doi:10.3772/j.issn.1002-0470.2020.04.002

一种无监督的软件复杂度度量与评估模型①

柯文俊②* ** *** 王泊涵③*** **** 杜泽峰*** ***** 姜 利*** ****** 缪沛恩*** *******

(*中国科学院计算技术研究所 北京 100190)

(**中国科学院大学 北京 100049)

(***北京计算机技术及应用研究所 北京 100089)

(****国防科技大学信息工程学院 长沙 410073)

(*****中国地质大学信息工程学院 北京 100190)

(******中国科学技术大学软件工程学院 合肥 230051)

(*******南昌大学信息工程学院 南昌 330000)

摘 要 软件复杂度度量作为软件工程的重要组成部分,可为软件的控制和降解、软件测试的资源分配和高质量软件的研制提供支撑。计算机控制软件往往规模复杂,开发、测试和维护难度大,其复杂度的准确度量意义重大。然而,现有方法大多依赖少量度量元或者人工设置各度量元的超参数权重,工作量大且准确度难以保证。本文提出了一种无监督的、自适应的软件复杂度度量算法,对度量元数据进行高斯混合模型(GMM)的概率建模和归一化;通过皮尔逊相关系数过滤度量元间的涌现特征,借助拓扑排序和图论思想,构建度量元的自适应线性加权模型,综合评估软件的复杂度。在3种数据集上的实验表明,本文提出的软件复杂度度量模型在定量和定性分析结果上取得了更好的度量和评估效果,可以有效解决软件的复杂度度量问题。

关键词 计算机控制软件;高斯混合模型(GMM);复杂度度量;顶点表示活动(AOV)网络;概率归一化

0 引 言

随着计算机控制系统智能化、信息化的深入,计算机控制系统越来越成为一种软件密集型装备,软件作为"计算机控制系统"的核心被广泛使用,计算机控制系统软件的规模呈超几何级数增长。随着软件规模的增长,软件从各方面都会变得越来越复杂,这给软件项目开发、测试、维护带来了重重困难。

针对高复杂度软件,要求设计时就找到合适的设计模式和开发方法,控制软件复杂度与性能,解决软件维护保障困难,保证其可信属性,保证软件质量,提高开发效率。通过度量和评估,掌握软件开发过程的复杂度数据,并加以分析、利用,可将软件过程管理发挥作用,使得软件开发工作持续改进、发展。

软件度量作为软件工程的重要组成部分,在软件工程及软件过程中的作用不言而喻。软件复杂性度量作为软件度量的重要分支,可以有效评估软件产品的复杂性。通过复杂性度量对软件复杂性进行预警,可以帮助人们客观地分析和评估软件生命周期不同阶段的复杂度情况,并采取一定的复杂度控

① 国防基础科研(JCKY2016204B503)和装发预研基金(61400020103,61400020403)资助项目。

② 男,1990年生,博士生;研究方向:自然语言处理和人工智能;E-mail:kewenjun@ict.ac.cn

③ 通信作者,E-mail:cnpeking@qq.com

(收稿日期:2019-05-05)

制和降解措施,来降低软件缺陷率,提高软件质量。本文旨在通过软件复杂性的度量与综合评估,改进软件开发过程、降低软件缺陷率、改进软件评估体系,从而完善计算机控制系统软件过程管理、提升软件质量、优化软件测试的资源分配,为高质量软件研制提供支撑。

软件复杂度度量元数据是一种客观存在的集合,如何针对客观数据,制定无监督的综合评估模型,成为现阶段软件工程领域的难题,本文通过研究软件复杂度度量元的概率分布,利用高斯混合模型(Gaussian mixture model,GMM)进行度量元概率归一化;进而,通过皮尔逊相关系数分析度量元间的涌现特征,建立最大权值的顶点表示活动(activity on vertex,AOV)网络,计算各领域软件度量元的融合概率,形成一种面向计算机控制系统领域的无监督软件复杂度评估模型。

