计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2020.20190626
57(3): 487-512, 2020

基于群体智能的软件工程方法综述

徐立鑫　　吴化尧
(计算机软件新技术国家重点实验室(南京大学)　南京　210023)
(141270043@smail.nju.edu.cn)

Collective Intelligence Based Software Engineering

Xu Lixin and Wu Huayao
(*State Key Laboratory for Novel Software Technology* (*Nanjing University*), *Nanjing* 210023)

Abstract　Collective intelligence based software engineering (CISE) aims to solve software engineering problems by techniques that exploit collective intelligence, which includes machine collective intelligence, human collective intelligence, and their combinations. CISE provides a new perspective for solving complex software engineering problems, and has become an important part of modern software development. This paper presents a survey of CISE, which systematically reviews the applications of different collective intelligence inspired techniques on solving problems of software requirements analysis, design, coding, testing and maintenance. Future research directions and challenges in the CISE area are also discussed. The goal of this study is to establish a uniform framework of CISE and provide references for the interactions and transformations between collective intelligence techniques of different levels.

Key words　collective intelligence; software engineering; search based software engineering; crowd-sourced software engineering; survey

摘　要　基于群体智能的软件工程旨在利用潜在高效的群体智能方法来解决软件工程问题,其中群体智能方法不仅包括机器群体智能,还包括人类群体智能以及人机结合群体智能.基于群体智能的软件工程研究为解决复杂软件工程问题提供了新的思路,已成为现代软件工程的重要组成部分.以软件工程生命周期中的需求分析、设计、构造、测试和维护为主线,系统梳理和总结不同层次群体智能方法在上述软件开发活动上的应用.在此基础上,为不同层次群体智能方法间的相互借鉴与转化提供参考,并探讨基于群体智能的软件工程的未来发展趋势和挑战.

关键词　群体智能;软件工程;基于搜索软件工程;众包软件工程;综述

中图法分类号　TP311

随着信息技术的飞速发展,现代软件的规模和复杂度正不断提高,开发高质量的软件已成为一项困难的工作;而软件需求的不断变化、开发周期的持续缩短以及软件产品的快速迭代更是对软件的可重用性、可扩展性和可靠性提出了更高的要求和挑战.为了应对当前软件开发面临的困境,软件工程领域

收稿日期:2019-08-30;**修回日期**:2020-01-15
基金项目:国家重点研发计划项目(2018YFB10038000);国家自然科学基金项目(61902174);江苏省自然科学基金项目(BK20190291)
This work was supported by the National Key Research and Development Program of China (2018YFB1003800), the National Natural Science Foundation of China (61902174), and the Natural Science Foundation of Jiangsu Province of China (BK20190291).
通信作者:吴化尧(hywu@nju.edu.cn)

的学者近几十年来一直在不断探索新的软件开发方法,形成了面向对象软件工程[1]、面向服务软件工程[2]、网构软件工程[3]、可信软件工程[4]和大数据软件工程[5]等一批各具特色的软件工程方法.基于群体智能的软件工程旨在利用群体智能方法来高效且富有创造力地解决软件开发各个阶段中遇到的各类复杂问题,是近年来软件工程领域研究的前沿和热点.

群体智能方法来源于智能行为在群体中的汇聚现象,其强调了在一群个体间的相互合作或竞争模式下,整个群体能表现出远超其中任何一个个体智能水平的智能能力[6].这样一群个体既可以是没有智能、或仅具有相对简单智能的动物群体,也可以是人类群体本身.对应地,对群体智能的利用也就可分为3种不同的层次:机器群体智能、人类群体智能以及人机结合群体智能.

1) 机器群体智能

机器群体智能方法利用模拟生物群体行为的进化优化算法[7]来进行复杂问题求解.基于搜索的软件工程(search based software engineering, SBSE)是机器群体智能在软件工程领域应用的一种代表性方法[8],其中软件工程问题首先被形式化地转化为优化问题,随后机器群体智能方法就能在适应值函数的指导下自动搜索平衡众多因素关系的最优解.

2) 人类群体智能

人类群体智能利用大规模人类群体的协同来进行复杂问题求解.众包软件工程(crowdsourced software engineering, CSE)是人类群体智能在软件工程领域应用的一种代表性方法[9],其通常需要3种角色的协同参与,包括问题的提出者、潜在解决问题的大规模分布式用户群体以及一个促进各方交互的公共平台.

3) 人机结合群体智能

人机结合群体智能是机器群体智能和人类群体智能的结合.对于某些软件工程问题来说,问题的某些方面可以使用计算机来自动优化,而在另一些方面上则需要手工进行求解,通过人机结合的方式有助于提高人们处理上述软件工程问题的有效性和自动化程度.

目前,在利用机器群体智能和人类群体智能来解决软件工程问题上已有很多研究工作.例如Harman等人[10]和Mao等人[11]曾分别对SBSE和CSE的基本概念和研究现状进行了全面细致的总结.然而,已有研究大多仅关注了单一层次的群体智能方法,忽视了不同层次间群体智能方法的内在联系,从而可能在一定程度上限制了基于群体智能的软件工程方法的有效应用和发展潜力.

基于此,本文首先提出了应用不同层次的群体智能方法进行问题求解的统一框架;在此基础上,系统梳理和总结了各类群体智能方法在软件工程生命周期各项活动(包含需求分析、设计、构造、测试和维护)中的应用.我们希望本文工作能为不同层次间群体智能方法的相互借鉴和转化提供参考,从而进一步增强人们利用和优化群体智能方法来解决复杂软件工程问题的能力.

1 基于群体智能的软件工程

1.1 群体智能

群体智能是在许多个体的合作和竞争中涌现的共享智慧,其在复杂问题求解上的优势主要在于群体对于个体智能的放大作用[12-15].人们对于群体智能的认识很大程度受到生物物种的启示.Winston等人[16]和Schutter等人[17]曾分别研究了蜂群和蚁群的协作行为,他们发现一群只有基本本能的个体在组成群体之后,整体上可以表现出远超其中任何一个个体的智能水平.这样一种生物群体现象随后启发人们提出了不同的进化优化算法(例如蚁群算法[18]、粒子群算法[19]和蜂群算法[20]),并将这些算法应用于解决难以通过传统方法构造和求解的现实复杂问题.

低智能生物体的群体行为让人们深刻认识到群体对智能的放大作用,大量学者因而开始考虑如何利用人群的智能解决某些难题.例如Engelbart在20世纪60年代就预言通过将人类的智力能力组织到更高层次的协同结构中可以放大人类的智力[21];而1714年英国政府公开征集能够准确测量海上船只经度的方法则是众包方法的最早实践[22].

然而,对大规模人类群体智能的挖掘和利用出现的很晚,这主要是由于物理时空对人类群体协同规模的限制,以使得人类个体之间的信息难于快速共享和传播[23].互联网的快速发展极大地克服了这个问题,为人类群体智能的大规模应用奠定了基础,大量基于人类群体智能的应用也因此逐步出现.例如免费协作、多语言的互联网百科全书Wikipedia①;

① https://en.wikipedia.org/wiki/Main_Page

利用互联网大规模人类群体来辅助开发药物的InnoCentive①;利用机器计算和人类计算[24]来高效解决任务的Mechanical Turk②;利用游戏设计蛋白质模型,为抗逆转录病毒药物的设计提供新见解的游戏平台[25];以及加强研究团体和社会中研究者的联系,丰富药物发现工作价值的众包平台[26]等都是人类群体智能应用的成功实践.

虽然人类群体智能在表现形式上与多人协作等通用的问题求解方式相似,它们在组织方式上具有本质不同:多人协作通常采用集中式层级结构,而群体智能则依赖分布式的独立个体.在软件开发活动中,人们已经发现当软件的规模和复杂度达到一定程度时,单纯增加多人协作团队的成员规模并不能提高软件开发的效率[23].与之相比,人类群体智能代表了一群相对独立、自组织的智能个体在相互交互时产生的一系列复杂行为,这些行为尤其能在群体的层次上表现出某种连贯、协调的活动,涌现的群体智能也会随着群体规模的增大而增大.计算机算力的提升使得机器群体智能可以在更大的解空间中搜索最优解;互联网的快速发展使得大规模人类群体智能得以涌现,利用人类群体智能进行软件开发成为可能.

1.2 群体智能在软件工程中的应用

群体智能作为一类通用的问题求解方法,其同样被广泛应用于解决软件开发过程中面临的各类复杂问题,这一应用趋势经过逐步发展即形成了基于群体智能的软件工程研究领域.目前基于群体智能的软件工程中已经有很多研究方向,例如基于搜索的软件工程、众包软件工程以及开源软件等,基于群体智能的软件工程也和智能化软件工程等领域有交叉.其中基于搜索的软件工程(SBSE)和众包软件工程(CSE)是该领域的2个重要且发展相对成熟的研究方向,分别代表了对机器群体智能和人类群体智能的有效探索和利用,将作为本文讨论的重点.

1.2.1 基于搜索软件工程

SBSE最早可以追溯到1976年,Miller等人[27]最早使用遗传算法来解决浮点数测试用例的生成问题.2001年Harman和Jones[8]系统性将软件工程问题转化为基于搜索的优化问题,并利用遗传算法等群智优化算法进行问题求解,标志着SBSE开始成为软件工程中的一个新研究方向.

不同于传统软件工程在问题空间中通过算法构造出一个解,SBSE将传统软件工程问题转化为优化问题,并以适应度函数作导向,利用群智优化算法在解空间中寻找问题的近似最优解.Harman等人[10]总结了应用SBSE的2个前提条件:

1) 能将问题表述成形式化的优化问题;

2) 能够定义出适应度函数(fitness function).

