面符号为()){
        记录word,即为节点;
        节点数目wordLen ++;
        }
    //建立无权网络图;
    NetNull[i][j]=1;//稀疏矩阵节点为1
    //求解节点间的相关度;
    dis(vi,vj);//两点之间路径
    c_w=ε*1/dis(vi,vj)//ε为调节系数
    //根据相关度建立加权网络图
if(word[i]与word[j]有相关度){
    WeiNet [i][j]=c_w;
    }
}
```

将软件系统中的一个函数映射为节点v_i,所有节点构成节点集V。如果两个节点之间有调用关系(包括间接调用),则对应的节点之间建立一条边,所有边构成一个边集。将建立由节点集和边集构成的一个无权复杂网络图。

节点的相关度具体表现为复杂网络图中边的权值不同。在此使用邻接矩阵softSide[n][n]存储,矩阵元素$S[i][j]$为节点i与节点j之间的边权值。不同函数调用距离越远,函数之间的相关度越低,边的权值越小。软件系统的加权复杂网络模型中的权值计算方法如下:

(1)若两个函数直接发生调用关系,则函数之间路径长度为$dis(v_i,v_j)=1$。

(2)若两个函数在一个文件中间隔n个函数发生调用关系,即间接调用时,函数之间的路径长度为$dis(v_i,v_j)=n+1$。

(3)当两个函数间有多条可以调用的路径时,定义距离$dis(v_i,v_j)=\min(dis(v_i,v_j))$。

相关度定义为:

$$c\_w(v_i,v_j)=\varepsilon*\frac{1}{dis(v_i,v_j)}$$,ε为调节系数

即如果两个函数之间路径长度为n,则这两个函数之间的相关度为$1/n$;若这两个函数在m个不同文件中都同时出现,则设两个函数在这m个文件中的相关度分别为$1/d_1$,$1/d_2$,$1/d_3\cdots 1/d_{m-1}$,$1/d_m$,则总相关度为:

$$\Delta=(1/d_1)\times m+(1/d_2)\times(m-1)+\ldots+(1/d_{m-1})\times 2+1/d_m$$

m个文件按照重要性排序,此处,假设第一个文件最重要,最后一个文件最不重要。

通过边权值可以得出每个节点在软件系统中的重要程度,显然,入度高的节点被调用次数比较多,因此其重要性比较高,一旦出错,后果比较严重。最开始时,初始化节点v_i权重为:

$$w_{in}^{v_i}=w_{out}^{v_i}=1$$

接着利用公式(1)和(2)对入权和出权进行计算:

$$w_{in}^{v_i}=\alpha\sum_{v_j\in V,v_j\to v_i}w(v_j,v_i) \quad (1)$$

$$w_{out}^{v_i}=\beta\sum_{v_j\in V,v_i\to v_j}w(v_i,v_j) \quad (2)$$

对于公式(1):v_i节点属于节点集合V,v_j到v_i存在连接边,且有权值存在,α为边权值系数;

对于公式(2):v_i节点属于节点集合V,v_i到v_j存在连接边,且有权值存在,β为边权值系数。

对于节点v_i,其权值为:

$$w_{v_i}=w_{in}^{v_i}+w_{out}^{v_i}$$

之所以权值计算要考虑到所有节点间的边,是因为深度较深的节点可能是最终实现目标的代码。真正的实现代码节点应该所占权值稍大。

3 模型聚类分析

3.1 模糊相似矩阵建立

通过第2章的相关步骤,已经建立起了软件系统的加权复杂网络模型,即可以看做一个庞大的加权图。接下来建立模糊相似矩阵。

建立模糊相似矩阵有许多种,经常用的有数量积法、夹角余弦法、统计相关系数法、最大最小法、几何平均最小法、绝对值指数法、指数相似系数法、绝对值倒数法、绝对值减数法、非参数法、贴近度法、专家打分法、模糊聚类法。模糊相似矩阵$similar_m[n][n]$是一个$n\times n$的对角线元素都为1的对称矩阵,其中$similar_m[i][j]$($similar\_m[i][j]\in[0,1]$)表示样本i与j之间相似性的量化表示,两个样本越相似,其值越接近1;否则,其值越接近0。