1 相关工作

1.1 软件复杂度度量元及度量方法

软件的复杂度度量方法有代码行度量法[1](line of code,LOC)、Halstead 软件科学度量法[2](Halstead software science,HSS)、McCabe 圈复杂度度量法[3](McCabe cyclomatic complexity,MCC),它们依次测量软件的长度、体积和结构复杂度,是经典的测量方法。其中,MCC 方法以图论为基础,通过分析程序的控制流图来获得模块的复杂度,HSS 是在程序中操作符和操作数做统计的基础上,建立起程序的总词汇表,对程序长度、算法的潜在最小容量、实际容量、程序级别、程序难度进行度量。还有 Oviedo 的数据流度量[4](Oviedo data flow,ODF)、Henry 的信息流度量[5](Henry information flow,HIF)、Emerson 的聚合度量[6]、Yau 和 Collofello 的稳定性度量[7](Yau and Collofello's stability,YCS)等度量法,它们从测量代码模块内部或外部的逻辑关系或依赖关系的层次进行软件复杂度的度量;此外,在面向对象的软件度量方面先后出现了一批有效的度量元来度量面向对象开发软件的复杂度,包括 C&K 度量集[8]、MOOD 度量集(metrics for object oriented design)[9]、改进 C&K 度量集的 Li 度量集[10]以及 Chen&Liu[11]度量集,它们对类、类之间的关联、方法以及面向对象的各种特征进行度量;在面向对象软件的复杂度评估方面,Chidamber 和 Kemerer[8]提出的基于面向对象的度量指标(Chidamber Kemerer six,CK6),使用 C++代码的 6 个度量指标来评估复杂度的方法被广泛使用,这 6 个度量指标分别度量每一个类中所有方法的复杂度度量,继承的深度,孩子个数,一个类的响应数,对象间的耦合程度以及方法内部的聚合的缺乏度。黄光燕和李晓维[12]提出软件复杂度的变量度量方法(variable metrics method,VMM),分别度量软件的变量个数、变量距离及变量聚合缺乏度来估计软件的复杂度。它通过度量构成软件的基本元素变量来度量,更能从底层抓住软件的本质,而且该方法在实际的软件复杂度估计中十分准确有效。文献[12]提出对软件系统模型的时间/堆栈分析的软件复杂度度量方法,可减少后期系统发生错误的概率。文献[13]提出了一种通过研究软件复杂度与统一建模语言(unified modeling language,UML)模型的关联关系进而基于 UML 模型来评估软件复杂度的方法,可在软件开发的早期评估软件的复杂度。文献[14]通过代表对软件复杂性理解和直觉感受的多个概念和公式,来评估软件的复杂度,这种基于概念、公式等信息的评估方法对此前造成评估不准确的因素是弹性可恢复的。

软件的度量方法虽多,正如文献[15]指出,或许单独的一种度量方法是有效的,把它们综合起来或许就没用了。在比较软件复杂度度量的有效性的文献中,文献[16]用确凿的大型软件实例证明最简单的 LOC 度量打败了精细化度量的 CK6(CK6 度量博采众家之长,综合了许多度量方法)方法。以上的这些度量方法(从 LOC 到 CK6)度量的抽象层次太高而且抽象层次也各不相同,都没有从根本上去度量软件的最原始核心。W9 和 P3 抓住了一些软件的本质属性,但是不能解释为什么有这样的属性。而软件的变量度量方法(VMM)分别度量软件的变量个数、变量聚合度以及变量距离来估计软件的复杂度。该方法通过度量构成软件的基本元素变量,

更能从底层抓住软件的本质。

1.2 数据归一化

数据的归一化能够提高求解速度和计算精度,归一化常用方法包括线性归一化(linear scaling)、0均值标准化(z-score standardization)、非线性归一化(nonlinear scaling)、局部响应归一化(local response normalization, LRN)、批归一化(batch normalization)、局部对比归一化(local contrast normalization)等。

线性归一化把原始数据取值转换到[0,1]之间,实际使用时,不同样本集得到的 max/min 可能不同,造成归一化结果不稳定,从而使模型也不稳定。0 均值标准化转换后的数值服从均值为 0、方差为 1 的高斯正态分布,适合原始数据近似高斯分布的情况,否则归一化后的效果会很差。非线性归一化使用对数函数、指数函数、正切函数等对数据归一化,当数据分化比较大,有些很大,有些很小时,此方法可将数值映射到一个比较小的范围进行处理。局部响应归一化多用于神经网络中,对局部神经元的活动创建竞争机制,使得其中响应比较大的值变得相对更大,并抑制其他反馈较小的神经元,增强了模型的泛化能力。批归一化由 Google 于 2015 年提出,是一个深度神经网络训练的技巧,它不仅可以加快模型的收敛速度,而且在一定程度上缓解了深层网络中"梯度弥散"的问题,从而使得训练深层网络模型更加容易和稳定。局部对比归一化主要进行的是局部做减和做除归一化,它会迫使在特征矩阵中的相邻特征进行局部竞争,还会迫使在不同特征矩阵的同一空间位置的特征进行竞争,多用于神经网络中。

