软件成本评估方法综述

赵小敏 费梦钰 曹光斌 朱李楠
(浙江工业大学计算机科学与技术学院 杭州 310023)

摘 要 如何做好软件项目预算一直是政府机关、企事业单位进行信息化建设的难题之一。软件成本评估是通过一套流程或模型对软件项目开发的工作量、工期和成本进行评估的行为,可以提高软件预算的精确度,有利于保障软件项目的交付周期,合理安排和调度研发人员。首先,对软件成本评估方法进行分类介绍和对比,分析其优缺点;然后,采用软件项目样本数据,对功能点、用例点、神经网络、类推 4 种评估方法进行实验分析;最后,指出现有的软件成本评估方法存在的问题和进一步研究的方向。

关键词 软件成本评估,功能点,用例点,神经网络,类推
中图法分类号 TP311 **文献标识码** A

Review for Software Cost Evaluation Methods

ZHAO Xiao-min FEI Meng-yu CAO Guang-bin ZHU Li-nan
(College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract How to do a good job of software project budget has always been one of the difficult problems in the information construction of government agencies, enterprises and institutions. Software cost assessment is a behavior that evaluates development effort, time limit and cost of software project through a set of processes or models. It can improve the accuracy of software budget, protect the delivery cycle of software project, and arrange and schedule the research and development programmer reasonably. First of all, the software cost assessment methods were classified and compared, and their advantages and disadvantages were analyzed. Then, the experiment and analysis of four evaluation methods, including function point, use case point, neural network and analogy, were carried out with the sample data of the software project. Finally, the existing problems of the existing software cost assessment methods and the direction of further research were pointed out.

Keywords Software cost assessment, Function point, Use case point, Neural network, Analogy

1 引言

如何做好软件项目预算一直是政府机关、企事业单位进行信息化建设的难题之一。由于没有可以参照的评估方法或标准,很多单位往往依靠经办人的经验来完成软件项目预算,导致偏差很大。比如,在软件项目招投标过程中,由于没有估算出合理的成本范围,常常出现恶意低价或高价竞标的现象;在软件开发过程中,由于没有估算出合理的工作量和工期,经常出现项目延期而导致费用超预算的情况。软件成本评估是通过一套流程或模型对待估算软件项目的开发工作量、工期和成本进行评估的行为[1],适用于软件项目的预算、招投标、项目计划、变更、结算、审计或交易等各个环节。采用合理的软件成本评估方法,可以提高软件预算的精确度,有利于保障软件项目的交付周期,合理安排和调度研发人员,从而提高软件项目的质量。

最初的软件成本评估主要依靠专家的经验来完成,评估过程较慢,而且不同的专家给出的评估结果可能会有较大的偏差。之后,业界通过代码行数来衡量软件价格[2]。但是,用代码行数估算软件成本的方法只能在软件开发完成以后才能进行计算,由于不同程序设计语言、不同的开发团队实现同样功能的代码行数有很大差异,导致评估结果也有很大偏差。因此,研究人员开始寻求更精确的、更易于使用的模型估算方法,结构性成本估算模型(COnstructive COst MOdel, COCOMO)是其中最有影响力的模型,代码行数仅作为它的一个模型参数。为了克服代码行数方法依赖于程序设计语言的缺点,Symons 提出了功能点法[3]。功能点法通过量化系统功能来度量软件的规模、估算软件成本,它基于系统的逻辑设计,从用户角度分析,在整个生命周期都能进行估算。近年来,国家和地方相继发布了一些软件开发成本度量规范[4-7]。这些标准主要采用功能点法,软件行业基准数据使得采用功能点数估算软件成本有了依据。随着统一建模语言(Unified Modeling Language, UML)在软件系统中的广泛应用,基于统

本文受国家自然科学基金(61701443)资助。

赵小敏(1976-),博士,副教授,CCF 会员,主要研究方向为无线传感器网络、信息安全、软件成本评估等,E-mail:zxm@zjut.edu.cn;**费梦钰**(1992-),女,硕士生,主要研究方向为软件成本评估;**曹光斌**(1992-),男,硕士生,主要研究方向为软件成本评估;**朱李楠**(1982-),男,博士,讲师,主要研究方向为云制造、制造业信息化等,E-mail:zln@zjut.edu.cn(通信作者)。

一建模语言的规模度量方法应运而生，其中最为典型的是用例点估算方法[8]。用例点估算方法是一种根据软件开发前期的 UML 用例图，以开发人员角度分析的评估方法。另外，基于人工智能和机器学习的方法，例如神经网络技术[9]，也被逐渐应用于软件成本评估。通过各种模型评估方法的计算公式发现，软件特性因子和评估工作量之间呈非线性关系。因此，只要有大量的历史数据，通过学习训练，人工智能和机器学习方法也可用于评估软件开发的工作量和成本。

本文第 2 节对软件成本评估方法进行了分类和介绍；第 3 节分析了各种软件成本评估方法的优缺点；第 4 节采用软件项目样本数据，对功能点、用例点、神经网络、类推 4 种评估方法进行了实验分析；最后，指出现有的软件成本评估方法存在的问题和进一步研究的方向。

2　软件成本评估方法

为全面了解软件成本评估方法，采用两个与搜索阶段查询相关的论文。初始阶段主要搜索综述性论文，关键字为 software cost estimation review 和 software effort estimation review，以了解一些主要的评估方法和技术。为获取更多相关的评估方法，第二阶段主要扫描初始阶段获取的论文的参考文献和软件成本、工作量估算方面的最新文献。本文搜索相关论文的主要数据库包括知网、Elsevier ScienceDirect (ESD)、Springer、IEL(IEEE/IET Electronic Library)、ACM、Google Scholar 和 PQDT 等。将搜索到的与软件成本评估相关的论文按照使用的技术和方法进行整理和归类，现有估算方法主要包含 COCOMO 模型、功能点、用例点、回归方法、神经网络、遗传算法及启发式算法、聚类、类比、迷糊逻辑、依靠专家经验判断等方法。

根据评估方法使用技术的分类，参考其他文献的分类总结[10-15]，本文将软件成本评估方法分为基于模型的方法、基于人工智能的方法、基于类比的方法、专家决策法。其中，基于模型的方法有 COCOMO 模型、功能点模型、用例点模型等；基于人工智能的方法包含神经网络、遗传算法、启发式算法、决策树、模糊逻辑等；基于类比的方法分为类比法、类推法和聚类分析法。软件成本评估方法的具体分类如图 1 所示。

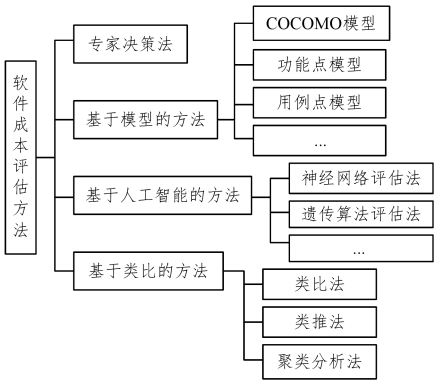

图 1　软件成本评估方法分类图

2.1　专家决策法

在传统的软件成本评估中，通常会依靠专家经验，即专家决策评估方法。它是一种由相关技术专家根据历史数据和资料以及个人经验给出评估结果，并且通过组织讨论后达成共识，最终得到软件规模、工作量、工期和成本的评估方法。