软件工程的研究对象和解决方案都不是物理实体,这种虚拟性使得几乎软件开发生命周期各阶段的问题都可以通过数学抽象形式化表示为优化问题.此外,软件工程中许多问题都有一组与之相关且丰富多样的软件度量标准,这些度量标准形成了良好的初始适应度函数候选集[28].软件工程问题的这2个特点使SBSE的应用成为可能:只要软件工程问题能够形式化表达为优化问题,就可以使用SBSE方法来求解.

经过长时间的发展,SBSE领域已形成了一定规模的研究者群体,基于搜索的软件工程国际研讨会(International Symposium on Search-Based Software Engineering, SSBSE)和基于搜索的软件测试国际研讨会(International Workshop on Search-Based Software Testing, SBST)每年都召开,在很多子研究领域上也都有综述论文发表[10,29-30].Harman等人尤其建立了一个SBSE领域论文数据库③,收集了该领域近2 000篇研究论文,是开展SBSE研究的重要资源.

1.2.2 众包软件工程

众包是一种新兴的开放式协作模式.2006年Howe[9,31]将众包定义为"公司或机构将需要员工完成的任务,以公开招募劳动力的形式外包给一个尚未界定(通常规模较大)的网络群体的行为",以用来描述企业是如何利用互联网将工作外包给大众.

众包软件工程使用众包解决软件工程问题.通过互联网在线平台,软件工程问题被分解成一个个任务,分配给有能力的个体解决,利用人类群体智能的作用高效高质量给出解决方案.如图1所示,众包软件工程的工作流程涉及3个主要角色:基于互联网的众包平台、需求提出方和开发者.需求提出方可以是一个公司,公司利用众包平台完成公司业务可以加快软件开发周期、节省成本,个人也可以通过众包平台发布任务,帮助自己实现软件开发.开发者可以是个人也可以是一个团队,可以根据个人或者团队的能力选择相应的开发任务.基于互联网的众包平台是软件众包的核心,管理整个软件众包项目的业务逻辑,一个理想的软件众包平台应该有在线软

① https://www.innocentive.com; ② https://www.mturk.com; ③ https://crestweb.cs.ucl.ac.uk/resources/sbse_repository/repository.html

件开发环境和工具、知识分享和协同工具、解决方案质量保证和改善工具以及项目管理工具[32].

众包软件工程属于人类群体智能在软件工程中的一种实践形式,可以极大提高软件开发速度和交付软件产品的质量.从近年来出现的众包软件工程研讨会(International Workshop on CrowdSourcing in Software Engineering, CSI-SE@ICSE)和与会文章可以看出,众包软件工程发展十分迅速,并且越来越完善,与众包软件工程相关的研究非常多,一些研究人员从不同角度对众包相关研究工作进行了总结[33-36],其中 Mao 等人[11]在 2016 年对众包技术在软件工程中的应用进行了系统的总结.

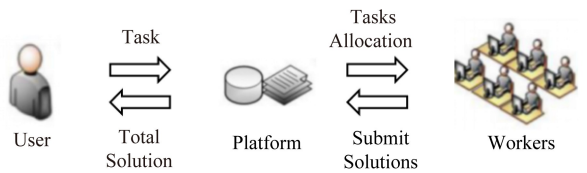

Fig. 1 The three participants in crowdsourced software engineering[11]

图 1 众包软件工程的 3 类参与者[11]

1.2.3 基于群体智能的软件工程方法统一框架

以基于搜索的软件工程为代表的研究旨在利用源自生物群体的进化优化算法来搜索软件工程问题的最佳解决方案,而以众包软件工程为代表的研究则更加关注人类群体智能的高效汇聚和应用.这些方法虽然具有不同的群体智能表现形式,但从问题求解的统一的视角来看,对于特定的问题解空间,每个个体能感知和搜索的范围通常是有限的,而群体智能能通过个体间的交互作用来更有效地探索整个问题解空间,并最终找到问题的最优解或局部最优解.

图 2 给出了利用群体智能方法进行软件工程问题求解的统一框架.对于一个给定的软件工程问题,首先群体中的每个个体分别产生问题的解决方案,随后在群体智能的帮助下对已有解决方案进行评估,并在此基础上对解决方案进行优化以产生更好的解决方案.这一过程将不断循环迭代,直到满足某种终止条件,最后在整个流程中找到的最优解即作为群体智能的最终求解结果.

产生解决方案、评估解决方案以及优化解决方案是上述基于群体智能的软件工程框架中的 3 个关键任务,这些任务既可以借助机器群体智能来实现,也可以借助人类群体智能来实现.当 3 个任务都使用机器群体智能来实现时,即代表了机器群体智能层次的基于群体智能的软件工程(例如使用蚁群算法自动化地构造、评估和优化测试数据[37]);当 3 个任务都使用人类群体智能来实现时,即代表了人类群体智能层次的基于群体智能的软件工程(例如使用众包机制公开召集一群人类个体来构造、评估和优化软件设计[38]);而当一部分任务由机器群体智能来实现,另一部分任务由人类群体智能来实现,即代表了人机结合群体智能层次的基于群体智能的软件工程.

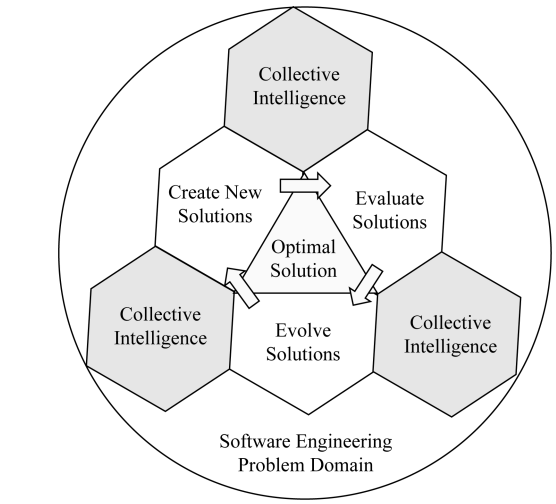

Fig. 2 A unified framework of collective intelligence based software engineering

图 2 基于群体智能的软件工程方法统一框架

具体地,图 3 给出了利用机器群体智能进行软件工程问题求解的示意图.为了自动化地使用机器群体智能构造、评估和优化候选解,软件工程问题首先需要被结构化地编码为可以被计算和搜索的形式,同时需要给出候选解的形式化评估指标,即定义结构化的适应值函数.在此基础上,以进化优化算法

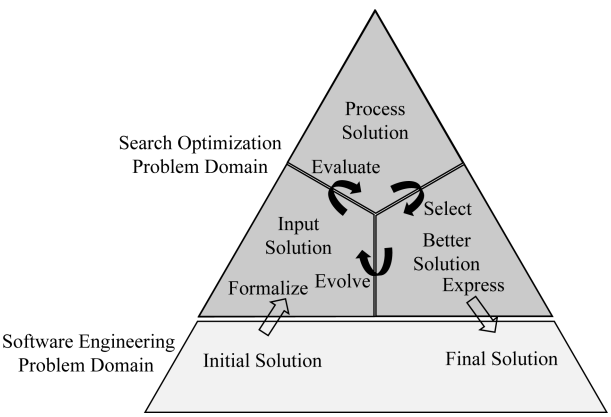

Fig. 3 Solving software engineering problem by machine collective intelligence

图 3 利用机器群体智能进行软件工程问题求解

为代表的机器群体智能方法就能在适应值函数的指导下自动化地演化候选解.

图 4 给出了利用人类群体智能进行软件工程问题求解的示意图.其中,一个管理平台是群体智能有效涌现的基础,项目请求者在平台上发布软件工程项目和获得解决方案.软件工程项目在平台中将经历产生解决方案、评估方案和选择一个比较好的解决方案的迭代过程,直到请求者获得一个满意的解决方案.在上述迭代过程中会产生许多的微任务,例如提交解决方案任务、评估其他人解决方案任务等.工人群体通过平台可以选择自己感兴趣的任务,提交任务结果.理想的平台还有工具支持,为大规模群体个体协作和通信奠定基础.

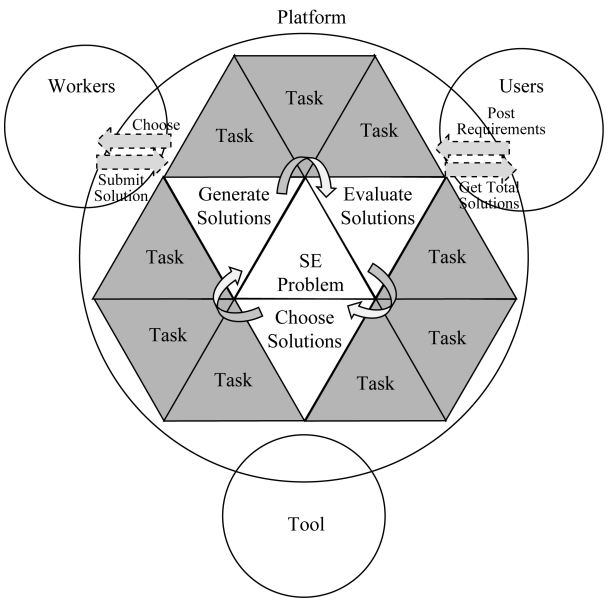

Fig. 4 Solving software engineering problem by human collective intelligence

图 4 利用人类群体智能进行软件工程问题求解

对于某些软件工程问题,其解决方案可能难于形式化描述、或者难于形式化度量,这限制了机器群体智能的应用;而人类群体智能则要求所有任务都由人类个体来完成,这又缺乏对机器运算效率的有效利用.人机结合群体智能是对上述两者的结合,图 5 给出了其对应的软件工程问题求解示意图.其中,机器群体智能既可以和人类群体智能结合,人机合力半自动化解决软件工程问题,也可以利用人群形成的知识库成为全自动化的间接人机结合群体智能来解决问题.