1.3 复杂度综合评估方法

多种软件复杂度采用多种度量元的综合评估方法(综合评判函数)被相继提出。文献[17]使用加权平均型综合评价函数来计算并比较学生所写的 2 个不同代码的复杂度。文献[18]提出 Telenor 方法从广泛的角度对 IT 复杂性进行综合评估。文献[19]提出满足 9 个 Weyuker 属性的软件复杂性综合评估方法,相比 McCabe、Halstead 和 Oviedo 等复杂性度量方法更具有鲁棒性。文献[14]为软件系统中的信息容量复杂度(information volumetric complexity, IVC)提供了一种计算方法,它捕获了软件模块中设计信息的数量,取得了良好的复杂度计算效果。

此外,还有多种度量元按权重计算的方法,软件复杂度各个度量元的重要性是不同的。文献[20]在软件可靠性评估领域,使用模糊层次分析法对评估进行管理。文献[21]在 iso/iec 25010 软件质量模型的基础上,采用模糊综合评价法和三角形模糊数层次分析法建立了软件质量的综合评价模型。

2 软件复杂度的无监督评估模型

文本首先提出了一个基于高斯混合模型的度量元概率归一化方法,旨在将不同量级的软件度量元都无监督地归一化到[0,1]之间;然后针对不同领域的软件对各类度量元敏感程度不同的特点,将软件复杂度评估定义为一个基于度量元归一化结果的线性加权模型。本文提出基于影响域 AOV 网络的特征融合算法,借助皮尔逊相关系数,分析度量元间的涌现特征,进而得到各度量元的最大影响域,并借助拓扑排序算法思想,计算得到度量元的特征权重,最终确定软件的综合复杂度。

首先用高斯混合模型分别对每个度量元数据建模,将各个度量元的高斯混合模型的分布概率作为归一化值;然后基于皮尔逊相关系数获得这些度量元间的相关关系,同时用现行回归模型拟合每 2 个度量元归一化后的软件度量元数据,获得每对度量元的斜率,作为权值;接下来用每组度量元间的皮尔逊系数,对斜率权值数据进行过滤、筛选,获得度量元间的相关性大小;最后,根据度量元间相关性大小构建 AOV 网络,获得各度量元的影响域,进而得到各度量元的重要度权值,从而可以根据这些权值对新的软件度量元数据计算软件的复杂度。其中,本文使用的度量元是由领域专家针对不同软件类分析大量软件数据、同时考虑各度量元的不足而筛选得到的。

2.1 软件程序度量元设计与选择

针对软件程序复杂度的度量,本文总结提炼常

用的程序复杂度度量元，有圈复杂度（圈复杂度度量一个模块逻辑复杂度的值）、代码行等。本文选择的度量元见表 1。

表 1 软件度量元设计及选择

度量元	度量内容	度量元简称
总行数	包括空行在内的代码行数	LN
语句数	代码中总的语句数目	SN
分支语句数	分支语句数目	BN
注释行数	注释行数	NN
函数数目	代码中函数的数目	FN
平均每个函数包含的语句数目	总的函数语句数目除以函数数目得到的值	AFSN
最复杂函数的行号	代码中最复杂函数所在的行号	CFLN
最复杂函数的复杂度	代码中最复杂函数的圈复杂度	CFC
最深层语句块的行号	代码中最深层语句块所在行号	DSL
最大函数深度	代码中最大函数深度	MFD
平均函数深度	总的函数深度除以函数数目得到的值	AFD
平均复杂度	平均圈复杂度	AC
顶层函数中的声明数目	顶层函数中声明的数目	DN0
第 1 层函数中的声明数目	第 1 层函数中声明的数目	DN1
第 2 层函数中的声明数目	第 2 层函数中声明的数目	DN2
第 3 层函数中的声明数目	第 3 层函数中声明的数目	DN3
第 4 层函数中的声明数目	第 4 层函数中声明的数目	DN4
第 5 层函数中的声明数目	第 5 层函数中声明的数目	DN5
第 6 层函数中的声明数目	第 6 层函数中声明的数目	DN6
第 7 层函数中的声明数目	第 7 层函数中声明的数目	DN7
第 8 层函数中的声明数目	第 8 层函数中声明的数目	DN8
第 9 层函数中的声明数目	第 9 层函数中声明的数目	DN9