本文提出共同路径与权值相乘的模型来建立相似矩阵,既考虑了不同函数同被调用的相似性,也考虑到了函数本身的重要性。设$V=\{v_1,v_2,\cdots,v_{m-1},v_m\}$为软件系统复杂网络模型的所有节点集合,每个测试路径由m个指标表示其引用函数的信息:

$$v_i=\{v_{i1},v_{i2},\ldots,v_{im}\}\ (i=1,2,\cdots,m)$$

根据所有测试路径引用函数,以及函数自身权值建立原始相似矩阵***soft_node***:

$$\boldsymbol{soft\_node}=\begin{bmatrix}sn_{11} & sn_{12} & \ldots & sn_{1m}\\ sn_{21} & sn_{22} & \ldots & sn_{2m}\\ \vdots & \vdots & & \vdots\\ sn_{m1} & sn_{m2} & \ldots & sn_{mm}\end{bmatrix}$$

其中,sn_{ij}是测试路径$t_i=\{t_{i1},t_{i2},\cdots,t_{im}\}$与测试路径$t_j=\{t_{j1},t_{j2},\cdots,t_{jm}\}$引用的所有相同测试节点权值乘以引用次数之后的和。矩阵生成后,接下来进行矩阵的标准化,消除量纲,标准化之后的数据形成一个0-1的矩阵。

3.2 模糊聚类方法

比较常用的基于模糊相似矩阵的模糊聚类方法有:Prim算法和Kruskal算法、动态直接聚类法、传递闭包法、系统聚类法、动态直接聚类法、模糊C-均值法、人工神经网络模糊聚类法等。本文采用基于结构建模[17]的聚类方法,结构建模是分析系统结构特性的分析方法,而一般面向过程的软件系统具有较强的结构性。结构建模方法区别于传统的传递闭包法是能够分析系统的子模块及其之间的关系。

对于邻接矩阵***soft_node***设定一个阈值γ有:

$$s\_n\_s[i][j]=\begin{cases}1, & s\_n\_s[i][j]\geq\gamma\\ 0, & s\_n\_s[i][j]<\gamma\end{cases}$$

经过大量实验发现,如果节点个数为n,则取γ使得矩阵元素一共有n^2-8时可以得到较好的聚类结果。这样,形成一个所有元素只能是0和1的邻接矩阵。如果有回路存在,则$\boldsymbol{R}\cap\boldsymbol{R}^{\mathrm{T}}$存在满阵。否则,计算可达矩阵。

将标准差后的相似矩阵进行转换,得到对角阵或者下三角矩阵即可以得到子类,并得到子类之间的关系。矩阵变化主要是把紧紧相关的测试路径集中到一起,此处,可以定义路径的紧密度:

$$Close\_path(v_x,v_y)=\sum_{i=1}^{m}s\_n\_s[x][i]*s\_n\_s[y][i]$$

进行变化处理后的最终结果使得路径紧密度的值最大,即$\sum_{x=1}^{m}\sum_{y=1}^{m}Close_path(v_x,v_y)$值最大。

经过矩阵变化以后,可以把邻接矩阵$s_n_s[n][n]$划分成分块矩阵,存在于同一块的矩阵的路径则属于同一类。根据要求即可对测试路径进行重新分配,减少测试数据。

上面的过程用算法描述如下:

void聚类分析过程(){

读取软件结构,构造复杂网络模型;

根据测试路径,构造相似矩阵***soft_node***;

设任意两条测试路径i,j的共同测试路径为$P_1,P_2,\cdots,P_n$,权值为$\omega_1,\omega_2,\cdots,\omega_n$;

矩阵***soft_node***元素值=$P_1*\omega_1+P_2*\omega_2+\cdots+P_n*\omega_n$;

对矩阵***soft_node***进行归一化和标准化,对角元素始终不变,保持为1;

计算***soft_node***的可达矩阵,并对其进行结构分解;

对角子矩阵所对应的元素为一类;

}

3.3 实例分析