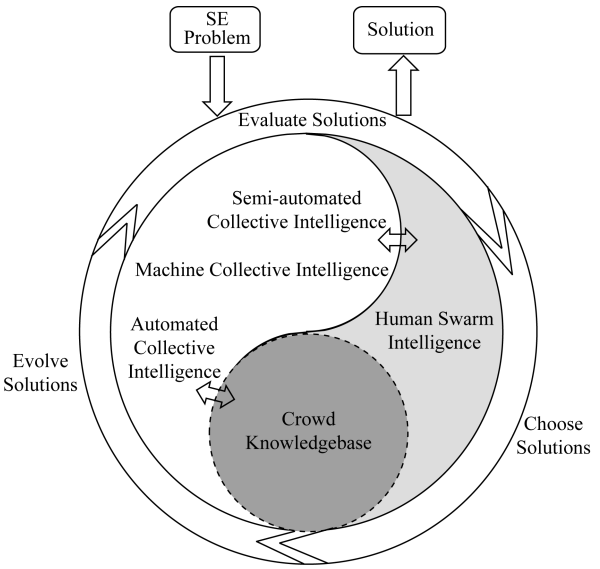

Fig. 5 Solving software engineering problems by human-machine collective intelligence

图 5 利用人机结合群体智能解决软件工程问题

1.3 基于群体智能的软件工程具体案例

基于机器群体智能、人类群体智能和人机结合群体智能的软件工程分别代表了利用群体智能进行软件工程问题求解的 3 种不同层次,它们统一于基于群体智能的软件工程框架中,同时也在很多领域都有着广泛且成熟的应用.

1.3.1 人类群体智能在软件工程中的实践

软件开发本身就是一项高度依赖人类智能的创造性活动.在基于人类群体智能的软件工程上,目前人们已开发了很多商业化的众包平台,这些平台通过不同的问题求解模型来解决软件工程问题.目前世界上最大规模的软件众包开发平台是 2001 年建立的 TopCoder①,用户可以通过 TopCoder 发布软件设计、软件开发、软件测试等任务.此外,UDesignIt②,StakeNet[39]和 StakeSource[40]专注于需求获取与分析;DesignCrowd③可以利用人类群体智能进行软件设计;Bountify④,HelpMeout[41]等可以帮助用户进行代码编写和软件实现;而软件测试领域有最多的工具与平台,例如 uTest⑤,Testin⑥,Testbirds⑦.

TopCoder 是利用人类群体智能解决软件工程相关问题的先行者.在 TopCoder 中,人类群体智能以在线竞争的形式产生、评估、优化解决方案.从顶层设计来看,系统过程类似于瀑布模型,每个开发阶

① https://www.topcoder.com; ② https://www.marketingideasforprinters.com; ③ https://www.designcrowd.com; ④ https://bountify.co; ⑤ https://www.utest.com/; ⑥ https://www.testin.cn/; ⑦ https://www.testbirds.com/

段都是由一系列在线竞争来实现的.每个参与者都产出特定任务的解决方案,由人群中选出的审查委员会对每个解决方案进行评估,解决方案的优化来自于参与者之间的竞争机制.由于人群无法处理复杂的软件问题,TopCoder将复杂任务分解为多个简单子任务发布,降低了对人类群体智能的限制,有利于充分利用人群的优势.

AppStori[①]是一个新颖的移动应用开发众包平台.和TopCoder直接利用人群智能解决问题不同,AppStori利用人群智能辅助软件开发过程,AppStori创新性地将消费者群体和软件开发过程紧密联系,消费者群体可以以众筹形式为项目提供资金;消费者群体中有能力的开发者可以加入项目的开发过程中;消费者群体可以在应用开发过程中提出建议和需求;消费者群体还可以充当原型软件测试人员的角色,为现有项目提供反馈.整个开发过程,从概念到发布,都是通过开发人员和消费者群体之间的协作来实现的,涉众群体帮助评估、优化现有的解决方案绝对是高效、高质量的人类群体智能利用方式.

StackOverflow[②]是一个编程知识问答网站.在这个平台中,人类群体以经验知识库的形式出现,提问者需要通过提问的方式主动从经验知识库中"拉出"问题解决方案.一方面,经验知识库为问题提供多种多样的解决方案,这个过程反映出人类群体智能对解决方案的优化;另一方面,提问者需要评估不同的回答找到最合适的解决方案.

1.3.2 机器群体智能在软件工程中的实践

与人类群体智能的相关实践相比,机器群体智能在软件工程中的代表性应用包括自动化测试数据生成和程序代码修复等工具,这些工具已被成功应用到很多真实的软件开发环境中.

Austin[③]是一个用于单元测试C程序的结构测试数据生成工具.Austin认为一个单元是一个被测试的函数,并且该函数中的所有函数都是可访问的.在函数代码覆盖率相关适应度函数的评估下,Austin不断生成给定函数的输入参数,形成新的测试用例,从而优化测试用例集.

EvoSuite[④]和Austin类似,不过EvoSuite针对Java语言,并且对测试过程中的Oracle问题提供了解决方案.EvoSuite通过添加小而有效的断言集来提供可能的指示,这些断言集简洁地总结了当前的行为,允许开发人员检测预期行为的偏差,并捕获当前行为,以防止未来的缺陷破坏这种行为.

GenProg[⑤]使用基因编程来修复代码缺陷.具体地来说,修复代码的解决方案以对源代码进行修改的操作序列表示.每一个候选解决方案被用于源代码,生成新的代码程序;测试套件充当评估解决方案的角色,通过的测试越多,说明该解决方案越接近完美;优化体现在不同解决方案的竞争当中.

1.4 文献筛选

为了搜集目前与基于群体智能的软件工程相关的主流研究论文,我们在这里主要考虑SBSE和CSE这2个较为成熟的基于群体智能的软件工程领域.

具体地,我们首先从SBSE和CSE对应的文献库[⑥]中获取所有相关研究论文,共计1 450篇,表1给出了这些论文的研究主题在软件工程生命周期各个阶段上的分布情况.随后,为了使本文调研工作聚焦到较高水平的研究论文上,我们进一步根据CCF期刊和会议推荐列表对上述论文进行筛选,最终共收集706篇研究论文作为本文调研的对象.

我们对上述每个软件工程阶段相关的论文进行深入分析,表1给出了基于群体智能的软件工程在软件开发各个活动上的应用和分布情况.从表1中可以看出,目前在软件工程的整个生命周期中都有基于群体智能的软件工程相关的研究和应用.

Table 1 Statistics of Literatures of Collective Intelligence Based Software Engineering

表1 基于群体智能的软件工程主流文献统计结果

Development Processes	SBSE	CSE
Requirement	95	17
Design	126	5
Coding	17	32
Testing/Verification	866	30
Maintenance	113	21
Other activities	112	16
Total	1 329	121

2 需求工程

需求工程是通过分析客户需求,帮助技术人员理解问题域并定义目标系统功能特征的方法论,是软

① https://www.appstori.com; ② https://www.stackoverflow.com; ③ https://github.com/kiranlak/austin-sbst; ④ http://www.evosuite.org; ⑤ https://squareslab.github.io/genprog-code/; ⑥ https://github.com/Rhapsod/software-crowdsourcing-papers

件开发生命周期中至关重要的一环[42].需求工程强调开发者对用户需求的理解,从而减少系统建设过程中可能出现的偏差与错误;同时,需要追踪与处理需求变更,减少需求变更带来的耗费,提高生产效率.

目前群体智能在需求工程中主要用于需求获取和分析,表2给出了不同层次的群体智能在解决具体需求工程问题上的应用.其中,机器群体智能在需求分析中主要用于解决需求冲突协商[43]、需求交互[44-48]、需求选择[49-53]和需求规范[54]等问题;人类群体智能主要在需求获取中发挥作用,如寻找需求涉众[39-40,55-57]、需求提取[58-61]和需求分析[40,55-56,62-63]等.目前,基于人机结合群体智能的需求工程研究较少,仅在需求规范[64]和需求优先级排序[65]中有相关研究.

Table 2 Research on Collective Intelligence Based Requirement Engineering

表2 基于群体智能的需求工程相关研究

Method	Problem	Number of Papers	Reference
Human Collective Intelligence	Determining Stakeholders	5	Ref[39-40,56-57]
	Requirement Extraction	17	Ref[58-61]
	Requirement Analysis	8	Ref[40,55-56,62-63]
Machine Collective Intelligence	Requirement Analysis	38	Ref[43-54]
Human-machine Collective Intelligence	Requirement Analysis	4	Ref[64-65]

2.1 涉众确定

需求获取指理解涉众的要求以及他们是如何使用新软件系统来支持他们的工作.如果项目需求分析的涉众群体不够多或者没有代表性,需求分析结果就会出现偏差,从而导致软件产品无法被用户接受.因此在需求获取之前应尽量全面地分析并寻找软件涉众.传统需求工程方法通常选取部分用户作为全体用户的代表,这会导致仅对有限的需求进行提取,从而难于适应互联网环境下大规模、多样和动态演化的用户群体和软件使用环境.

基于大规模人群涌现人类群体智能在寻找涉众任务中具有特殊优势.人类群体规模越大,涉众的多样性越丰富;同时人类群体规模越大,涌现的群体智能越强大,应用在涉众确定任务中的效果越好.