专家决策可以调和各类专家的意见，在历史数据集缺失或不足时也能进行评估。但是专家模型非常依赖于专家的经验，软件项目评估的精确性取决于专家的能力。事实上，并不能保证每个软件项目团队都有足够多的专家，并且专家的经验也很难度量，所以专家决策评估法很容易受专家主观因素的影响[16]，并不是一种非常有效的评估方法。

2.2　基于模型的方法

2.2.1　COCOMO 模型

完全依靠专家经验估算软件成本过于主观，以代码行数估算软件的方法也存在局限性，所以需要寻求一种更精确、更易于使用的模型用于软件评估。Boehm[3] 于 1981 年提出了结构化成本估算模型，被称为 COCOMO81 模型。本质上，COCOMO 模型是一种将软件特征作为参数的估算模型，分为基本、中间和详细三级 COCOMO 模型。基本 COCOMO 模型将代码行数作为自变量计算软件开发的工作量，可用于估算整个系统的工作量(Effort)。具体的模型如式(1)所示。

$$Effort = A \times Size^B \tag{1}$$

其中，$Size$ 是每人每小时开发的千行代码数，A 和 B 是固定的参数值。软件项目分为组织型、半独立型和嵌入型。对于不同分类的软件项目，参数值 A 和 B 是不同的。

中间 COCOMO 模型在基本 COCOMO 模型的基础上，对涉及产品、硬件、人员、项目等因素的调整工作量进行估算，可用于估算各个子系统的工作量。相比于基本 COCOMO 模型公式，中间 COCOMO 模型公式增加了工作量调节因子(EAF)。EAF 是由产品、硬件、人员、项目等方面的 15 个因子相乘产生的，如式(2)所示。

$$Effort = A \times EAF \times Size^B \tag{2}$$

详细 COCOMO 模型是对中间 COCOMO 模型的细化，按开发周期的不同阶段给出工作量因素分级表，因而可以估算出系统、子系统和模块 3 个层次中每个阶段的工作量。

1995 年，Boehm 等[17] 进一步提出了 COCOMOII 模型。COCOMOII 模型包含 5 个规模因子和 17 个成本驱动因子。EM_i 表示成本驱动因子，W_i 表示规模因子，模型公式如式(3)和式(4)所示。

$$Effort = A \times Size^B \times \prod_1^{17} EM_i \tag{3}$$

$$B = 1.01 + 0.01 \sum_{i=1}^{5} W_i \tag{4}$$

总体来说，COCOMO 模型方法的优点是因子比较细化，将影响工作量的因素都进行了分类和等级划分，包括产品因素、平台因素、人员因素和项目因素等，因而便于计算。但是 COCOMO 模型是基于国外数据库 161 个数据的经验得到的，特定条件下各个因子之间的内在关系还没有完全明确，所以它不能适应软件行业精细化程度越来越高的要求。COCOMO 模型是根据国外开发模式下的软件项目数据设计的，直接用它估算国内软件的成本会有较大误差，可以根据中国软件项目开发过程的实际情况，适当增加、删除或者合并一些成本驱动因子，从而找到适合中国软件开发模式的通用评估模型。通常可去掉那些仅在一些特定情况下具有局部意义的因子，以大大减小不同项目差异所造成的影响，使得所选的因子可适用于大多数的软件项目。

通过优化 COCOMO 模型的参数可以得到新的模型，比

如 SEER-SEM 模型[18]。SEER-SEM 模型比 COCOMO 模型复杂，参数超过 50 个，软件特性更详细，可以评估项目的设计、开发、交付和维护等各阶段的工作量[19]。

2.2.2 功能点模型

为了克服代码行方法依赖于程序设计语言的缺点，Symons 提出了通过量化系统功能度量软件规模的功能点法[3]。功能点法的国际标准分为 IFPUG 标准、MarkII 标准、NESMA 标准、COSMIC 标准和 FISMA 标准 5 种[20]，其中基于 IFPUG 标准的功能点法应用最为广泛。

IFPUG 标准由 20 世纪 70 年代末期 IBM 公司的工程师 Albrecht[21]首先提出，后来国际性学术组织 IFPUG(The International Function Point Users Group)在 1994 年出版了功能点计算实践手册，并于 1999 年正式命名为 IFPUG[22]。IFPUG 功能点法的思想是首先计算功能点的内部逻辑文件(ILF)、外部接口文件(EIF)、外部输入(EI)、外部输出(EO)和外部查询(EQ) 5 个要素；然后与每个要素的复杂度因子相乘，合并计算出一个初步的总功能点数(UFC)[5]；接着根据项目的具体情况对各个技术复杂度参数进行调整，功能点技术复杂度参数如式(5)所示；最后得出经调节后的功能点数，计算公式如式(6)所示。

$$TCF=0.65+0.01\sum_{i=1}^{14}E_i,E_i\in[0,5] \tag{5}$$

$$FP=UCF\times TCF \tag{6}$$

其中，E_i 是复杂度调整因子，包括数据通讯、软件性能、可配置性、事务效率、实时数据输入、用户界面复杂度、在线升级、复杂运算、代码复用性、安装简易性、操作方便性、跨平台要求、可扩展性和分布式数据处理等因素。

功能点的 14 个调整因子可以全面涵盖不同的软件开发项目，对于每一类软件开发项目可选择每个调整因子其中的一个值，但容易造成度量上的偏差。因此，功能点法在调整因子的选择方面受主观因素的影响。

2.2.3 用例点模型

随着 UML 的广泛应用，一种基于软件开发前期 UML 用例图的用例点法应运而生。与功能点法以用户角度分析不同，用例点法以开发人员角度分析，更适合开发人员对软件进行评估。1993 年，Karner 设计用例点法用于预测软件开发项目的软件规模[8]。用例点法主要通过计算系统的用例来评估软件的开发工作量，其关键点在于用例的标准化[23]和用例点到工作量之间计算方法的准确性[24-25]，评估步骤如图 2 所示。