2.2 基于高斯混合模型的度量元概率归一化算法

由于样本数据中各个特征的数量级存在较大的差异，数量级大的特征容易掩盖数量级小的特征的变化，获得较大的影响力，从而影响算法的精度、有效性和收敛速度。此外，在多类计算机控制系统中，某度量元值出现频率越高，该度量元的影响力越大。针对系统中存在的以上特点，本文使用一种新的基于高斯混合模型的度量元概率归一化算法，从概率的角度分别对数据中的各个度量元进行概率归一化处理。首先，针对数据中的各个度量元的频率分布，使用无监督的最大期望算法（expectation maximization algorithm，EM）对其进行高斯混合建模，拟合该度量元的概率密度函数；然后，基于该度量元的概率密度函数计算其累积分布函数，使用累积分布函数的值作为该度量元的归一化处理后的数值，分布函数的非降性、有界性和右连续性，保证了本概率归一化算法的有效性和准确性。

2.2.1 度量元频率分布高斯混合建模

高斯混合模型（GMM）是多个高斯分布函数进行线性组合后得到的概率分布模型，理论上可以拟合任意类型的分布。GMM 的概率分布如式（1）所示，任一分模型的高斯分布密度函数如式（2）所示。

$$p(m) = \sum_{k=1}^{K} \pi_k N(m \mid u_k, \sum_k) \tag{1}$$

$$N(m \mid u_k, \sum_k) = \frac{1}{\sqrt{2\pi}u_k} \exp\left(- \frac{(m - u_k)^2}{2\sum_k^2} \right) \tag{2}$$

其中，π_k 是系数，$\pi_k \geqslant 0$。

在大量控制类软件的样本数据度量中，与高斯

混合模型相比,使用高斯混合模型能够更好地拟合各度量元 M 的分布。在对度量元的频率分布进行高斯混合建模时,本文选择样本数据似然最大的超参数 K 值作为最终高斯混合模型的 K 值。选择算法如下:针对各个 K 值,首先使用无监督的 EM 算法进行度量元频率分布的高斯混合建模;然后计算所有样本数据的似然概率,选择似然概率最大的 K 值作为高斯混合模型的 K 值。

2.2.2 度量元的概率归一化计算

分布函数定义为随机变量 X 取值小于 x 的累积概率函数,用于描述随机变量落在任一区间上的概率,也称为概率累积函数。使用高斯混合分布拟合特征 X 的概率密度函数 $p(x)$ 时, X 的分布函数如式(3)所示。

$$F(x) = \int_{-\infty}^{X} p(x)\mathrm{d}x \tag{3}$$

分布函数是随机变量最重要的概率特征之一,可以完整地反映随机变量的统计规律,并且决定随机变量的其他概率特征。此外,分布函数具有非降性、有界性和右连续性,其取值范围是[0, 1],具有良好的统计特性。故使用特征 x 的分布函数值 $F(x)$ 作为特征 X 的归一化处理后的值,实现特征 X 的概率归一化计算, X 取 x 值的概率越大时,可以认为 x 值越重要。

在软件复杂度评估中,本文首先使用 GMM 拟合任一软件度量元的数据,然后计算得到该 GMM 模型的分布函数,则度量元数值对应的分布函数值即作为该度量元的概率归一化值,实现度量元数据的归一化。针对任一软件度量样本 $X = \{x^1, x^2, \cdots, x^j, \cdots, x^m\}$,基于高斯混合模型的度量元概率归一化结果为 $R = \{R^1, R^2, \cdots, R^j, \cdots, R^m\}$,其中第 j 个度量值 x^j 的归一化值 R^j 的计算方式如式(4)所示。

$$R^j = \int_{-\infty}^{x^j} \sum_{k=1}^{K_j} \pi_k \frac{1}{\sqrt{2\pi}u_k} \exp\left(-\frac{(x-\mu_k)^2}{2\sum_k^2}\right)\mathrm{d}x \tag{4}$$

2.3 基于 AOV 网络的软件复杂度评估算法

本文在基于高斯混合模型对度量元数据进行概率归一化的基础上,借助 AOV 网络,基于度量元在软件复杂度评估中的重要程度来赋予不同的权重,通过线性组合来计算软件的整体复杂度,对软件复杂度进行综合评估。本文的无监督复杂度度量方法采用了多种度量元,不同度量元之间相关影响,一个度量元的变化可能与其他度量元相关,本文将度量元之间的相关关系引入到度量元的权值计算中,从而获取更优度量元融合策略。