首先,可以通过涉众的背景来寻找初始的涉众群体.例如为了寻找到足够多的特定领域涉众,Wang等人[57]根据具有特定领域知识的人群会在特定的时空间聚集的现象,提出了一种新型的参与者招募框架来以更有效的方式招募所需领域的参与者.其次,涉众群体可以通过涉众之间的关系进行优化,例如Lim等人[39-40,55]提出了StakeNet方法并开发了相关工具.在StakeNet方法中,为了寻找和项目有关的所有涉众群体,让涉众推荐可能的涉众群体,在真实项目中取得了良好的效果.最后,通过将涉众关系抽象定义为网络图,定义相关度量函数来评估涉众群体是否已经满足要求.StakeNet方法中采用了PageRank等算法指标来计算涉众群体的优先级,并且在后续的工作[56]中提出了StakeRare方法,使用社交网络分析和协同过滤技术来识别大型软件工程中的需求并对需求优先级进行排序.

2.2 需求提取

需求提取是将各种形式的需求意见提取为可用于后续需求分析的规范需求文档的过程.需求提取面临的困难主要来自于从大量自然语言形式文档中提取需求的时间成本,限制了需求提取的规模.基于群体智能的软件工程方法通过增加处理问题的并行程度,缩减时间成本,从而增加了需求提取的规模.但是非专业的人群手动从大型自然语言文本源中提取需求是一项艰巨的任务,限制了人类群体智能需求提取的一般化.Breaux等人[61]进行了3个实验,雇佣未经培训的人群工作人员手动从隐私政策文档中提取需求.实验结果表明,合理的任务分解可以降低人群提取难度,提高了人类群体智能提取需求的能力.此外,Hosseini等人[58]还发现众包应用于需求提取可以适应软件需求提取的高要求,该研究深入研究了如何使用众包可以提高提取需求的质量,优化基于群体智能的软件工程方法的评估、优化解决方案过程.

人类群体智能方法充分利用了用户群体的主体作用,通过将需求提取任务以一定方式众包出去,从而尽可能全面地感知和收集需求[60].Adepetu等人[59]提出了一个概念化的众包平台CrowdREquire,人群中的个体需要向客户端定义的任务提交规范的需求文档解决方案.在竞赛机制作用下,个体会相互竞争,不断优化自身的解决方案,最后由任务提出方从人群提交的众多解决方案中选出最优的解决方案.

2.3 需求分析

需求分析包括提炼、分析和审查已经收集到的需求,以确保所有风险承担者都知道其中含义,并找到需求中冲突、错误和其他不足的地方.需求分析相关研究集中在需求优先级排序、需求分类、需求冲突协商以及需求交互研究等活动.

在需求分析问题上,对不同层次群体智能来说,非专业人类群体很难利用人类群体智能对需求分析问题进行直接集体决策,因此人类群体智能应用有限;需求优先级、客户满意度、项目成本等因素的需求分析能够表述为多目标优化问题,因此机器群体智能在需求分析问题中具有广泛的应用;通过机器群体智能对问题进行优化,转换为比较容易的子任务分配给人类群体智能解决,可以充分利用人机群体智能的优势.

目前,人类群体智能在需求分析中的应用主要是利用众包技术来进行需求优先级排序[40,55-56].此外,Nascimento 等人[62]还利用众包对需求进行分类以分析用户对产品的真正期望;而 Liang 等人[63]提出了基于 Web 2.0 的需求分析框架来辅助需求决策,以降低用户参与需求分析的难度.

当需求分析问题可以较容易地转化为结构化的优化问题时,机器群体智能方法就能得到广泛的应用,其中最有代表性的例子就是下一版本问题(next release plan, NRP),即如何根据现有资源选择新的功能加入到软件系统中来满足客户需求[64].机器群体智能在 NRP 问题中应用的关键是解决方案的评估,由于功能选择时需要考虑多方面因素,例如客户之间的优先级关系、成本和需求之间的关系等,人们尤其关注基于多目标优化的 NRP 问题.例如 Brasil 等人[46]提出了一种多目标优化公式,在考虑客户满意度、需求之间相关性、业务价值和可用资源等多种因素的条件下确定系统理想的版本数量,实验中使用带精英策略的非支配排序的遗传算法(nondominated sorting genetic algorithms Ⅱ,NSGAⅡ)和 MOCell[65]算法验证了公式的有效性.

评估 NRP 问题还需要考虑项目资源和成本,在项目资源限制内找出一组需求满足不同涉众的诉求.Zhang 等人[49]采用 Kiviat 图直观显示出预算压力对于多方涉众满意度的影响,并且比较了 Two-Archive 算法、NSGAⅡ算法和随机算法在此问题的性能.实验表明,随着问题规模扩大,Two-Archive 算法的收敛性更好.此外,项目预算不准确也会对 NRP 问题的最终结果有很大影响.Harman 等人[51]在单目标或多目标需求优化问题基础上进行了成本的可接受敏感范围分析,决策者可以根据分析结果的可视化图表识别数据中的敏感范围.

在某些情况下,机器群体智能评估需求分析解决方案还需要考虑需求交互问题[66],即某些特定目标只有多个需求同时满足时才能达到.Zhang 等人[44]修改了传统 NSGAⅡ算法[67],使之可以用于解决考虑需求交互约束的多目标需求优化评估问题.Zhang 等人[47]随后对算法进行了进一步改进,提高了算法的收敛性和最终解决方案的多样性.此外,Sureka 等人[48]将 NPR 中的需求交互看作需求之间的协同作用,分为积极协同作用和消极协同作用,研究了机器群体智能对此类问题的性能,验证了机器群体智能对 NRP 问题的适用性.

此外,针对多客户需求之间的相互冲突和竞争问题, Finkelstein 等人[43]提出了 3 种公平性衡量公式来评估和客户优先级相关的需求分析解决方案,并且利用遗传算法寻求需求公平性权衡的需求协商方案.针对用户输入需求的不准确或不完全问题, Devries 等人[54]结合符号分析和演化计算将需求进行分解,枚举所有必要的分解需求和确定分解需求的影响范围.

在上述研究中可以发现,利用机器群体智能产生解决方案的方法也十分重要,机器群体智能算法的选择对解决问题过程的收敛性和最终解决方案的多样性有很大影响.Souza 等人[45]认为 NRP 问题可以看作是背包问题,蚁群算法在这类问题上的收敛性和产生解决方案的质量上表现非常好,因此作者对蚁群算法进行改进,使之可以解决存在互相依赖的需求的 NRP 问题.Kumari 等人[50]结合量子计算和演化计算的特性,提出了一种量子启发的多目标演化算法来解决 NRP 问题中的需求选择问题. Chaves 等人[52]提出一个蜂群算法解决多目标下一版本问题(multi-objective next release problem, MONRP),该方法可以生成高质量的需求集供工程师进行决策.José 等人[53]改进了早期的蚁群系统用于项目迭代中需求集的选择问题,算法效果得到了很大提升.

人机结合群体智能能够使用人类智能来弥补机器智能的不足.Tonella 等人[68]提出了一种基于人机交互的遗传算法来解决需求优先级排序问题,该方法将评估解决方案的任务交给具有相关专业知识的人群,实验表明该方法的有效性和鲁棒性均优于机器群体智能方法,并且可以容忍一定的人为错误

率.Wever 等人[69]采用协同演化算法通过确定性有限状态机半自动化寻找合适的形式化软件规范,该方法需要用户对有限的训练数据进行标记,帮助机器群体智能产生高质量的解决方案.

此外 Sultanov 等人[70]的研究表明机器群体智能也适用于需求跟踪问题.Yue 等人[71]为需求审查提出了评估需求审查分配结果的适应度函数,机器群体智能也可以用于自动需求审查分配问题.

和传统需求工程方法相比,应用群体智能方法的优势包括大规模群体的问题并行处理能力、涉众群体的多样性以及对需求分析结果的优化等.但是具体到不同层次的群体智能时,可以发现机器群体智能很难进行涉众确定活动;而人类群体中个体无法处理复杂任务,因而在进行需求分析时存在任务分解难题.针对上述问题,我们可以通过不同层次群体智能的相互转化来增强群体智能方法的普适性.例如,在涉众确定活动中,机器群体智能的局限性正是人类群体智能的优势所在,将涉众群体看作群体智能涌现的源泉即可解决问题;同样,利用机器群体智能优化分解问题,人类群体智能解决子问题的人机结合群体智能模式在需求分析中能够发挥巨大作用.

3 软件设计

软件设计是从需求说明到软件具体实现的过渡阶段.人类群体智能在这方面的研究比较少,如表 3 所示,主要有软件概念建模[72]、UI 界面设计[38]和 Web 网站页面设计[73].人类群体智能方法在上述问题上的应用主要来源于 2 方面因素:1)在激励的作用下,大量工人会给出自己的软件设计方案,众多软件设计方案的竞争会带来一个当下最优的设计方案;2)通过不断学习人类群体智能形成的知识库,可以不断优化已有设计,从而获得近似最优解决方案.

Table 3 Research on Collective Intelligence Based Software Design

表 3 基于群体智能的软件设计相关研究

Method	Problem	Number of Papers	Reference
Human Collective Intelligence	Software Modeling	1	Ref[72]
	UI Design	4	Ref[38]
	Web Design	1	Ref[73]
Machine Collective Intelligence	Component Design	2	Ref[74]
	Software Architecture Design	13	Ref[75-77]
	Software Product Line Design	12	Ref[78-83]
	Service Design	8	Ref[84-85]
Human-machine Collective Intelligence	Class Design	2	Ref[86]

与之相比,利用机器群体智能解决软件设计问题目前已有大量的应用,相关研究领域包括软件体系结构设计[75-77]、软件产品线设计[78-83]和面向服务的软件设计[84-85]等.Räihä[87]从更广泛的角度总结了机器群智优化算法在软件设计中的应用,不仅包括软件开发前期的软件设计,还包括软件维护和重构工程中的软件设计.机器群体智能能够在上述软件设计方法中发挥作用,主要是因为复用和模块化已成为现代软件系统设计的基本准则,因此软件设计问题通常能转化为在成本、资源和性能约束下选择并组合已有软件组件的问题.由于软件组件的多样性和庞大的数量,机器群体智能应用于软件设计的关键在于如何选择并组合已有软件组件,优化设计方案.例如 Lagerstrom 等人[74]根据非功能性需求、效用理论和涉众需求设计适应度函数,从组件库中选取新的组件对 UML 进行修改,搜索最佳解决方案.