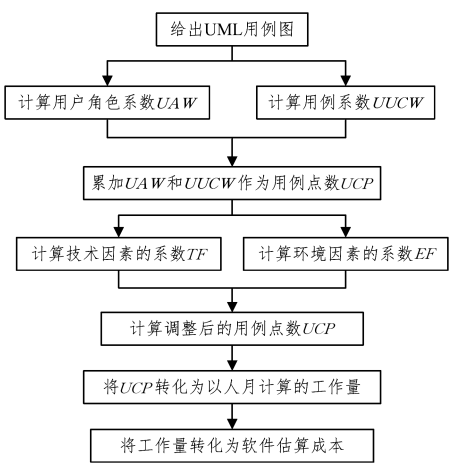

图 2 用例点法的流程图

图 2 中，用户角色系数(UAW)用于表示使用该软件的用户角色复杂度，如式(7)所示。其中 m_i 表示各种项目参与者的数量，c_i 表示相应参与者的复杂度权重值。该系数通常分为简单、平均和复杂 3 种。用例系数($UUCW$)用于表示该软件各个用例的复杂度，如式(8)所示。其中，n_i 表示各个项目用例的数目，w_i 表示相应用例的复杂度权重。每个事务根据复杂度分为简单、平均和复杂 3 个等级，不同等级的系数不同。一般只考虑与参与者有直接关系的事务。调整前的用例数($UUCP$)的计算如式(9)所示。

$$UAW=\sum_{i=1}^{3}m_i\times c_i \tag{7}$$

$$UUCW=\sum_{i=1}^{3}n_i\times w_i \tag{8}$$

$$UUCP=UUCW+UAW \tag{9}$$

计算技术因素的系数(TF)与计算环境因素的系数(EF)属于软件的非功能性需求部分，用于调整用例点数。技术因素包括分布式系统、响应或吞吐量性能目标、内部处理复杂度、代码的可重用性、可移植性、并发性事务、特殊的安全功能等 13 个因素，每个因素都有对应的权重和系数，如式(10)所示，其中 T_i 表示技术因素，W_i 表示其相应系数。该系数的取值为 0,1,2,3,4,5。越接近 0 表示该因子越无关，越接近 5 表示该因子越相关。计算环境因素包括工作团队、应用程序开发熟练度等 7 个因素，如式(11)所示，其中 E_i 表示技术因素，W_i 表示其相应系数。该系数越接近 0 表示经验越低，或者难度越低；越接近 5 表示经验越丰富，或者难度越高。用例点数 UCP 的调整如式(12)所示。

$$TF=0.6+0.01\sum_{i=1}^{13}T_i\times W_i \tag{10}$$

$$EF=1.4-0.03\sum_{i=1}^{8}E_i\times W_i \tag{11}$$

$$UCP=UUCP\times TF\times EF \tag{12}$$

最后需要将用例点数(UCP)转换为以人力计数的软件工作量。以用例点计数的工作量和以人月计数的工作量之间并不是一个完全的线性关系。因为软件规模越大，需要投入的人力越多，用于人员交流的工作量也会增加，所以真实的工作量往往会比线性计算的工作量多。研究发现，当软件规模较小(小于 200UCP)时，可以简单认为是线性关系，当软件规模比较大(大于 200UCP)时，两者的关系就不能视为线性关系，尤其当软件规模超过 1000UCP 时，如果还视为线性关系，误差就会很大[26]。为了提高软件评估的精确度，需要通过模型的公式来计算。实际应用中将用例点数目转换为以人月计数的软件工作量还需要考虑团队生产力和项目负责度，可以通过神经网络和指数线性模型等方法计算。将用例点数和软件工作量两者间的关系刻画成非线性模型，其与 COCOMO 模型公式类似，也是一种指数线性回归模型。根据 Minitab 软件统计历史数据得到该指数线性模型[26]，如式(13)所示，其中 Pro 为团队生产力因子，根据环境因子(EF)的大小以等级划分，不同等级对应不同的数值。

$$Effort=\frac{8.16}{Pro}\times Size^{1.17} \tag{13}$$

由于用例点法是根据软件项目的业务用例图或系统用例图来评估的，只要对软件项目做好需求分析或概要设计即可进行评估，即使项目的部分需求出现变更，也可以很快地修改用例数以进行二次评估。

2.3 基于人工智能的方法

基于人工智能和机器学习的技术主要包括神经网络[27-28]、支持向量机回归[29]（Support Vector Regression，SVR）、基于案例推理[30]（Case Based Reasoning，CBR）、遗传算法[31]及启发式算法[32-33]、模糊逻辑[18,34-35]等。其中，神经网络和遗传算法及启发式算法的相关研究论文较多，理论比较成熟，将作为典型技术重点阐述。

2.3.1 神经网络评估法

软件成本评估的影响因素有很多，比如程序员能力、开发灵活度、软件先例性等，所以软件的成本评估属于复杂的非线性预测问题。神经网络技术是模拟生物神经系统，对输入进行每一层网络学习检测和调整，使输出接近于理想输出值的一种训练模型[27]。将神经网络技术应用于软件项目工作量和成本的评估，通过一定的软件项目样本对网络进行训练，形成具有对软件项目进行工作量估算能力的网络模型。神经网络的输入表示方式各式各样[28]，输入值可以是 COCOMOII 模型的 17 个成本驱动因子值，输出值是工作量，也可以是换算后的研发成本，经过反复学习验证后建立合适的神经网络结构[36]。

传统的 BP 神经网络采用误差反向传播来调整网络连接权值，容易陷入局部最优解；而粒子群算法（PSO）可以在更大的空间内搜索，能起到优化权值和阈值的作用，在一定程度上避免了陷入局部最优解的问题[37]；也可以通过人工蜂群算法优化神经网络[38-39]。人工蜂群优化神经网络输入的权重值，然后用误差方程计算误差并反向传播作为输入进行神经网络的调整。与粒子群的优化算法类似，人工蜂群算法对神经网络的优化也是在训练之前，对权重和阈值进行优化。通过对神经网络的优化，可以加快神经网络的训练，减少相应的迭代次数；同时，也可以尽可能地避免神经网络陷入局部最优解，提高软件成本的估算精确度。

Attarzadeh 等提出一种基于 COCOMOII 的神经网络模型[40]，用于提高软件成本的估算精确度。该模型有效地处理了不精确和不确定的输入，分别将第一个调整因子与 17 个成本驱动因子和第二个调整因子与 5 个规模因子以及代码行数作为两个神经网络的输入参数，将两个神经网络的输出做线性处理得到软件的成本估算，提高了软件成本估算的可靠性。

通过神经网络等人工智能技术也可以实现用例点数到工作量的计算[41-42]。常用的神经网络有多层感知器神经网络、BP 神经网络、径向基神经网络（RBF）等前馈神经网络。Nassif 等[42]在提高用例点数换算成工作量的精确度方面做了一系列研究工作，将多层神经网络和指数线性回归模型进行了对比，发现指数线性回归模型对大规模项目（大于 3000/（人/小时））的评估结果较好，多层感知神经网络对小规模项目（小于 3000/（人/小时））的评估结果较好。其他算法还有随机梯度和随机森林等[43]。

采用神经网络的方法估算软件工作量和成本需要大量的样本集进行训练，将软件项目的特征形式化为数字作为输入，学习过程类似于一个黑盒，并不能直观地了解和检验。但是，人工神经网络具有很好的非线性拟合能力和很强的自学习能力，比较适用于有基准数据的软件项目成本评估问题。