皮尔逊相关系数用来度量 2 个变量间的线性相关性,其值在 -1 与 1 之间。本文中,首先计算不同度量元间的皮尔逊相关系数,评估其相关关系的大小,得到度量元的皮尔逊相关系数矩阵 $\boldsymbol{P}$;然后,基于线性拟合方法得到不同度量元间线性相关关系的斜率大小,得到度量元的斜率矩阵 $\boldsymbol{K}$;继而,基于皮尔逊相关系数的大小与斜率的大小对斜率进行过滤处理,剔除不相关或者斜率值小于阈值的斜率,得到斜率矩阵 $\boldsymbol{T}$;进而,将斜率矩阵转化为一个有向有环图,度量元作为图中的节点,斜率矩阵中度量元间的斜率作为图中对应边的权重;接下来,通过最大生成树算法将有向有环图转换为 AOV 网;最后,基于 AOV 的权重计算每个度量元的影响域,根据度量元影响域的大小分配每个度量元的权重。基于 AOV 网络的软件复杂度评估算法,见算法 1。

算法 1 基于 AOV 网络的软件复杂度评估算法

输入: 归一化后的软件度量样本集为 $R = \{R_1, R_2, \cdots, R_n\}$,其中第 i 个软件度量样本是 $R_i = \{h_i^1, h_i^2, \cdots, h_i^m\}$,共 m 个度量元, h_i^j 是第 i 个样本的第 j 个度量元的归一化度量值,第 i 个度量元的数据集是 $S^j = \{h_1^j, h_2^j, \cdots, h_n^j\}$, $j \in [1, m]$ 且 $j \in N^+$;用户设定的阈值 β, γ。

输出: 样本各度量元权重集合 $W = \{W_1, W_2, \cdots, W_p\}$。

(1) 计算不同度量元间的皮尔逊相关系数矩阵 $\boldsymbol{P} = \boldsymbol{P}^{m\times m}$, p_{ab} 为样本集 R 中第 a 个和第 b 个度量元的数据集 S^a、S^b 间的皮尔逊相关系数,计算方式如公式(5)所示。

$$p_{ab} = \frac{Cov(S_a, S_b)}{\sqrt{D(S_a)D(S_b)}} \tag{5}$$

(2) 计算不同度量元间的斜率矩阵 $\boldsymbol{K} = \boldsymbol{K}^{m\times m}$,

k_{ab} 为样本集 R 中第 a 个和第 b 个度量元的数据集 S^a、S^b 间使用线性拟合方法计算的斜率。

(3) 斜率矩阵 $\boldsymbol{K}$ 的过滤处理。当 2 个度量元间的相关系数 p_{ab} 小于阈值 β, 或者斜率 k_{ab} 小于 γ 时,认为第 a 个和第 b 个度量元间不具备线性相关关系,置 $w_{ab}=0$, 本文 β_1 取 0.8,从而得到权重矩阵 $\boldsymbol{T}=\boldsymbol{T}^{m\times m}$, 计算方式如式(6)所示。

$$t'_{ab}=\begin{cases}0 & p_{ab}<\beta\\ k_{ab} & p_{ab}\geqslant\beta \text{ 且 } k_{ab}\geqslant\gamma\end{cases} \tag{6}$$

(4) 基于邻接矩阵 $\boldsymbol{T}$ 构造有向有环图 $G(N, E)$; 其中 $N=\{n_1, n_2, n_3, \cdots, n_m\}$, 节点 n_i 表示第 i 个度量元构成的节点; $E=\{\cdots, e_{ab}, \cdots\}$ 为有向有环图中所有边的集合,其中 e_{ab} 表示度量元 a 指向度量元 b 的边且 e_{ab} 的权重等于 t_{ab}。

(5)基于最大生成树算法将有向有环图 $G(N, E)$ 转换为 AOV 网(有向无环图) $G'(N, E')$。其中 $E'=\{\cdots, e'_{ab}, \cdots\}(E'\subseteq E)$ 表示去除无用边之后得到的有向无环图 $G'(N, E')$ 中所有边的集合, e'_{ab} 表示 n_a 指向 n_b 的边。每条边权重不变仍为 t_{ab}。

(6)计算 AOV 网 $G'(N, E')$ 中的节点 n_i (代表第 i 个度量元)的影响域 I_i, 得到各节点影响域集合 $I=\{I_1, I_2, \cdots, I_m\}$, 计算见式(7)。

$$I_i=\begin{cases}1 & i \text{ 无后续节点}\\ \sum_k t_{ik}I_k & i \text{ 有直接后续节点 } k\end{cases} \tag{7}$$