3.1 软件体系结构设计

在软件体系结构设计上,人们通常关注的是软件系统组织方式和整体结构,其目的是确定一个系统的主要构件以及构件之间的相互关系.机器群体智能在这一问题上已有很多应用,例如 Andrade 等人[76]提出了 DuSE 方法并开发了对应的 DuSE-MT[75]工具,该工具能够给出设计空间系统化表示元模型,在运行过程中不断利用元模型对已有架构进行自动修改,生成新的体系架构,并利用帕累托最优理论评估系统体系结构,为决策提供价值权衡.Li 等人[77]则提出了 PETUT-MOO 工具,在给定软件体系结构设计模型时通过模型转换自动优化和改进

该设计方案的非功能性需求.该工具利用集成环境来实时分析架构模型性能,从而利用处理器利用率、成本和系统延迟等软件质量因素评估新的软件架构方案.

3.2 软件产品线架构设计

软件产品线架构(product line architecture, PLA)指的是一个共性的体系结构以及共享构件的一组应用,其中每个应用都可以通过特征化来满足特定用户需求.Harman 等人[88]系统总结了应用机器群智优化算法解决软件产品线工程中问题的研究,为软件产品线的自动演化提供了方向.在软件产品线设计问题上,如何利用机器群体智能优化软件设计是问题的关键.现有优化搜索算子改变 PLA 的结构,可能违反分层体系结构,使得 PLA 灵活性和可维护性降低.为此,Mariani 等人[79]引入考虑分层体系结构样式的搜索算子,可以保证 PLA 风格不变并且与之前的方法相比有更好或等价的适应度函数用于软件设计方案的评估过程.同时,PLA 是根据软件特征进行设计的,将特征模块化可以保证 PLA 的扩展,Colanzi 等人[81]提出基于特征的算子用于 PLA 设计方案的优化,可以解决此类问题.Le 等人[82]通过引入变异算子将设计模式引入 PLA 的设计优化过程中,提高了产生和优化的解决方案的质量.

为了能够直观评价机器群智方法优化 PLA 设计方案的实际效果,Wu 等人[89]提出了一个寻找所有解决方案的 3 阶段算法,可以促进决策者选择的决策.之后 Wu 等人[80]还分享了使用该算法和机器群智优化算法解决实际 PLA 设计问题的计算经验,实验证明该算法能够帮助决策者确定算法的最优参数.Colanzi 等人[83]提出了新的基于搜索 PLA 直接表示方法,比现有研究中的表示方法更有效率.Féderle 等人[78]提出一个工具 OPLA-Tool,实现机器智能在多目标优化 PLA 任务上的辅助.

3.3 面向服务软件设计

近年来,面向服务的软件体系结构也正受到人们的日益关注,其中每个服务实现一个特定的功能,并具有松耦合、可复用和技术异构等特征.在面向服务的软件设计上,人们通常面临服务组合和服务选择的问题,其中服务质量[90](quality of service, QoS)是机器群体智能评估解决方案的主要指标.例如,云服务提供商提供的云服务及其相关定价模型各不相同,导致部署云服务十分复杂.Ardagna 等人[84]提出了一种 MILP 方法,可以利用机器群体智能生成一个可行的初始解决方案用于自动搜索优化过程,该方案利用成本和 QoS 作为评估函数,实验证明该方法可以降低云应用程序的成本,并提高最终系统的质量.为了实现服务组合成所需系统的所有阶段自动化,Fanjiang 等人[85]提出了一种服务组合自动化的方法,该方法利用遗传算法生成和优化服务组合方案,利用功能性和非功能性需求相关度量对服务组合方案进行评估.

此外,Simons 等人[91]提出了一种新型的交互式蚁群算法,可用于软件开发生命周期早期的软件设计中.

3.4 其他设计问题

在人类群体智能的应用上,由于单个建模者无法建立高质量概念模型,Jiang 等人[72]设计了一个工具,可以使在线建模者突破时间空间限制进行协作,完成概念模型建模,Huang 等人[38]利用人群已有的设计方案建立了在线的程序应用线框图和 UI 设计示例之间的映射,为开发者提供参考.同样的,Lasecki 等人[92]提出了一个实时众包系统 Apparition,通过理解需求方对软件功能的描述和画出的草图,在线群体帮助给出软件的原型设计.Latoza 等人[93]研究了众包软件设计中解决方案之间的关系,其他人的解决方案可以帮助自己的解决方案演化,获得更好的解决方案,这为软件设计竞赛提供了好的改善方向.Nebeling 等人[73]提出了轻量级 Web 信息系统开发方法,网站设计的演化基于人群提供的数据和功能,网站所需组件的开发也由众包实现.

人机结合群体智能在软件设计问题上同样有一定程度的应用.例如,在面向对象编程中,Simons 等人[86]发现使用演化计算和软件代理的基于交互式搜索的人机结合群体智能方法进行类设计非常有前景.

与传统软件设计方法相比,人类群体智能方法的使用能够提供高质量、多样性的设计方案[93];机器群体智能能够有效量化软件设计中的各种约束,科学地进行体系结构、产品线和服务组合等设计问题.从表 3 中相关论文数量来看,关于如何使用人类群体智能进行软件设计和应用的研究相对较少,其原因在于软件设计任务很难分解为简单的子任务,因此无法有效降低处理问题的复杂性.

4 软件构造

软件构造是将设计模型实现为目标软件系统的过程,其中面临的主要问题是如何高效编写高质量的代码.表 4 给出了不同群体智能方法在解决软件

构造问题上的代表性工作.其中人类群体智能方法在辅助调试[41,94-96]、程序优化[97-98]和程序自动修复[99-100]中都有应用,这主要是因为软件开发者通常数量众多,能够大规模涌现群体智能.开源软件代码仓库和社区的快速发展带来大量的代码、文档和测试等编程资源和经验,机器群体智能方法复用这些知识,可以自动高效地解决很多编码问题,例如程序优化[101-106]和程序自动修复[107-112];人机结合群体智能则在程序优化[113]和程序自动修复[114-115]中具有很好效果.

Table 4 Research on Collective Intelligence Based Software Coding

表 4 基于群体智能的软件构造相关研究

Method	Problem	Number of Papers	Reference
Human Collective Intelligence	Coding & Debugging	18	Ref[41,94-96]
	Program Optimization	2	Ref[97-98]
	Program Repair	7	Ref[99-100]
Machine Collective Intelligence	Program Optimization	8	Ref[101-106]
	Program Repair	24	Ref[107-112]
Human-machine Collective Intelligence	Program Optimization	2	Ref[113]
	Program Repair	3	Ref[114-115]

4.1 编码和调试

在编码和调试问题上,人类群体智能主要体现在知识与经验的复用.将人群的编码调试经验、相似功能代码整合和代码依赖库的说明文档等知识融合,形成编码知识库.开发者在开发新的软件系统时可以利用知识库对自己的代码进行优化,修改错误的函数使用方法、优化代码结构、代码算法优化以及提高调试效率等.

例如经验不足的新手开发者在编码和错误调试上通常会浪费太多的时间,针对这一问题,Hartmann等人[41]设计了面向静态语言的 Help-MeOut 推荐系统,通过展示其他程序员是如何纠正类似错误的示例来帮助用户调试错误和异常消息.Mujumdar 等人[95]开发了一种辅助工具 Crowd Debug,能够为动态解释的 Web 开发语言提供调试帮助.此外,Fast等人[94]提供的 Codex 工具能够记录程序员是如何使用编程语言的,新手程序员可以通过 Codex 掌握经典函数调用方式,避免出现不必要的程序语句.

这种利用人类群体智能的辅助工具需要预先收集大量代码方面的知识,如何建立对应的知识库因而也是相关研究中的一个重要问题.一般来说,人们通常从编程相关的问答社区中获取信息来建立知识库,但其中出现的代码片段往往不可直接运行.基于此,Terragni 等人[116]致力于将问答社区中代码片段自动合成可以直接编译的程序;Chen 等人[96]从 Stack Overflow 数据中建立代码匹配数据库,可以帮助开发者发现有缺陷的代码,并且提出修复缺陷的解决方案.

4.2 程序优化

利用群体智能对程序进行优化的研究主要关注编译优化.在人类群体智能层次上,Fast 等人[98]设计了一个 Python 语言扩展 Meta,可以为开发者寻找代码优化和错误修复方案.Auler 等人[97]通过收集 Web 客户端的性能数据,为 JavaScript 建立了在线编译标记参数推荐系统,可以对给定的平台达到5倍的性能优化.

在人机结合群体智能层次上,Cochran 等人[113]提出了 Program boosting 方法来解决专业人士都难以解决的编程问题.例如在书写正则表达式时,通过熟练的开发人群提供问题的初始方案,通过演化来生成改进的解决方案,在这期间,未经训练的人群评估程序输出来帮助产生改进的方案,实验证明这样使得正则表达式精度得到了提升.