2.3.2 遗传算法评估法

为了提高模型的精确度，研究人员想到运用遗传算法和其他启发式算法。将 COCOMOII 模型公式中的 a，b 参数和输入的 15 个参数中比较有影响力的 5 个输入参数这 7 个参数作为遗传算法的染色体编码进行优化[31]。COCOMOII 的 5 个输入参数分别为软件可靠性、产品复杂性、主存储约束、软件工具的使用和开发进度。通过归一化方法使得输入参数值的区间在 0 到 1 之间，组成的遗传算法染色体类似于二进制编码。遗传算法的选择方法参照适应度函数，使用轮盘赌方法。遗传算法的适应度函数是平均相对误差（Mean Relative Error，MRE），计算方法如式(14)和式(15)所示。

$$RE_i=\frac{|Estimate_i-Actual_i|}{Actual_i} \tag{14}$$

$$MRE_i=\frac{\sum_i^n RE_i}{n} \tag{15}$$

其中，RE_i 为项目 i 估算值与实际值的相对误差（Relative Error）。遗传算法的交叉算子如式(16)和式(17)所示。

$$y_1=alpha\times x_1+(1-alpha)\times x_2 \tag{16}$$

$$y_2=alpha\times x_2+(1-alpha)\times x_1 \tag{17}$$

其中，$alpha$ 是一个区间中的随机数，x_1 和 x_2 是遗传算法中的父辈，y_1 和 y_2 是遗传算法中的子代。

遗传算法的变异算子如式(18)和式(19)所示，Max 和 Min 分别表示基因的最大值和最小值，y_2 不能超出[Min，Max]的范围，$NormalRandom$ 是正太分布的随机数。

$$Sigma=0.1\times(Max-Min) \tag{18}$$

$$y_2=y_1+Sigma\times NormalRandom \tag{19}$$

遗传算法通过将 COCOMO 中比较有影响力的输入参数进行染色体编码，通过选择、交叉和变异，以误差率的公式作为适应度函数，最后得到与实际评估结果比较相近的局部最优解。

为了加快速度和提高精度，各类启发式算法也被应用到软件成本评估中，比如蜂群算法[44]、和声算法[45]和粒子群优化算法[32]。启发式遗传算法是将启发式算法与遗传算法相结合解决最优化问题的一种算法。所有优化的遗传算法都是优化软件评估模型，降低评估误差，提高评估精度，通常搜寻结果为局部最优解，也能大大提高求解速度。启发式遗传算法的搜索范围会大大减小，适应性函数的值会减少，继承遗传算法的优点，能通过遗传算子和适应度函数移除不适合的解，增强了局部搜索能力，并且缩短了搜索时间。

2.4 基于类比的方法

基于类比的方法包括类比法、类推法和聚类分析法等，在软件需求极其模糊或不确定时，也可粗略估算出工作量、工期和成本。

2.4.1 类比法

类比法是将待评估软件项目的部分属性与类似的一组基准数据进行比对，进而获得该项目的工作量、工期或成本估算值的方法。类比法依赖于已有项目的基准数据，但软件项目往往有个性化需求，该方法不一定能适用于实际场景。

2.4.2 类推法

类推法将待评估软件项目的属性与已有项目的基准数据进行比对，选取相似度比较高的几个项目的数据集进行调整，然后对软件项目的规模、工作量、工期和研发成本进行评估。该方法首先要选取能够准确描述软件项目的属性信息，然后进行相似度的定义，并且通过属性计算相似度，将匹配的项目

筛选出来，并对相似的项目进行调整，最后完成成本估算[46]。软件项目相似性的定义方法通常有不加权的欧氏距离、基于欧氏距离的加权调整、非欧氏距离中的曼哈顿距离(AMH)和明可夫斯基距离(AMK)，其他的相似度函数定义还有基于最大属性差的度量和基于模型的指数调整等[47]。

类推法是基于历史数据集实现的，它的精确性与数据集的性质、类推的过程和评估的方法及标准有关。历史数据集数据的缺失、属性的选择、相似度的定义和适应性调整，以及不同的验证和评价标准，都会影响类推法的评估结果[48]。

类推法的关键在于相似度的定义，它体现了待评估软件项目与已有软件项目之间各个属性的相似程度，直接影响着工作量和成本估算的精确性。相似度的定义通常存在没有充分应用模型信息、参数校准不精确等问题。最常见的相似度比较方法是基于欧氏距离的，具体公式如式(20)所示，其中 c_{x_i} 和 c_{y_i} 表示不同项目的第 i 个属性。

$$Similarity(C_x,C_y)=\sqrt{\sum_{i=1}^{m}(c_{x_i}-c_{y_i})^2} \tag{20}$$

通过加权的欧氏距离公式来体现不同属性的影响度，提高评估的精确度。加权调整的欧氏距离如式(21)所示，与式(20)相比，添加了一个权重值 W。

$$Similarity(C_x,C_y)=\sqrt{\sum_{i=1}^{n}W_i(c_{x_i}-c_{y_i})^2} \tag{21}$$

加权类推法的特征属性权重值可以通过专家确定或者利用启发式算法计算出最优估算结果来确定。有学者提出通过优化的粒子群算法进行特征属性权重值的优化，采用粒子群算法对类推法估算项目的各特征属性进行权重拟合，以 MRE 和 Pred(0.25)作为评价标准来确定最佳的权重值[49]。MRE 的计算如式(14)、式(15)所示，Pred 的计算如式(22)所示。

$$Pred(0.25)=\frac{1}{n}\sum_{i=1}^{n}I\{RE_i\leqslant 0.25\} \tag{22}$$

其中，RE_i 表示相对误差，I 表示 $RE_i\leqslant 0.25$ 的样本数。因此，$Pred$ 表示估算的相对误差在 25%以内的样本百分比。

除了优化特征属性权重值，还可以通过其他方式定义软件项目的相似性。针对软件属性信息并未充分应用的问题，利用 COCOMOII 模型工作量计算公式，采用项目工作量比值公式作为相似度计算公式，较大程度地表示项目特征信息的参数[50]。

为了克服经典类推法的局限性，Idri 等[50]提出了一种模糊类推法，用以处理语义属性(分类数据)，比如高、中、低等。

2.4.3 聚类分析法

聚类分析法是将聚类分析技术应用于软件项目的成本评估，它是类推法的一种改进方法。聚类分析法的思想是将所有软件项目数据集按相似性进行分类，然后通过分类算法判别新项目所属的聚类，最后计算出所属聚类均值并将其作为评估结果。软件项目属性除了常见的数值属性，还常常涉及到离散属性，数值属性可以用 K-means 方法处理，离散属性可以用 K-modes 方法处理。K-prototypes 算法结合了 K-means 方法和 K-modes 方法，通常适用于混合属性数据聚类的处理。数值属性相异度通常采用欧氏距离公式，离散属性相异度定义为属性是否相同[51]，属性相同时相异度值为 0，属性不同时相异度值为 1。

对 K-modes 算法进行改良，将总体相异度设为离散属性和数值属性的相异度之和[52]。数值属性的相异度如式(23)所示。

$$dis_{\text{numeric}}(o_i,o_j)=\frac{\sum_{f=1}^{d}|u_{if}-v_{if}|}{\max_f-\min_f} \tag{23}$$

其中，u_{if} 和 v_{jf} 分别代表 o_i 和 o_j 在属性 f 上的取值，$\max_f$ 和 $\min_f$ 分别代表所有样本属性 f 取值的最大值和最小值。离散属性相异度的定义如下：

$$dis_{\text{text}}(o_i,o_j)=1-\frac{\alpha_{ij}}{\beta_{ij}} \tag{24}$$

$$\alpha_{ij}=\sum_{e=1}^{n}\chi(a_e,b_e) \tag{25}$$

$$\beta_{ij}=2h-\alpha_{ij} \tag{26}$$

$$\chi(a_e,b_e)=\begin{cases}1, & a_e=b_e\\ 0, & a_e\neq b_e\end{cases} \tag{27}$$