(7)计算各节点(度量元)的权重 $W=\{W_1, W_2, \cdots, W_m\}$, 其中节点 n_i 对应的权重是 W_i, 计算方式如式(8)所示。

$$W_i=\frac{I_i}{\sum_{j=1}^{m}I_i} \tag{8}$$

3 实验结果

本节首先介绍了本文使用的数据集,然后阐述了对比算法和评价指标等实验设计,最后分别从定性和定量 2 个方面分析本文算法在软件复杂度评估上的结果及其原因。

3.1 实验数据集

在 3 种数据集上验证本文所提出的模型,分别是信号处理类软件(signal processing software, SPS)、显示控制类软件(display control software, DCS)和算法控制类软件(algorithmic control software, ACS)3 类软件的度量原始数据,数量分别是 980、870、950 个。尽管不同种类控制软件的度量元存在一定的差异,本文的软件复杂度度量算法可以针对不同的软件类型,无监督地、自适应地学习各度量元的权重,赋予不同度量元以不同的重要性,在选择度量元时应该尽可能多地选择度量元。本文中,在度量 3 类软件的复杂度时,均选择了相同的度量元来分别度量软件复杂度,表 1 列出了本文 3 类软件选择的度量元。

实验时,本文将任一数据集切分为 70% 的训练数据、20% 的验证数据和 10% 的测试数据,训练数据用于构建模型,验证数据用于确定 GMM 模型的子模型数量、阈值 β 和 γ 等系数,测试数据用于输出预测效果。

为了方便分析阐述本文复杂度度量算法的有效性,表 2 罗列了部分软件的原始度量数据。

3.2 实验设计

本文采用 LOC、VMM、CK6、人工专家评估(记为 HEE)作为对比算法,与本文模型进行比较,为了方便评估,将各种方法度量的复杂度通过均值方差标准化到 0 ~ 1 之间比较。采用根均方差(root mean square error, RMSE)和均值平均精度(mean average precision, MAP)作为本文实验的评价指标,其定义如下。

根均方差是估计值与真实值之差平方的期望值的开方,是衡量平均误差的一种较方便的方法。计算公式如式(9)所示。

$$RMSE=\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_t-p_t)^2} \tag{9}$$

单个主题的平均准确率是每篇相关文档检索出后的准确率的平均值。主集合的平均准确率(MAP)是每个主题的平均准确率的平均值。MAP 是反映系统在全部相关文档上性能的单值指标。系统检索出来的相关文档越靠前(rank 越高), MAP 就可能越高。如果系统没有返回相关文档,则准确率默认为 0。

3.3 实验结果及分析

分析本文所提算法与对比算法在整体性能上的

结果,首先分别在3个数据集上构建所提出的复杂度度量模型以及对比算法模型,然后以专家度量评估结果为基准,分别计算本文模型和对比模型的复杂度度量结果与人工专家度量结果的RMSE和MAP,表3和表4分别给出了3次实验结果计算的平均值。

表2 软件度量的部分原始数据

软件	SPS1	SPS2	SPS3	DCS1	DCS2	DCS3	ACS1	ACS2	ACS3
LN	80 123	74 881	49 947	998	9 492	51 951	444	4 028	2 887
SN	23 765	21 877	4 945	580	4 534	32 662	155	1 735	2 232
BN	175	163	398	0	0	0	0	0	657
NN	35 239	33 675	12 556	407	3 237	2 064	282	2 243	425
FN	372	352	20	0	0	0	0	0	50
AFSN	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
CFC	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
MFD	5	5	4	0	1	4	0	0	0
AFD	0.06	0.07	0.07	0	0.41	1.55	0	0	2.42
AC	1.42	1.41	3.1	0	0	0	0	0	15.92
DSL	5 426	1 382	28 364	385	5 632	4 821	291	852	1 041
CFLN	10 366	59 128	17 290	286	4 825	20 462	108	2 811	1 056
DN0	22 619	20 763	3 848	580	2 667	1 131	0	0	639
DN1	912	895	130	0	1 857	4 320	0	0	751
DN2	148	134	130	0	1 857	4 871	0	0	477
DN3	63	62	16	0	0	12 023	0	0	484
DN4	21	21	1	0	0	132	0	0	679
DN5	2	2	0	0	0	0	0	0	278
DN6	0	0	0	0	0	0	0	0	119
DN7	0	0	0	0	0	0	0	0	31
DN8	0	0	0	0	0	0	0	0	16
DN9	0	0	0	0	0	0	0	0	7

表3 软件度量的RMSE实验结果

软件类	SPS	DCS	ACS
LOC	0.34	0.27	0.33
VMM	0.30	0.29	0.35
CK6	0.29	0.31	0.37
本文	**0.29**	**0.26**	**0.31**