除了人类群体智能利用编码经验知识辅助程序优化的方式以外,利用搜索算法创建程序变体,生成自动修复方案,在多个版本中选取最优解决方案的机器群体智能同样也被成功应用于程序优化.遗传优化(genetic improvement, GI)是其中的代表性自动修复方案生成和优化方法,其能通过自动搜索生成新的程序代码来改进软件的现有版本.在程序优化任务中,有非常多的优化评估函数,例如软件执行效率[101-106]、能耗[117-119]和内存消耗[120]等非功能性指标都可以用于评估机器群体智能方法生成的优化结果.

4.3 程序自动修复

程序修复包括故障定位、补丁生成和补丁验证

3 个阶段.程序自动修复的评估准则是通过测试用例的数量,修复过的代码通过的测试用例越多,说明修复效果越好.为了利用人类群体智慧进行故障定位,Adriano 等人[99]讨论了将复杂系统故障定位分解成可以众包的微小任务的可能性,而 Fryer 等人[100]则对使用奖励机制鼓励有能力的人发现软件漏洞的现象进行了系统的研究.

与上述研究相比,机器群体智能在程序自动修复中的应用更为广泛,相关方法主要利用搜索算法对故障代码进行修改,希望能够在不改变代码功能的情况下自动修复故障.例如 Nguyen 等人[109]提出了一种利用遗传编程(genetic programming, GP)自动进行软件修复的方法.通过将修改集中在与 bug 发生位置相关的区域,该方法有效地将搜索空间复杂性最小化,能在短时间内生成大量的高质量修复方案.Weimer 等人[107]提出了一种用于故障定位和修复缺陷的全自动方法 GenProg.一旦发现程序错误,GenProg 就通过使用一种扩展形式的遗传编程演化程序变体来生成修复方案,直到既保留了所需的功能又修复程序中的缺陷为止.Le 等人[108]进一步通过实验验证了该方法的有效性.

除了高级程序语言外,程序修复还可以在汇编语言层次进行.面向汇编语言的修复不需要访问源代码,具有很强的普适性,而且所有修复都可以在更细粒度的级别上进行.例如 Schulte 等人[110]描述了一种利用进化计算在汇编代码级自动修复遗留软件的方法,在 Java 字节码和 x86 汇编程序上的实验展示了如何获得在保留所需程序特征的同时纠正缺陷的程序变体.

此外,为了提高机器群体智能方法在程序修复应用中的有效性,需要提高机器群体智能评估修复结果的能力,Fast 等人[111]研究了适应度函数的设计问题,其提出的适应度距离相关度量方法有助于提高算法的效率和准确性.Qi 等人[112]则从测试用例优化角度进行了研究,将回归测试优先级分析技术引入自动化程序修复领域,并提出了一种新的优先级分析技术 FRTP,能够减少修复过程中所需执行测试用例的数量,大大提高自动修复的效率.

在程序自动修复领域中,人机结合群体智能中人类群体智能的作用往往体现在为机器群体智能提供知识库,增强机器群体智能的效果.机器群智方法应用于故障修复时,解决方案的搜索空间非常巨大,处理问题时复杂性非常高.例如虽然 GenProg 方法在应用时能显示出非常好的效果,但是由于突变操作的随机性,GenProg 会生成无意义的补丁.Kim 等人[115]利用人机结合群体智能,提出了基于模式的自动程序修复技术,将现有的人工编写的补丁作为知识库,通过学习总结出常见的修复模式为自动生成补丁服务.该方法生成的补丁比 GenProg 生成的补丁更容易被接受,且实验中的生成的补丁成功修复错误的概率更高.Nguyen 等人[114]在使用 GI 对程序进行修复时也参考了人工开发的补丁来提高程序自动修复的效果.

目前自动化软件构造方法以形式化方法和模型驱动软件开发方法为主,机器群体智能方法尚无法实现完全自动化的软件构造,因此软件构造阶段的相关文献数量最少,研究集中在程序优化和修复上[94-96].与之相比,人类群体智能基本上是以经验知识库的形式出现,在软件构造中发挥了辅助作用.机器群体智能反映了对自动化软件构造目标的不懈追求,将人力从繁杂的编码任务解放出来,但是机器群体智能在程序优化和自动修复上的效果仍有待提高,搜索复杂性和解决方案本身距离满足实用的要求也存在一定差距.

5 软件测试和验证

基于群体智能的软件测试是基于群体智能的软件工程研究的一个重要分支.软件测试是搜索软件中存在潜在错误的过程,这非常契合机器群体智能的使用方式,其中基于搜索的软件测试便是这一方向的代表,也是 SBSE 中最为热门的研究领域[121].同时,让终端用户参与到软件测试过程中也是人类群体智能研究的关注点,目前已提出了很多基于众包的软件测试方法[122].

群体智能在软件测试中的应用包括软件测试和软件验证,如表 5 所示.在软件测试中,人类群体智能研究包括测试用例生成[123-125]、Oracle 问题[126]、GUI 测试[127-128]、性能测试[129]和可用性测试[130]等;机器群体智能方法应用于测试用例生成[27,37,131-135]、测试用例排序[136]、Oracle 问题[137]、GUI 测试[138-141]、性能测试[142]、功能测试[143-144]和回归测试[145]等和软件模型检测[146-149];人机结合群体智能在测试用例生成[150]和 Oracle 问题[150-151]中效果显著.在软件验证中,人类群体智能在错误定位等方面有一些应用[152-153].

Table 5 Research on Collective Intelligence Based Software Testing

表 5 基于群体智能的软件测试相关研究

Method	Problem	Number of Papers	Reference
Human Collective Intelligence	Test Case Generation	2	Ref[123,125]
	Software Verification	4	Ref[152-153]
	Oracle Problem	1	Ref[126]
	GUI Testing	2	Ref[127-128]
	Performance Testing	1	Ref[129]
	Usability Testing	7	Ref[130]
Machine Collective Intelligence	Test Case Generation	117	Ref[27,37,131-135]
	Test Case Prioritization	12	Ref[136]
	Model Checking	10	Ref[146-149]
	Oracle Problem	10	Ref[137]
	GUI Testing	5	Ref[138-141]
	Performance Testing	5	Ref[142]
	Function Testing	7	Ref[143-144]
	Regression Testing	9	Ref[145]
Human-machine Collective Intelligence	Test Case Generation	4	Ref[150]
	Oracle Problem	2	Ref[150-151]

5.1 测试数据生成

机器群体智能在软件工程领域应用的最早尝试就是解决软件测试数据生成问题.Miller 等人[27]使用遗传算法来自动生成浮点数测试数据,首次将测试数据生成问题转变为基于搜索的优化问题.随后,相关研究的发展十分迅速,基于搜索的软件工程领域中超过一半的研究都关注在软件测试[27],其中包括遗传算法、蚁群算法等机器群体智能方法已被广泛应用于测试用例的生成[37,131-135]、优先级排序[136]等各种问题.

在回归测试中,时间往往不足以执行所有的测试用例,需要对测试用例的优先级进行排序.Singh 等人[136]实现了回归测试排序技术的算法形式,使用 ACO 优化了排序结果.在路径测试中,Li 等人[133]使用 ACO 自动化测试用例生成,使用代码覆盖率作为评估准则.类似地,Srivastava 等人[134]也使用 ACO 自动生成测试用例来达到更好的代码覆盖率.在特殊情况下,对于某些容易出错的程序路径需要重点测试,Rao 等人[135]提出了一种适用于测试比较容易出错的路径的方法,该方法提高了测试性能.Mahajan 等人[131]曾提出一个模型,可以用来检验遗传算法被用于使用数据流测试技术自动创建测试套件的性能,帮助评估测试用例生成的效果.

利用机器群体智能生成测试数据就是不断搜索被测函数的输入空间,生成新的测试数据,不断迭代直到测试数据集满足给定覆盖率等测试要求的过程.上述使用基于机器群体智能的测试用例生成方法能节省大量的人力资源,代码覆盖率也很高,但是在实现更高层次的代码覆盖上仍存在一些困难[154].

除了机器群体智能外,人类群体智能同样也被应用于测试数据生成.其目的通常是利用人类智力的主观性有意识地生成测试数据,有助于减少重复或无用的测试用例数量,同时可以提高代码覆盖率.例如 Chen 等人[125]提出基于谜题的半自动测试环境,它将对象变异和复杂约束解决问题分解为小难题,采用游戏性的测试流程吸引人群,从而利用人群涌现的群体智能生成较优的测试用例,该方法较当前一些自动测试用例生成技术提高了代码覆盖率.Pham 等人[123]对 Github 开展了一项研究,GitHub 社区中有能力的人可以很轻松地解决别人代码库中出现的问题,从这个现象中发现了从社会性代码社区中生成测试用例的可能性[124],但是这个想法还有待完善.

研究者也尝试使用人机结合群体智能来产生测试数据,尤其是以人类知识库的方式为机器群体智能的应用提供支持,希望可以提高自动化测试用例生成的效率和生成的测试用例质量.例如 McMinn 等人[150]曾提出一种从开发者、源代码和文档抽取

知识的自动测试数据生成方法,从而通过利用这些知识来生成更加贴近真实应用场景、和更加容易判断测试结果的测试数据.

5.2 Oracle 问题

除了测试用例生成和排序外,群体智能方法在其他测试问题上也得到了广泛的应用.在测试用例的预期输出问题中,预期输出 Oracle 的判定通常需要人工来完成.Pastore 等人[126]曾开展一项实验来让人群判断程序断言和代码文档描述的行为是否一致,他们发现人类群体智能对解决预期输出问题有一定的帮助,但如何有效组织人类群体仍旧面临巨大的挑战.