其中，α_{ij} 代表两个样本项目相对应的所有相同离散属性的个数；h 代表离散属性的维度，即离散属性的个数。

Kashyap 等[53]通过优化模型参数，改进聚类估算方法，提高了软件评估精度。将用于评估样本中的参数分为字面属性、语言属性和数值属性 3 类。字面属性包括开发平台和开发语言等。字面属性的相异度定义为：若属性相同，则相异度为 0，否则为 1。语言属性包括 COCOMO 中的 17 个成本驱动因子和 5 个规模因子。根据这些因子，参数取值从 v_l 到 e_h，把相邻级别的差值定义为 1，从而差值取值为 1 到 5。语言属性的相异度定义为：如果差值为 0，则相异度为 0；如果差值相差为±1，则相异度为 0.5；否则为 1。数值属性包括代码千行数等。数值属性的相异度距离定义为：如果相等，则差异度为 0；如果差值在 5%以内，则差异度为 0.25；如果差值在 10%以内，则差异度为 0.5；如果差值在 20%以内，则差异度为0.75；其余情况都为 1。把每个输入参数的相异度都控制在 1 之内。再通过 K 邻近算法计算得到目标项目在整个样本中所在的聚类，最后得出软件项目的评估结果。

3 各类评估方法的对比分析

不同的软件成本评估方法对软件项目评估的结果也会有偏差，目前还没有一种通用的评估方法能够很好地适用于各类软件项目在各个软件工程阶段的估算。本文选取应用广泛的几种方法进行优缺点对比分析，如表 1 所列。

从表 1 可以看出，专家决策法的优点在于仅凭专家经验和个人能力也能对全新的软件系统进行评估。其缺点是专家的专业能力和经验差异对软件评估的精确度有很大的影响。

COCOMO 模型的方法逻辑性较强，并且简单便捷。但是，确定 COCOMO 模型的计算公式非常困难，国外研究人员依靠大量的软件项目数据对模型进行验证和调整；另外，COCOMO 模型评估时的参数取值还是依靠经验，并且在参数设定时偏向国外软件开发，并不适用于国内软件的成本估算。

功能点法的优点在于从用户的功能需求角度考虑，逻辑性强，概念容易理解，并且不受代码行数的限制，适合评估功能需求明确的软件项目。但是，功能点法的理论依据并不是很严谨，因为很多功能并不一定独立，而且功能点五要素的计算和调整方法因子的选择受主观因素影响较大；除此之外，在复杂度问题的解决上也与真实情况相差很大，因此它不适用于解决逻辑较复杂的实时系统[3]，比较适用于评估业务管理信息系统软件。

表 1　各种软件成本评估方法的比较

方法名称	方法描述	优点	缺点
专家决策	相关专家根据历史资料和个人经验给出反馈信息，并且组织讨论达成共识，得到最终评估结果	对于全新的项目也能依靠专家的经验进行评估	专家的主观因素影响较大，容易造成很大的误差
COCOMO 模型	确定软件成本的特性因子和模型公式，通过模型公式计算	计算方法便捷，易于操作，模型的逻辑结构清晰	模型公式和作为输入参数的软件特性因子难以确定，输入值也依赖主观判断
功能点	通过功能点五要素和调节因子计算	具有逻辑性，适合评估功能需求明确的软件项目	功能点五要素难以计算，不适用于解决逻辑较复杂的软件系统
用例点	通过用例图中的用例数和执行者数及技术复杂度因子和环境因子计算	具有逻辑性，是一种基于功能性的测算方法，简单易用，适合开发者评估软件	比较适用于有用例图的 Web 工程项目，工作量和用例点规模以间线性关系衡量时不够准确，应用较少
神经网络	先对已有项目数据通过人工神经网络进行训练，再用于新项目的评估	具有很好的非线性拟合能力和很强的自学习能力	需要大量的样本集进行训练，学习过程不能直观地了解和检验
遗传算法	遗传算子用于提高模型方法的精度	和神经网络方法相结合可提高速度和精度	需要和其他方法相结合，过程复杂
类推法	与已有项目属性的相似性进行比对，选取高度相似的项目进行类比或类推，最后完成评估	易于理解，在需求极其模糊或不确定时，也可进行评估	需要大量的项目样本作为类似的项目数据，相似度评价方法难定义

用例点法是以开发人员的角度计算的，从 UML 的用例图计算用例点数比较方便，在软件需求分析或概要设计阶段可以通过 UML 用例图进行估算。用例法的缺点是仅适用于面向对象类语言开发的软件，而且计算系数过于依赖专家的经验，具有很大的不确定性，需要大量的项目进行验证。此外，将用例点数转换为工作量需要通过模型来实现，这也会造成估算误差。

神经网络评估法运用归纳学习的训练方法，优点在于学习和训练是自发的，获取知识和信息比较全面，具有很好的非线性拟合能力和很强的自学习能力。其缺点是需要大量样本集进行训练，不能直观地了解和检验学习过程；此外，由于神经元的输入参数是人为设计的，如果输入参数不能很好地描述项目信息，也会影响评估效果。

遗传算法评估法的优点在于编码简单，搜索能力强，几乎可以搜索到全局最优解，可以通过训练模型的参数提高精确度。但是，遗传算法的计算过程复杂，启发式优化算法又会造成过早收敛。目前，对于已有项目的历史数据，通过遗传算法能够得到与实际数据非常接近的最优解，但是对于新行业领域的软件项目，其并不能保证很好的评估结果。该方法对于合适的模型有优化作用，能提高精确度，但对于模型估算误差较大的项目往往优化效果不明显。

类推法的优点是易于理解，对软件需求比较模糊或不确定时也能根据现有的项目信息进行评估。其缺点是必须有相近的历史项目样本，软件项目间的相似性定义函数很大程度上决定着软件评估的精确性。总体来说，类推法适用于需求比较模糊的待评估软件项目，并且有大量与该项目类似的历史数据集的情形。

4　各类评估方法的实验分析

4.1　数据集和评定标准

实验数据集来自某省政府采购的软件项目数据，一共有 29 个软件项目的样本数据，将其中 24 个作为训练样本，剩余 5 个作为测试样本。所有项目都是业务应用管理系统，数据特征包含应用类型、质量特征、开发语言、开发团队背景、产品规模、执行性能等。作为测试样本的 5 个项目分别为事业单位工资信息管理系统、地方海事船舶登记管理系统、地方政府性债务管理系统、水路交通行政许可管理系统和扶贫管理信息系统，分别将其记为项目 1、项目 2、项目 3、项目 4 和项目 5。软件成交价格是通过政府采购的中标或谈判价格来确定的，称为成交价或交易价。实验的目的是验证软件评估价格的精确度，评定的标准是通过真实的软件成交价格和软件评估价格进行比较，通过相对误差计算评估的精确度。相对误差 RE 如式(14)所示，是项目真实交易价格和评估价格间的误差。平均相对误差 MRE 是用同一种评估方法来对 n 个项目进行评估得到的相对误差的平均值，可以用来评价评估方法的精确性，如式(15)所示。