下面介绍一个简单例子,用来说明聚类过程。图3来自于McCabe,这是一张可以认为是某个程序的程序图的有向图。

根据公式$V(G)=e-n+2p$计算圈复杂度,可得从源节点到吸收节点的线性独立路径数5[19]。因此,有5个测试路径可以用来测试图3的程序。

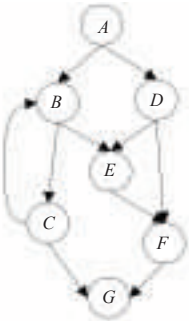

图3 程序结构图

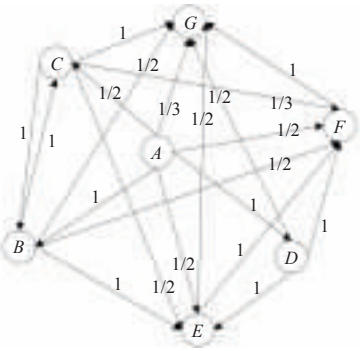

图4 复杂网络图

测试路径$p1$:A,B,C,G

测试路径$p2$:A,D,E,F,G

测试路径$p3$:A,B,E,F,G

测试路径$p4$:A,B,C,B,C,G

测试路径$p5$:A,D,F,G

仔细观察上述软件结构,可以计算出节点之间的相关度,因此可以得出此系统的复杂网络模型图如图4所示。

当α取值为2/3,β取值为1/3时,计算权重可以得出如表2所示的测试路径表。

表2 测试路径表

节点	权值	$p1$	$p2$	$p3$	$p4$	$p5$
A	23/18	1	1	1	1	1
B	7/3	1	0	1	2	0
C	35/18	1	0	0	2	0
D	3/2	0	1	0	0	1
E	5/2	0	1	1	0	0
F	23/9	0	1	1	0	1
G	23/9	1	1	1	1	1

$$\boldsymbol{s\_n\_s}=\begin{bmatrix}1.00 & 0.47 & 0.76 & 1.00 & 0.47\\ 0.43 & 1.00 & 1.00 & 0.43 & 0.89\\ 0.69 & 1.00 & 1.00 & 0.69 & 0.72\\ 1.00 & 0.47 & 0.76 & 1.00 & 0.47\\ 0.49 & 1.00 & 0.81 & 0.49 & 1.00\end{bmatrix}$$

对得到的如上的相似矩阵,根据第3章相关公式,取γ=0.5,对相似矩阵进行归一化,有:

$$\boldsymbol{s\_n\_s}=\begin{bmatrix}1 & 0 & 1 & 1 & 0\\ 0 & 1 & 1 & 0 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 0 & 1 & 1 & 0\\ 0 & 1 & 1 & 0 & 1\end{bmatrix}$$

计算可达矩阵得$\boldsymbol{R}$为满阵,整个系统为强连接。

为了让矩阵能够变换成分块矩阵,进而得出数据之间的关系,对矩阵$\boldsymbol{s_n_s}$列2,4以及行2,4进行调换[8],则得到:

$$\boldsymbol{s\_n\_s\_n}=\begin{bmatrix}1 & 1 & 1 & 0 & 0\\ 1 & 1 & 1 & 0 & 0\\ 1 & 1 & 1 & 1 & 1\\ 0 & 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 1 & 1\end{bmatrix}$$

因此测试路径分为三组:{1,4},{3},{2,5}。

路径类1:

测试路径$p1$:A,B,C,G

测试路径$p4$:A,B,C,B,C,G

路径类2:

测试路径$p3$:A,B,E,F,G

路径类3:

测试路径$p2$:A,D,E,F,G

测试路径$p5$:A,D,F,G

划分准确率100%。

因此,原系统可以简化为采取$p2$,$p3$,$p4$作为测试路径。在采用聚类分析前,需要5条测试路径。在采用该算法运算后只需要3条即可得到与原来一样的测试效果,节约了40%的测试用例。

4 实验分析

为验证软件测试路径聚类分析的实际应用价值,从嵌入式领域广泛使用的Linux2.6.24[19]内核源代码中随机选取7个代表性的文件作为研究对象,计算其聚类前后的测试路径数目分别为多少。具体如图5和图6所示,聚类前为经过圈复杂度的计算方法得到的测试用例数,聚类后为对经过圈复杂度计算得到的测试用例进行聚类分析后测试用例数。

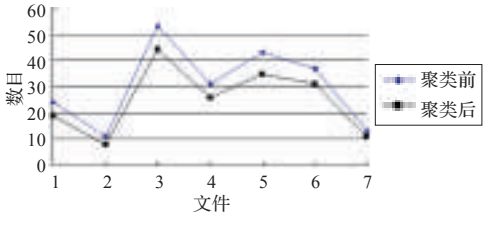

图5 测试用例数聚类前后对比