目前,尽管人工判定 Oracle 的成本很高,但很少有研究来提高这项工作的效率.人工判定 Oracle 成本的一个来源是机器生成的测试输入通常具有不可读性,例如自动生成的字符串输入往往是难以读取的任意字符序列.Afshan 等人[151]提出了一种将自然语言模型引入基于搜索的输入数据生成过程中,以提高生成字符串的可读性的方法.实验结果表明参与者在判定使用语言模型生成输入的 Oracle 时,速度显著加快,部分实验在准确性上也有显著提高.

Just 等人[137]发现 Oracle 问题很难利用机器群体智能自动化解决,而利用底层函数或算法的约束来识别测试系统中的故障是比较有前景的判定 Oracle 的方法.

5.3 群体智能在不同待测软件中的应用

群体智能在针对不同开发方式和应用场景的软件测试方法中也有着广泛的应用,其中一个代表性例子是基于群体智能的图形用户界面(GUI)测试.GUI 是如今软件的重要组成部分,它的正确执行是保证整个软件正确性的必要条件.但是在机器群体智能的应用中,创建一个可用于自动化测试用例生成的模型是困难的[155],很少有工具能够帮助自动化测试过程.大部分研究只针对 GUI 测试的一个小阶段进行优化.

在 GUI 测试上,目前工业界最常用的技术仍然是捕获和重放工具,它简化了输入序列的记录和执行,但是不支持测试人员寻找故障敏感的测试用例.Bauersfeld 等人[138-139]针对 GUI 测试序列生成问题,使用蚁群优化算法来生成对故障敏感的测试用例.此外,在进行 GUI 测试时常常会因为只研究用户界面而遗漏许多底层程序行为.Gross 等人[140]开发出 EXSYST 原型工具,使用遗传算法来生成测试用例,优化 GUI 测试套件,以实现在探索用户界面的同时保证最大可能的代码覆盖率.

上述研究表明群智优化算法可以为 GUI 生成复杂的测试序列,但是测试序列有很强的随机性.对于希望用户在 GUI 中输入特定输入值的应用程序,测试效率会非常低.Salvesen 等人[141]利用动态符号执行生成特定输入值,实验中使用 DSE 对基于搜索的测试生成工具 EXSYST 进行拓展,成功增加了 2 个案例程序的测试代码覆盖率.

GUI 测试难以自动化,而人工测试 GUI 非常费时、昂贵且难以集成到连续的测试过程中.Dolstra 等人[127]展示了使用人类群体智能解决 GUI 测试的可能.实验证明,通过实例化测试系统的虚拟机,让测试人员通过互联网访问测试系统可以让大量参与者参与到测试过程中,从而在软件系统的连续测试中实现 GUI 的半自动连续测试,并且成本很低.Vliegendhart 等人[128]在 Amazon 的众包平台 Mechanical Turk 上对一个多媒体应用程序进行 A/B 测试的例子同样证明了人类群体智能在 GUI 测试上的巨大优势.

5.4 群体智能在软件特性测试中的应用

除了针对不同的待测软件外,群体智能方法还被应用于软件不同质量属性的测试.其中,性能测试是群体智能广泛应用的一个领域,其目标通常是发现影响系统服务质量 QoS 的因素,如响应时间、执行效率和可靠性等.

在机器群体智能应用中,Gu 等人[142]针对组合服务软件系统提出一种基于遗传算法的测试用例自动生成方法,该方法根据系统的工作流程,将系统使用模式建模为基于性能需求的 QoS 敏感因子,采用群智优化算法自动寻找具有最大或最小 QoS 的测试用例.实验证明该自动测试数据生成方法生成的测试数据优于随机测试.

然而,对系统建模仿真测试或者在受控环境中进行性能测试很难准确评估在极端异构环境中软件运行的性能.利用人类群体智能,将用户的真实使用性能数据收集起来用于性能测试评估能够帮助开发者团队识别和确定性能问题的优先级.Musson 等人[129]在 Lync① 平台上的实验证明了该方法的有效性.

① https://products.office.com/en-us/previous-versions/microsoft-lync-2013

并不是所有软件测试问题都可以用机器群体智能解决.例如可用性测试是检查软件程序的人为因素和易用性等问题,人类群体智能基于人群的特点在这个领域有一些研究[130],但是机器群体智能很难在这个领域发挥作用.总的来说,群体智能被广泛应用于解决各种软件测试问题,除上面提到的研究以外,功能测试[143-144]、回归测试[145]等不同测试方法中遇到的问题都可以借助群体智能解决.

5.5 软件验证

软件测试的目的是发现程序中的错误,而软件验证则更加关注软件是否满足其所声明的功能性和非功能性需求.目前,机器群体智能在软件验证中的应用主要集中于模型检测.例如 Alba 等人[146]将蚁群算法应用于模型检测,在状态空间中寻找反例.Banerjee 等人[147]利用蚁群算法和粒子群算法来优化模型检测.Katz 等人[148-149]使用基于模型检查的 GP 在程序规格说明中验证和合成代码,其中模型检查用于提供评估每个阶段突变结果的适应度函数值.

软件验证一般由经过专业训练的工程师进行,如何让未经训练的普通人也能够进行软件验证工作是人类群体智能相关研究的重点.例如 Dietl 等人[152]将验证任务转化成为小游戏,普通人熟悉游戏玩法便可以通过游戏证明软件的正确性.Li 等人[153]建立了一个系统 CrowdMine,通过模拟实验证明未经训练的人群确实在一些软件验证工作如错误定位等有帮助作用.

从表 5 相关文献数量可以看出,人类群体智能和机器群体智能在软件测试与验证阶段都能发挥巨大作用,利用群体智能可以提高测试效率,降低成本.但是不同层次群体智能方法存在各自的优缺点,需要根据测试的实际要求进行选择.例如机器群体智能是一种广泛适用的、有效的生成测试数据和优化测试过程的方法,能够解决许多人类群体智能不能解决的软件测试问题[156];而人类群体智能利用大量在线工人参与完成测试任务,对真实应用场景和真实用户表现的良好模拟是机器群体智能无法提供的[122].整体上来说,相比于传统测试方法,群智方法更加高效、成本低廉,易于集成到持续集成过程中.

6 软件维护

软件维护阶段的主要任务包括软件演化以及软件文档和相关制品的更新.如表 6 所示,人类群体智能主要应用于软件演化[157-159],机器群体智能在软件演化[160]和软件重构[161-163]及软件模块化[164-166]中都有研究,人机结合群体智能出现在软件重构[167]中.其中,Mariani 等人[168]曾总结了应用于确定软件构件的最佳重构序列的基于搜索方法的共同特点,并讨论了未来的研究机会.

Table 6 Research on Collective Intelligence Based Software Maintenance

表 6 基于群体智能的软件维护相关研究

Method	Problem	Number of Papers	Reference
Human Collective Intelligence	Software Evolution	12	Ref[157-159]
	Document Maintenance	5	Ref[169]
	Software Localization	4	Ref[170]
Machine Collective Intelligence	Software Evolution	16	Ref[160]
	Software Version Refactoring	11	Ref[161]
	Optimize Refactoring Sequence	6	Ref[162-163]
	Software Modularization	12	Ref[164-166]
	System Migration	5	Ref[171]
Human-machine Collective Intelligence	Software Version Refactoring	1	Ref[167]

6.1 软件演化

软件的运行环境往往会随着系统使用时间的增长而发生变化,对应地,软件也需要不断进行演化和更新.在软件开发中,软件的上下文环境是很难监视的,Ali 等人[157]提出 Social sensing 以利用终端用户监视软件上下文环境的变化,不断发掘用户需求,从而进行软件演化.He 等人[158]提出一个模型,鼓励用户参与到软件运行适应的流程中,用户可以提出和修改他们的意见,应用程序借鉴这些建议可以使自身运行时更好地适应环境.例如 Nebeling 等人[159]采用众包的方式,让用户参与网页接口自适应过程中,从而让网页可以适应广泛的使用环境.

人类群体智能通过引入人群作为软件系统的评估对象对软件进行演化和优化,相比于上述人类群体智能的应用,机器群体智能在软件演化上的研究较少.Schulte 等人[160]利用程序的"中性网络"和演化算法可以对软件进行自动演化,这项研究将对现实软件系统的开发人员和维护人员有巨大帮助.

6.2 软件重构和模块化

软件重构是提升程序性能以减缓其由于更改而性能退化的过程.重构意味着通过修改程序来改进程序结构,让程序更容易理解.SBSE 在软件维护阶段的研究重点就就是软件重构问题,相关研究主要关注 2 个方面[10]:1)利用群体智能方法将软件优化,形成重构后的版本[161,167];2)优化软件的重构步骤,使重构获得最好的效果[162-163].无论是软件优化过程还是重构步骤的选择,在机器群体智能应用中都可以转化为搜索问题,进而探索最优的解决方案.在软件重构中,机器群体智能应用的问题在于如何评估方案的好坏.

例如软件系统反复修改会导致代码坏味[172]的出现.重构可以消除这些坏味,可以将代码坏味作为评估重构方案的指标,但是手工确定和执行有用的重构是一项艰巨的挑战.Kebir 等人[162]提出一种基于遗传算法的基于构件软件自动重构方法.该方法包括利用遗传算法生成优化重构的最佳序列,检测并消除与构件相关的代码坏味.Mkaouer 等人[163]进一步引入了一个基于 NSGA Ⅱ的多目标健壮模型,该模型将软件质量改进、代码坏味严重性和代码的重要性作为评估的适应度函数,通过对 3 个因素的权衡以获取重构效果最大化.

人机结合群体智能同样被应用于解决软件重构问题.例如 Ghannem 等人[167]提出了一种交互式遗传算法,该算法利用人的智力对产生的重构建议进行评估,从而提高重构活动的自动化程度.