表4 软件度量的MAP实验结果

软件类	SPS	DCS	ACS
LOC	0.88	0.88	0.79
VMM	0.89	0.86	0.86
CK6	0.84	0.81	0.87
本文	**0.91**	**0.87**	**0.89**

根据表3可知,在3种类型的软件数据集上,本文算法相对其他3种对比算法取得了更好的效果,进一步说明了本文算法的有效性。在RMSE的评估结果上,对于LOC算法,在DCS类软件上取得了相对更好的度量效果,误差最低;对于CK6算法,则在SPS类软件上取得了更好的度量效果;对于VMM算法,则在DCS类软件上取得了更好的度量效果;而本文提出的度量模型,则在3类软件上的度量效果相差不多,由此可见,LOC算法、CK6算法或者VMM算法,在软件复杂度度量上与软件类别较为相关,可能在一类软件上能够取得较好的结果,但在其他类软件上度量效果极有可能不理想,而本文提出的度量模型对不同类软件的度量结果,误差较低,且不依

赖于软件类别,是一种更优越的度量算法。此外,分析 SPS、DCS、ACS 这 3 类控制类软件,本文模型的度量效果均比对比模型取得了更好的效果。由此可见,而本文模型的无监督、自适应的权重确定方式能够根据不同的软件类别,自适应地调整不同度量元的权重,从而更好地对不同类别的软件进行复杂度的度量,适用范围更广,这充分说明了本文模型的概率归一化和 AOV 网络权重自适应算法的有效性。

表 4 中 MAP 评估指标的计算是 2 种度量算法根据软件测试数据集中复杂度度量的相对顺序来计算的,同样以人工专家的度量相对顺序为基准顺序,分别计算文本模型和对比模型的软件复杂度度量顺序与基准顺序得到,这种软件度量的相对顺序具有更具说服力的效果。由表 4 可知,根据 MAP 实验结果,各度量模型在不同类型的软件数据上的评价精准度差异相对更小,但可以看到,文本的模型仍然在 SPS、DCS、ACS 类软件上取得了最好的结果,且在不同类软件上的度量平均精准度仍然相差最小,结果说明了本文所提出模型的优越性,优于 LOC、VMM 和 CK6 算法。

本文通过 MAP 定性的评估指标以及 RMSE 定量的评估指标的实验结果,可以看出本文算法的高斯混合模型的数据概率归一化方式和基于 AOV 网络进行度量元权重的自适应确定方式的有效性和合理性,本文所提出的算法在软件复杂度度量上,对软件类型的适用范围更广,且效果更好。

4 结 论

本文提出了一个融合高斯混合模型(GMM)概率归一化和 AOV 网络自适应权值的无监督的软件复杂度度量算法,软件度量元数据的概率归一化,表征了度量元数据在同类型数据中所处的概率位置,避免了度量元间数量级不同对算法精度、有效性等的影响。此外,度量元间的关系通过皮尔逊相关系数和线性回归相关系数来计算,并通过 AOV 网络自适应地确定各度量元的影响域和权值,从而摆脱了人工专家的干预。通过在不同类型的软件度量元数据集上的实验发现,本文所提出的算法在软件复杂度度量评估上要优于作为对比的传统方法,尤其是优于基于多度量元的传统方法。但本文的工作还存在一些可改进之处,如,对度量元进行高斯混合建模时通过人工观察该度量元数据分布来选择其子模型个数的方式较为费时,可以通过参数搜索的方式自适应地找到最适合的参数。另外,根据 AOV 网络及其权值确定各度量元的权重时,是对各度量元的出度节点进行归一化加权计算方式来确定,其存在拥有不同层数出度节点的度量元的权重差异过大的问题,在未来的工作中,会更为合理地根据 AOV 网络进行权重自适应的方式来更好地解决这些问题。

参考文献

[1] Wolverton R W. The cost of developing large-scale software[J]. *IEEE Transactions on Computer*,1974, 23(6): 615-636

[2] Halstead M H. Elements of Software Science[M]. New York: Elsevier, 1977: 1-127

[3] McCabe T J. A complexity measure[J]. *IEEE Transaction on Software Engineering*, 1976, 2(4):308-320

[4] Oviedo E I. Control flow, data flow and program complexity[C]// Proceedings of IEEE Signature Conference on Computers, Software, and Applications, Chicago, USA, 1980: 146-152