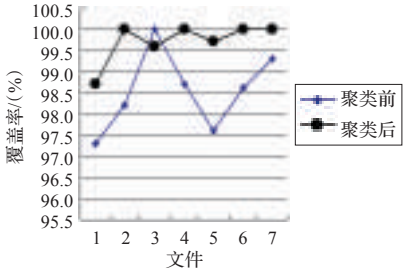

图6 测试用例覆盖率聚类前后对比

(1)文件do_mounts.c主要用来挂载Linux根设备和根文件系统,一共474行,利用圈复杂度的计算方法可得到测试路径24条,而利用聚类方法后测试路径减少到19条,节约了20.83%,覆盖率达到了98.7%。

(2)文件irq_32.c主要用来实现Linux的时间系统,一共652行,利用圈复杂度的计算方法可以得到测试路径13条,而利用聚类方法后测试路径减少到11条,节约了15.38%,覆盖率达到了100%。

(3)文件softirq.c主要用来实现支持Linux软中断部分函数,一共659行,利用圈复杂度的计算方法可以得到测试路径53条,而利用聚类方法后测试路径减少到44条,节约了16.98%,覆盖率达到了99.6%。

(4)文件main.c主要用来初始化Linux,以及启动第一个线程,一共857行,利用圈复杂度的计算方法可以得到测试路径31条,而利用聚类方法后测试路径减少到26条,节约了16.12%,覆盖率达到了100%。

(5)文件workqueue.c主要是用来实现Linux的下半部机制,文件一共883行,利用圈复杂度的计算方法可以得到测试路径43条,而利用聚类方法后测试路径减少到35条,节约了18.60%,覆盖率达到了99.7%。

(6)文件spinlock.c主要是用来实现Linux的自旋锁机制,文件一共448行,利用圈复杂度的计算方法可以得到测试路径37条,而利用聚类方法后测试路径减少到31条,节约了18.92%,覆盖率达到了100%。

(7)文件tick_sched.c主要是用来实现Linux的时间系统,文件一共652行,利用圈复杂度的计算方法可以得到测试路径13条,而利用聚类方法后测试路径减少到11条,节约了15.38%,覆盖率达到了100%。

5 结束语

对软件运行时调用的函数建立加权复杂网络,提出了软件测试路径的相似性模型,利用该模型构造相似矩阵,在此基础上进行结构分解,最终对测试路径进行聚类分析,达到减少测试路径的目的。尽管复杂网络原理已广泛应用于多种行业,但在软件测试方面却应用很少。本文将二者结合后,通过聚类分析压缩测试路径来得到更少的测试数据,且不影响测试效果。在软件的生产过程中使用该测试方法将对软件质量改善和生产效率提高起到重要作用。以作者所在实验室模拟仿真SPARC系统进行测试,整体节约19.37%的测试数据量;以Linux的三个文件进行分析,平均节约21.83%的数据量,结果验证这个算法是可行的。

软件的复杂网络模型具有社团结构,如何通过合理划分社团结构,达到更好的获得测试数据的目的,是下一步的研究目标。

参考文献:

[1] Selding P B.Faulty software caused Ariane 5 failure[J].Space News,1996,25(7):24-30.

[2] Leveson N G,Turner C S.An investigation of the Therac-25 accident[J].IEEE Computer,1993,26(7):18-41.

[3] Watts D J,Strogatz S H.Collective dynamics of "small-world" Networks[J].Nature,1998,393:440-442.

[4] Strogatz S H.Exploring complex networks[J].Nature,2001,410:268-276.

[5] Potanin A,Noble J,Frean M,et al.Scale-free geometry in object-oriented programs[C]//Comm ACM,2005,48:99-103.

[6] Myers C.Software systems as complex networks:structure,function,and evolvability of software collaboration graphs[J].Physical Rev.E,2003,68:100-104.