同理，平均价的相对误差(Relative Error of Average Price，REAP)是用 n 种评估方法对同一个项目进行评估时得到的平均价与交易价的相对误差，计算公式如(28)所示。

$$REAP=\frac{|ActualPrice-AveragePrice|}{ActualPrice} \tag{28}$$

其中，$AveragePrice$ 为用 n 种评估方法对同一个项目进行评估时得到的平均价。平均价的平均相对误差(Mean Relative Error of Average Price，MREAP)是用 n 种评估方法对同一个项目进行评估时得到的平均价的相对误差的平均值，计算公式如式(29)所示。

$$MREAP=\frac{1}{n}\sum_{i=1}^{n}REAP_i \tag{29}$$

4.2　实验结果与分析

用 MATLAB R2014a 版软件对功能点、用例点、神经网络、类推 4 种经典的评估方法进行仿真实验：功能点法采用 IFPUG 标准；用例点评估方法的调整因子须进行本地化处理，以符合中国企业的软件开发模式；神经网络评估法采用隐藏层为一层的拓扑结构，分布 9 个节点，输入参数为软件项目改写成 COCOMO 方法的因子，输出为通过基准数据(元/人月)的评估价格；类推法基于权重的欧氏距离来计算相异度，$k=5$，即选取 5 个相似样本取平均值。采用以上 4 种评估方法分别对 5 个测试样本项目进行评估得到评估价与交易价(单位：万元)，利用式(14)计算出每一种评估方法对每个项目进行评估的相对误差 RE，采用式(15)计算出每种评估方法的平均相对误差 MRE，并运用式(28)计算出 4 种评估方法对每个项目进行评估的平均价的相对误差 $REAP$，最后采用式(29)计算出平均价的平均相对误差 $MREAP$，得到的评估结果如表 2 所列。

表 2 4 种评估方法对 5 个测试样本项目的评估结果

软件项目		功能点		用例点		神经网络		类推		平均结果	
项目	交易价	评估价	RE_i/%	评估价	RE_i/%	评估价	RE_i/%	评估价	RE_i/%	平均价	*REAP*/%
项目 1	40.00	65.92	64.80	30.85	22.88	24.13	39.68	43.96	9.90	41.00	2.50
项目 2	53.10	74.35	40.02	33.17	37.53	20.28	61.81	51.40	3.20	44.80	15.63
项目 3	80.00	89.36	11.70	89.98	12.48	105.42	31.78	48.72	39.10	83.37	4.21
项目 4	71.66	94.98	32.54	137.95	92.51	72.12	0.64	51.41	28.26	89.15	24.41
项目 5	125.00	149.31	19.45	82.42	34.06	94.72	24.22	91.27	26.98	104.43	16.46
		MRE	33.70		39.89		31.63		21.49	*MREAP*	12.64

从表 2 可以看出，每一种方法的评估价与实际的交易价都有一定的偏差，平均相对误差 *MRE* 均超过 20%。总体上，类推法的 *MRE* 值最小，主要是因为实验中的测试样本项目都是基于 Web 的业务系统，与历史数据集中的项目数据类似。如果遇到差异相对大的项目，类推法也会有较大的误差，如项目 3 的相对误差达 39.10%。用例点的 *MRE* 值最大，主要是因为用例点法没有直观地统计数据规模，只通过调整因子进行调节，所以对于数据表比较多的软件项目，评估结果往往偏低。此外，用例点法的生产力是离散属性，受环境因子的影响较大，而环境因子的选择依赖评估专家的主观判断，略微地调整环境因子也会造成生产力值较大的变化，给评估结果带来很大影响。比如，评估项目 4 时，设定的环境因子使得生产力刚好到达最大值的最低要求，属于一种临界状态，造成评估结果较高，相对误差高达 92.51%。

从表 2 还可以看出，功能点法的评估价普遍比交易价高，这可能是选择调整因子受主观判断的影响，评估过程中选择的调整因子偏高，导致 *RE* 值较大。神经网络方法对 5 个项目评估的 *RE* 偏差也较大，主要是因为神经网络估算方法需要大量的训练样本集，否则很容易过拟合，造成测试集的估算效果不佳，从而导致结果存在偏差，如项目 2 的相对误差达到 61.81%。

评估软件价格是对项目开发团队智力劳动过程所需成本的衡量，有一定的相对误差是比较正常的，在实际的软件招投标过程中，甚至会出现预算与交易价有几倍的误差。从表 2 可以看出，每个项目的 *REAP* 值都不大，*MREAP* 仅为 12.64%，比 4 种方法的 *MRE* 都小得多。通过实验表明 *MREAP* 值相对较小，因此对一个软件项目进行评估时，可采用多种不同的方法分别评估并进行交叉验证，取评估价的均值作为该软件项目的预算。在实际应用中，如出现评估价偏差较大的情况，可使用加权平均，也可去掉最低价和最高价，然后取其余评估价的均值作为软件评估价格；也可以考虑将评估结果的最低价作为软件招投标项目最低限价的参考依据，避免出现恶意低价中标的情况。

结束语 本文归纳了现有的软件成本评估方法，分析了各种软件成本评估方法的优缺点，并采用软件项目样本数据，对功能点、用例点、神经网络、类推 4 种评估方法进行了实验对比。通过实验分析发现，目前的软件成本评估方法仍然存在许多亟待解决的问题：

1）缺乏足够多的样本数据集。算法模型、神经网络和类推等评估方法的参数选择都需要大量的样本数据集。样本数据集的参数值需要由专家统一规范化确定。目前，国内已成立了软件企业评估联盟来负责整理国内外相关的数据集。

2）实用性有待进一步提高。依据工信部发布的标准和行业基准数据，可以开发基于功能点法的软件成本评估系统，但需要详细填写项目要素，对于功能不明确或业务比较复杂尤其涉及算法类的软件较难评估。

3）通用性不足。没有一种方法能对各种软件类型在各个工程阶段进行评估，现有的方法都需要在一定条件或特定的软件需求下具备良好的评估效果。

未来我们将对类推法和用例点法进一步改进，提高评估的精确度和实际可操作性。类推法中重点改进加权方法和相异度距离的衡量方法。类推法考虑到属性为可能具有相关性的软件特性，因此可以将具有相关性的马氏距离作为相异度距离衡量方法。用例点法可将生产力、调整因子的权重由离散属性转变为函数关系。此外，开发实现一个集成多种软件评估方法的应用系统，对不同软件选择合适的方法进行评估，对于多种方法都适用的情况，取几种方法评估结果的加权平均作为软件成本的参考依据。

参 考 文 献

[1] RONALD J. Software cost estimation[J]. Information & Software Technology, 1992, 34(92): 627-639.

[2] KEMERER C F. An empirical validation of software cost estimation models[M]. ACM, 1987: 416-429.

[3] SYMONS C R. Function Point Analysis: Difficulties and Improvements[J]. IEEE Transactions on Software Engineering, 1988, 14(1): 2-11.