[5] Henry S. Software metrics based oil information flow[J]. *IEEE Transaction on Software Engineering*, 1981,7(5): 510-518

[6] Emerson T. A discriminate metric for module cohesion [C]// Proceeding of the 7th International Conference on Software Engineering, NJ, USA, 1984: 294-303

[7] Yau S S, Collofello J S. Some stability measures for software maintenance[J]. *IEEE Transactions on Software Engineer*, 1980, 6(6):545-552

[8] Chidamber S R, Kemerer C F. A metrics suite for object oriented design[J]. *IEEE Transactions on Software Engineering*, 1994, 20(6):476-493

[9] Harrison R, Counsell S J, Nithi R V. An evaluation of the MOOD set of object-oriented software metrics[J]. *IEEE Transactions on Software Engineering*, 1998, 24(6):491-496

[10] Li W. Another metric suite for object-oriented programming[J]. *The Jounal of Sytems and Software*, 1998, 44(2):155-162

[11] Chen J, Liu J F. New metric for object oriented design [J]. *Information and Software Technology*, 1993, 35(4):232-240

[12] 黄光燕,李晓维,宫云战. 变量度量估计软件的复杂度

[J]. 装甲兵工程学院学报, 2004,18(2):13-16
[13] 付晓东, 邹平. 一种 UML 类图结构复杂度度量方法[J]. 计算机应用, 2007, 27(b06):302-307
[14] Ghazarian A. A theory of software complexity[C]//IEEE General Theory of Software Engineering, Florence, Italy, 2015: 29-32
[15] Jung H W, Lim Y K, Chung C S. Modeling change requests due to faults in a large-scale tele-communication system[J]. *Journal of Systems and Software*, 2004, 72(2):235-247
[16] Denaro G, Gavazza L, Pezze M. An Empirical Evaluation of oo Metrics[M]. Milano: Bicocca degli Arcimboldi, 2003
[17] Parker J R, Becker K. Measuring effectiveness of constructivist and behaviourist assignments in CS102[J]. *ACM Sigcse Bulletin*, 2003,35(3):40-44
[18] Sj B, Dag I K, Warlo B, et al. The challenge of assessing and controlling complexity in a large portfolio of software systems[C]// International Conference on Product Focused Software, Limerick, Ireland, 2010: 71-74
[19] Misra S, Misra A K. Evaluation and comparison of cognitive complexity measure[J]. *ACM SIGSOFT Software Engineering Notes*, 2007,32(2):1-5
[20] Febrero F, Moraga M A, Calero C. Software reliability as user perception: application of the fuzzy analytic hierarchy process to software reliability analysis[C]//IEEE International Conference on Software Quality, Reliability and Security, Prague, Czech Republic, 2017: 224-231
[21] Wang J, Wang H, Jiang Y, et al. Research on software quality evaluation model based on triangular fuzzy number analytic hierarchy process[J]. *Computer & Digital Engineering*, 2017,45(9):1693-1697

An unsupervised measurement and evaluation model of software complexity

Ke Wenjun[*,**,***], Wang Bohan[***,****], Du Zefeng[***,*****], Jiang Li[***,******], Miao Peien[***,*******]

([*] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100049)

([***] Beijing Institute of Computer Technology and Application, Beijing 100089)

([****] Institute of System Engineering, National University of Defense Technology, Changsha 410073)

([*****] Department of Information Engineering, China University of Geosciences, Beijing 100190)

([******] Department of Software Engineering, University of Science and Technology of China, Hefei 230051)

([*******] Department of Information Engineering, Nanchang University, Nanchang 330000)

Abstract

As an important part of software engineering, software complexity measurement supports for software control, degradation and resource allocation in the field of software testing and high-quality software development. Computer control software is often complex and difficult to develop, test and maintain. So it is significant to measure its complexity accurately. However, existing methods often use a small number of metric metadata or set the weights of each metric metadata manually, which makes it difficult to guarantee the accuracy and workload. An unsupervised and adaptive software complexity measurement algorithm is proposed. Firstly, an probability modeling and normalization of each metric metadata are carried out with Gaussian mixture model (GMM). Then, with correlation of each two metric metadata filtered with Pearson correlation coefficient, an adaptive linear weighting model of those metric elements is constructed based on topological sorting and graph theory. Finally, experiments on three data sets show that the proposed complexity measurement model achieves better measurement and evaluation results in both quantitative and qualitative analysis, and measures software complexity effectively.

Key words: computer control software, Gaussian mixture model (GMM), complexity measurement, activity on vertex (AOV) network, probability normalization