[7] Wheeldon R,Counsell S.Power law distributions in class relationships[C]//Proc Third IEEE Int'l Workshop Source Code Analysis and Manipulation,2003:128-218.

[8] 周宽久,王艳萍,李瑶.Web用户聚类算法[J].计算机工程与应用,2006,42(16):184-187.

[9] Girvan M,Newman M E J.Community structure in social and biological networks[C]//Proc of the National Academy of Science,2002,9(12):7821-7826.

[10] Palla G,Derenvi I,Farkas I,et al.Uncovering the overlapping community structures of complex networks in nature and society[J].Nature,2005,435(7043):814-818.

[11] Boccaletti S,Latora V,Moreno Y,et al.Complex networks:Structure and dynamics[J].Physics Reports,2006,424.

[12] 吴金闪,狄增如.从统计物理学看复杂网络研究[J].物理学进展,2004,24(1):18-46.

[13] Valverde S,Sole R.Hierarchical small worlds in software architecture[C]//Working Paper of Santa Fe Institute,2003,SFI/03-07-44.

[14] Moura A P S,Lai Y C,Motter A E.Signatures of small-world and scale-free properties in large computer programs[J].Phys Rev E,2003,68:100-102.

[15] 闫栋,祁国宁.大规模软件系统的无标度特性与演化模型[J].物理学报,2006,55(8).

[16] 韩明畅,李德毅,刘常昱,等.软件中的网络化特征及其对软件质量的贡献[J].计算机工程与应用,2006,42(20):29-31.

[17] 王众托.系统工程[M].大连:大连理工大学出版社,1990:128-148.

[18] Jorgensen P C.韩柯,软件测试[M].杜旭涛,译.2版.北京:机械工业出版社,2006:132-135.

[19] Linux2.6.24[CP/OL].http://www.kernel.org.

(上接63页)

的研究成果。如文献[9]等。在本文中,沿着该研究思路,首次在模态逻辑中提出了公式的模态真度理论,及◊真度与Δ真度。先给出在一个给定Kripke模型之下的模态真度理论,此后利用均匀概率思想,提出了更为合理的(n)模态真度理论,定义了两公式之间的(n)模态相似度,并由此导出了(n)模态伪距离,得到了相应的模态度量空间。本文的结果同文[9]相比更能体现模态词的思想特点。此外,本文的结果只是一个较为宽泛的框架而已,沿着该研究思路,有大量的工作可做,例如在模态逻辑度量空间讨论逻辑词的连续性问题,以及如何将本文的研究思路应用于粗糙逻辑等。我们将会在后续的工作中继续予以研究。

参考文献:

[1] 王国俊,傅丽,宋建设.二值命题逻辑中命题的真度理论[J].中国科学:A辑,2001,31(11):998-1008.

[2] 王国俊,李璧镜.Lukasiewicz n值命题逻辑中公式的真度理论和极限定理[J].中国科学:E辑,2005,35(6):561-569.

[3] WANG G J,SHE Y H.A topological characterization of consistency of logic theories in propositional logic[J].Mathematical Logic Quarterly,2006,52(5):470-477.

[4] 李骏,黎锁平,夏亚峰.n-值Lukasiewicz逻辑中命题的真度理论[J].数学学报,2004,47(4):769-780.

[5] 惠小静,王国俊.经典推理模式的随机化研究及应用[J].中国科学:E辑,2007,37(6):801-812.

[6] 王国俊.计量逻辑学(I)[J].工程数学学报,2006,23(2):191-215.

[7] WANG G J,ZHOU H J.Introduction to mathematical logic and resoulution principie[M].Co-published by Oxford:Alpha Science International Limited and Beijing:science Press,2009.

[8] Wang G J,Qin X Y,Zhou X N.An intrinsic fuzzy set on the universe of discourse of predicate formulas[J].Fuzzy Sets and Systems,2006,157:3145-3158.

[9] 王国俊,段巧林.模态逻辑中的(n)真度理论与和谐定理[J].中国科学:F辑,2009,39(2):234-245.

[10] 王国俊.非经典数理逻辑与近似推理[M].2版.北京:科学出版社,2008.