[4] 软件研发成本度量规范：SJ/T 11463-2013[S]. 北京：中华人民共和国工业和信息化部，2013.

[5] 政府投资应用软件开发项目价格评估及计算方法：DB44/T 635-2009[S]. 广东：广东省质量技术监督局，2009.

[6] 信息化项目软件开发费用测算规范：DB11/T 1010-2013[S]. 北京：北京市质量技术监督局，2013.

[7] 中国软件行业基准数据：CSBMK-201710[S]. 北京：工业和信息化部电子工业标准化研究院，2017.

[8] KARNER G. Resource estimation for objectory projects[J]. Objective Systems SF AB, 1993, 9: 1-9.

[9] HEIAT A, COMPARISO KUMAR K V, RAVI V, et al. Software development cost estimation using wavelet neural networks[J]. Journal of Systems and Software, 2008, 81(11): 1853-1867.

[10] WEN J, LI S, LIN Z, et al. Systematic literature review of machine learning based software development effort estimation models[J]. Information & Software Technology, 2012, 54(1): 41-59.

[11] KUMAR G, BHATIA P K. Empirical Assessment and Optimization of Software Cost Estimation Using Soft Computing Techniques[M] // Advanced Computing and Communication Technologies. Singapore: Springer, 2016.

[12] SEHRA S K, BRAR Y S, KAUR N, et al. Research Patterns and Trends in Software Effort Estimation[J]. Information & Software Technology, 2017, 91: 1-21.

[13] IDRI A, HOSNI M, ABRAN A. Systematic literature review of ensemble effort estimation[M]. Elsevier Science Inc, 2016: 151-175.

[14] USMAN M，MENDES E. Effort estimation in agile software development：a survey on the state of the practice[C]// International Conference on Evaluation and Assessment in Software Engineering. ACM，2015：12.

[15] WU H，SHI L，CHEN C，et al. Maintenance Effort Estimation for Open Source Software：A Systematic Literature Review[C]// IEEE International Conference on Software Maintenance and Evolution. IEEE，2017：32-43.

[16] TAN C H，YAP K S，YAP H J. Application of genetic algorithm for fuzzy rules optimization on semi expert judgment automation using Pittsburg approach[J]. Applied Soft Computing，2012，12(8)：2168-2177.

[17] BOEHM B，CLARK B，HOROWITZ E，et al. Cost models for future software life cycle processes：COCOMO2. 0[J]. Annals of Software Engineering，1995，1(1)：57-94.

[18] DU W L，HO D，CAPRETZ L F. Improving Software Effort Estimation Using Neuro-Fuzzy Model with SEER-SEM[J]. Computer Science，2015，10(12)：51-63.

[19] PHONGPAIBUL M，AROONVATANAPORN P. Standardized cost estimation in Thai government's software development projects[C]// 2015 International Conference on Computer Science and Engineering Conference (ICSEC). IEEE，2015：1-6.

[20] JONES C. Software Industry Goals for the Years 2014 through 2018[J]. Journal of Cost Analysis & Parametrics，2014，7(1)：41-47.

[21] ALBRECHT A J. Measuring Application Development Productivity[C]// IBM Applications Development Joint Share/guide Symposium. 1979：83-92.

[22] BUNDSCHUH M，DEKKERS C. Variants of the IFPUG Function Point Counting Method[M]// The IT Measurement Compendium. Springer Berlin Heidelberg，2008：397-407.

[23] HUANCA L M，ORE S B. Factors affecting the accuracy of effort estimation in software projects using Use Case Points[J]. RISTI-Revista Iberica de Sistemas e Tecnologias de Informacao，2017，2017(21)：18-32.

[24] BADRI M，BADRI L，FLAGEOL W，et al. Source code size prediction using use case metrics：an empirical comparison with use case points[J]. Innovations in Systems & Software Engineering，2016，13(2-3)：1-17.

[25] SILHAVY P，SILHAVY R，PROKOPOVA Z. Evaluation of Data Clustering for Stepwise Linear Regression on Use Case Points Estimation[M]// Software Engineering Trends and Techniques in Intelligent Systems. Springer，2017：491-496.

[26] NASSIF A B，HO D，CAPRETZ L F. Towards an early software estimation using log-linear regression and a multilayer perceptron model[J]. Journal of Systems & Software，2013，86(1)：144-160.

[27] RIJWANI P，JAIN S. Enhanced Software Effort Estimation Using Multi Layered Feed Forward Artificial Neural Network Technique[J]. Procedia Computer Science，2016，89：307-312.

[28] KAUSHIK A，SONI A K，SONI R. An improved functional link artificial neural networks with intuitionistic fuzzy clustering for software cost estimation[J]. International Journal of System Assurance Engineering & Management，2014，7：1-12.

[29] SONG L，MINKU L L，YAO X. The impact of parameter tuning on software effort estimation using learning machines[C]// International Conference on Predictive MODELS in Software Engineering. 2013：1-10.

[30] MENDES E，WATSON I，TRIGGS C，et al. A Comparison of Development Effort Estimation Techniques for Web Hypermedia Applications[C]// International Symposium on Software Metrics. IEEE Computer Society，2002：131.

[31] GHAREHCHOPOGH F S，POURALI A. A new approach based on continuous genetic algorithm in software cost estimation[J]. Journal of Scientific Research and Development，2015，2(4)：87-94.

[32] RAO G S，KRISHNA C V P，RAO K R. Multi Objective Particle Swarm Optimization for Software Cost Estimation[M]// ICT and Critical Infrastructure：Proceedings of the 48th Annual Convention of Computer Society of India- Vol I. Springer International Publishing，2014：125-132.

[33] KUMARI S，PUSHKAR S. Software Cost Estimation Using Cuckoo Search[M]// Advances in Computational Intelligence. Singapore：Springer，2017：167-175.

[34] DU W L，CAPRETZ L F，NASSIF A B，et al. A Hybrid Intelligent Model for Software Cost Estimation[J]. Journal of Computer Science，2013，9(11)：1506-1513.

[35] MITTAL A，PARKASH K，MITTAL H. Software cost estimation using fuzzy logic[J]. Acm Sigsoft Software Engineering Notes，2010，35(1)：1-7.

[36] SARNO R，SIDABUTAR J，SARWOSR I. Comparison of different Neural Network architectures for software cost estimation[C]// International Conference on Computer，Control，Informatics and ITS Applications. IEEE，2016：68-73.

[37] BENALA T R，CHINNABABU K，MALL R，et al. A Particle Swarm Optimized Functional Link Artificial Neural Network (PSO-FLANN) in Software Cost Estimation[M]// Proceedings of the International Conference on Frontiers of Intelligent Computing：Theory and Applications (FICTA). Berlin：Springer，2013：59-66.

[38] ARAR，FARUK M，AYAN K. Software defect prediction using cost-sensitive neural network[M]. Elsevier Science Publishers B V，2015：263-277.

[39] WANI Z H，QUADRI S M K. Artificial Bee Colony-Trained Functional Link Artificial Neural Network Model for Software Cost Estimation[C]// Proceedings of Fifth International Conference on Soft Computing for Problem Solving. 2016：729-741.

[40] ATTARZADEH I，OW S H. Proposing a New Software Cost estimation Model Based on Artificial Neural Networks[J]. Ssrn Electronic Journal，2010，3：V3-487-V3-491.

[41] AZZEH M，NASSIF A B. A hybrid model for estimating software project effort from Use Case Points[J]. Applied Soft Computing，2016，49：981-989.

[42] NASSIF A B，CAPRETZ L F，HO D，et al. A Treeboost Model for Software Effort Estimation Based on Use Case Points[C]// International Conference on Machine Learning and Applications. IEEE Computer Society，2012：314-319.

[43] SATAPATHY S M，ACHARYA B P，RATH S K. Early stage software effort estimation using random forest technique based on use case points[J]. IET Software，2016，10(1)：10-17.

[44] GHAREHCHOPOGH F S，MALEKI I，TALEBI A. Using Hybrid Model of Artificial Bee Colony and Genetic Algorithms in Software Cost Estimation[C]// 2015 9th International Conference on Application of Information and Communication Technologies (AICT). IEEE，2015：102-106.

(下转第 91 页)

ver,Canada,2010:1-6.
[36] OpenFlow Switch Specification, version 1. 0. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 0. 0. pdf.
[37] SUZUKI K, SONODA K, TOMIZAWA N, et al. A Survey on OpenFlow Technologies[J]. IEICE Transactions on Communications, 2014, E97. B(2):375-386.
[38] LARA A, KOLASANI A, RAMAMURTHY B. Network innovation using openflow: A survey[J]. IEEE Communications Surveys & Tutorials, 2014, 16(1):493-512.
[39] NUNES B A A, MENDONCA M, NGUYEN X N, et al. A survey of software-defined networking: Past, present, and future of programmable networks[J]. IEEE Communications Surveys & Tutorials, 2014, 16(3):1617-1634.
[40] OpenFlow Switch Specification, version 1. 1. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 1. 0. pdf.
[41] OpenFlow Switch Specification, version 1. 2. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 2. 0. pdf.
[42] OpenFlow Switch Specification, version 1. 3. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 3. 0. pdf.
[43] OpenFlow Switch Specification, version 1. 4. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 4. 0. pdf.
[44] OpenFlow Switch Specification, version 1. 5. 0[EB/OL]. http://www. openflow. org/documents/openflow-spec-v1. 5. 0. pdf.
[45] KREUTZ D, RAMOS F M V, VERISSIMO P E, et al. Software-defined networking: A comprehensive survey[J]. Proceedings of the IEEE, 2015, 103(1):14-76.
[46] KATTA N, ALIPOURFARD O, REXFORD J, et al. Infinite cacheflow in software-defined networks[C]// Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, 2014:175-180.
[47] 李向文,吉萌,曹敏,等. 基于资源复用的 Openflow 流表存储优化方案[J]. 光通信研究,2014(2):8-11.
[48] KOGAN K, NIKOLENKO S, EUGSTER P, et al. Strategies for mitigating TCAM space bottlenecks[C]// 2014 IEEE 22nd Annual Symposium on High-Performance Interconnects (HOTI). IEEE, 2014:25-32.
[49] QIAO S, HU C, GUAN X, et al. Taming the Flow Table Overflow in OpenFlow Switch[C]// Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference. ACM, 2016:591-592.
[50] ZHOU B, GAO W, WU C, et al. AdaFlow: Adaptive Control to Improve Availability of OpenFlow Forwarding for Burst Quantity of Flows[C]// International Conference on Testbeds and Research Infrastructures. Springer International Publishing, 2014:406-415.
[51] KATTA N, ALIPOURFARD O, REXFORD J, et al. Rule-Caching algorithms for Software-Defined Networks[R]. Technical report, 2014.
[52] MOLNÁR L, PONGRÁCZ G, ENYEDI G, et al. Dataplane Specialization for High-performance OpenFlow Software Switching [C]// Proceedings of the 2016 Conference on ACM SIGCOMM 2016 Conference. ACM, 2016:539-552.
[53] WANG T, LIU F, GUO J, et al. Dynamic sdn controller assignment in data center networks: Stable matching with transfers [C]// The 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016. IEEE, 2016:1-9.
[54] IYER A, KUMAR P, MANN V. Avalanche: Data center multicast using software defined networking[C]// 2014 Sixth International Conference on Communication Systems and Networks (COMSNETS). IEEE, 2014:1-8.
[55] SHERWOOD R, GIBB G, YAP K K, et al. Flowvisor: a network virtualization layer [R]. OpenFlow Switch Consortium, 2009:1-13.
[56] AL-SHABIBI A, DE LEENHEER M, GEROLA M, et al. OpenVirteX: make your virtual SDNs programmable [C]// Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, 2014:25-30.
[57] LI L E, MAO Z M, REXFORD J. Toward software-defined cellular networks[C]// 2012 European Workshop onSoftware Defined Networking (EWSDN). 2012:7-12.
[58] AKYILDIZ I, WANG P, LIN S. SoftAir: A software defined networking architecture for 5G wireless systems[J]. Computer Networks, 2015(85):1-18.
[59] GALLUCCIO L, MILARDO S, MORABITO G, et al. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks[C]// IEEE Conference on Computer Communications (INFOCOM). 2015:513-521.
[60] GANTE D, ASLAN M, MATRAWY A. Smart wireless sensor network management based on software-defined networking[C]// 27th Biennial Symposium on Communications (QBSC). 2014:71-75.

(上接第 83 页)

[45] JAFARI S M S, ZIAADDINI F. Optimization of software cost estimation using harmony search algorithm[C]// Swarm Intelligence and Evolutionary Computation. IEEE, 2016:131-135.
[46] MITTAS N, ANGELIS L. LSEbA: least squares regression and estimation by analogy in a semi-parametric model for software cost estimation [J]. Empirical Software Engineering, 2010, 15(5):523-555.
[47] 李效云,杨达,杨叶. 一种改进的类推估算方法及案例研究[J]. 计算机应用与软件,2011,28(7):5-9.
[48] IDRI A, AMAZAL F A, ABRAN A. Analogy-based software development effort estimation: A systematic mapping and review [J]. Information & Software Technology, 2014, 58:206-230.
[49] BARDSIRI V K, JAWAWI D N, HASHIM S Z, et al. A PSO-based model to increase the accuracy of software development effort estimation[J]. Software Quality Journal, 2013, 21(3):501-526.
[50] IDRI A, HOSNI M, ABRAN A. Improved estimation of software development effort using Classical and Fuzzy Analogy ensembles[J]. Applied Soft Computing, 2016, 49:990-1019.
[51] HUANG Z. Clustering Large Data Sets With Mixed Numeric and Categorical Values[EB/OL]. http://www. doc88. com/P-7728902324900. html.
[52] BISHNU P S, BHATTACHERJEE V. Software cost estimation based on modified K-Modes clustering Algorithm[J]. Natural Computing, 2016(3):1-8.
[53] KASHYAP D, MISRA A K. Software Cost Estimation Using Similarity Difference Between Software Attributes[J]. Advances in Intelligent Systems & Computing, 2013, 236:1-6.