软件模块化是降低软件复杂性的重要方法,有助于软件实现可控、可维护和可扩展.Mancoridis 等人[164]最早将机器群体智能算法用于软件的自动模块化.该方法利用了传统的爬山算法和遗传算法,取得了不错的效果.随后 Mancoridis 等人[165]还开发了一个聚类工具 Bunch,实现了软件模块聚类.在 Mancoridis 等人研究影响下,其他一些研究者也将软件模块聚类看作可以利用 SBSE 方法解决的研究问题.例如 Harman 等人[166]研究了将模块粒度、内聚性和耦合性度量作为评估的适应度函数的影响.

6.3 其他软件维护工作

除上述软件维护问题以外,群体智能还被应用在其他维护活动中.例如在把遗留系统迁移到面向对象技术体系中的过程中,在代码中识别对象是整个过程的关键任务,Sahraoui 等人[171]利用遗传算法在过程代码中识别对象解决了这一问题.

而在人类群体智能的应用上,众包还被用来解决软件文档维护问题[169]和软件本地化问题[170].例如 Jiau 等人[169]提出了一种众包软件文档重用的方法,能够有效解决软件相关文档不足的问题.研究团队在 Stackoverflow 平台上的研究证明了文档重用的可行性,同时实验还表明文档重用能够有效提高文档的覆盖率和质量.

软件本地化是指将软件产品的用户界面和辅助文档从其原产国语言向另一种语言转化,使之适应某一外国语言和文化的过程.Exton 等人[170]发现众包是解决本地化的可行方法,其使得软件本地化的决策从大型企业转移到服务用户,从而使软件可以有更多的语言可用,因此提出并描述了一种新的众包模式,它可以为实现突破语言障碍的"信息与知识的平等获取"提供一个平台.在这个平台上,软件本地化的决策由用户执行,用户个体出于自身利益不断优化本地化方案,并接受其他人评估和优化,最终得到的语言解决方案应该是大家都认可,最优的结果.

在软件维护阶段中,以用户群体作为人类群体智能涌现主体可以在软件文档编写、软件本地化等任务发挥优势,给出更符合用户群体需求的解决方案;而软件重构问题,如重构序列确定问题对于非专业人群是十分困难的任务,但是非常符合搜索算法的优化思想,因而更加适合使用机器群体智能解决[161].从表 6 相关论文数量可以看出,软件维护是软件生命周期的最后一个阶段,常常被开发者忽视,但是其实是一个非常重要且成本很大的开发阶段,群体智能有利于软件维护工作自动化和简单化发展.

7 基于群体智能的软件工程的其他应用

群体智能在软件工程中的应用十分广泛,除了在软件开发生命周期 5 个阶段中有丰富的研究,在软件工程的其他方面也有很多的应用.如表 7 所示,人类群体智能还被应用于软件安全[17,173]、软件用户

支持[174-176]和软件创意竞赛服务[177]等研究,机器群体智能则在软件项目管理上拥有巨大优势[178-179].

为了利用人类群体智能保证软件安全,Arellano等人[180]开发了一个基于众包的Web扩展,不仅将用户看作程序的使用者,更是程序想法设计的贡献者,以保证Web应用程序的安全性和完整性.类似地,Sharifi等人[173]实现了一个名为SmartNotes的系统,利用人群众包检测潜在的Web浏览安全威胁.

而在软件用户支持方面,Chilana等人[174-175]发现即使在以用户为中心的设计和可用性方面做出了最大的努力,也不是所有的用例中的细微差别都能在设计时预料到,用户在寻找软件相关帮助也十分困难.基于此,他们设计了LemonAid方法,允许用户从其他用户那里找到问题的答案,让软件支持能够被重用.Chilana等人[176]之后还对LemonAid进行实例研究,验证了LenmonAid的价值和有效性.

开展创意竞赛有益于众包的推广和软件开放创新过程.Leimeister等人[177]发现许多基于信息技术的创意竞赛不能吸引用户积极参与,相关发现强调了主办方的荣誉、奖励和专业知识作为激励的重要性.

Table 7 Research on Other Collective Intelligence Based Software Activities

表7 其他基于群体智能的软件工程活动相关研究

Methods	Problem	Number of Papers	Reference
Human Collective Intelligence	Software Security	9	Ref[173,180]
	User Support	3	Ref[174-176]
	Creative Competitions	4	Ref[177]
Machine Collective Intelligence	Software Project Management	43	Ref[178-179]

软件工程项目管理关注的是软件生命周期不同阶段执行的复杂活动管理,目标是优化软件生产过程以及这个过程产生的产品.机器群体智能对于这类多目标约束优化问题有很强的处理能力,不同的解决方案生成算法和不同的评估适应度函数是研究的重点,其中Chang等人[178]最先将群智优化算法应用在软件项目管理中,而Kapur等人[179]使用遗传算法在有限的时间限制下为客户提供最优的产品发布人员和最优的服务质量.

8 基于群体智能的软件工程挑战及未来方向

8.1 机器群体智能

以基于搜索软件工程为代表的机器群体智能在软件工程中的应用已经是一个相对成熟的研究领域了.但是,在已有的研究领域中还有许多问题无法使用机器群体智能来解决,包括在需求工程中如何寻找尽可能多且具有代表性的用户群体、软件测试中涉及人因工程的可用性测试和Oracle问题等.此外,各种新的软件工程形式,如面向服务软件工程[2]、云平台软件工程[181]、大数据软件工程[182]的出现也向SBSE的广泛应用带来了新的挑战.

在研究新的软件工程问题外,机器群体智能的发展还需要新方法和新技术的支持.机器学习能够在一定程度上弥补机器群体智能的不足,增强机器群体智能算法的性能,主要表现在其能够代替人工,加深机器群体智能方法的自动化程度,同时还能优化群体智能算法性能.然而,这个交叉领域尚未成熟,许多问题还没有现成可用的机器学习解决方案,使用机器学习解决问题的质量也不能得到保证.

对于特定问题,机器群体智能使用者一直在寻找性能良好的群体优化算法[183],然而这是一个非常困难的过程.同时,机器群体智能通常依赖定制化的搜索算法来确保在具体问题上取得理想的效果[184].未来机器群体智能应用的一种新方式应该是在广泛的问题实例中学习搜索策略,提供一种类似超启发式搜索的单一的泛型方法[168].另一方面,通过机器学习的数据分析预测能力能够为机器群体智能的应用提供新思路,提高机器群体智能算法的效率.例如,在软件测试中,根据已有测试结果预测哪些剩余的测试用例会失败,哪些测试应该有更高的优先级,通过预测能力加快软件测试的速度.

如今软件工程以及机器学习都和大数据技术紧密相连,大数据软件工程因而也为机器群体智能的应用提供了借鉴.在此基础上,基于机器群体智能的软件工程范式应该转变为数据驱动的软件工程方法,即首先通过大数据技术从已有软件系统、开源软件和社区中采集海量的数据并组织形成信息,随后通过对相关信息的整合和提炼,以机器群体智能为基础形成自动化的软件工程问题解决模型.

8.2 人类群体智能

人类群体智能通常从大规模人群的竞争与合作中涌现,其有效性高度依赖于人类群体的组织管理模式、人类群体智能的激励机制以及质量保证3个方面的因素.

当前人类群体智能方法还停留在组织管理单个人类群体解决实际问题的阶段,未来人类群体智能应用的新方式应该是将项目分解为小任务,交给多个人群完成,利用不同人群的特点解决不同的任务,从而提高群体智能的利用效率.例如在基于多人类群智的软件开发前期,需求捕获和原型开发可以同时进行,一个人群用于收集需求,另一个人群为这些需求开发原型,人群中个体的特征取决于这个人群需要处理的问题.

人类智能方法应用的困难在于如何有效促进人群中个体之间的协同作用[32],未来人类群体智能领域需要加强对人类群体智能的形成机理和人群协作规律的探索,为高效组织管理人类群体和促进协同作用提供坚实的理论基础.另外,当前互联网是人类群体智能涌现的主要平台,需要解决人类群体智能方法在互联网平台上应用的相关问题,例如人群组织管理相关的核心服务如何整合到互联网平台中.人类群体智能本身十分抽象,在使用时如何对群体协作程度进行度量与调控,保证群体智能的工作效率需要更多的研究.

人类群体智能的激励机制是群体智能涌现的动力,对于激发参与者积极性和确保可靠的交付平台至关重要[185].Groen等人[186]在基于人群的需求工程中将参与者的自身动力分为7种,但是大部分已有人类群体智能应用激励单一,因而仅能吸引小部分人群;或者激励机制不合理,打击大部分参与者的积极性.最好的方式就是针对不同类型的个体给予相应的激励机制,但是这种方式在实现中十分的不切实际,还有很多问题需要解决.

使用人类群体智能解决问题的质量保证是最重要的问题.参与者短期参与项目,参与者的个人背景和工作能力各不相同,依赖上述人群完成的任务往往容易产生质量问题[187].众包软件开发中存在23个和质量有关的因素[188],质量保证问题非常复杂.许多质量相关工作有待评估,软件质量保证将会是人类群体智能方法未来的关注重点.

8.3 人机结合群体智能

人机结合群体智能结合了机器的高度并行性和人类智力的优点,是一种发挥基于群体智能的软件工程优点的有效方法.其中,人机之间的相互关系和组织形式是人机结合群体智能应用所面临的关键问题,即如何针对人类和机器的特点设计并分配最为合适的任务.

现有的人机结合群体智能应用的出发点在于机器群体智能在某些问题上存在局限性,需要人类智力的帮助才能解决问题.例如在Cochran等人[113]提出的Program boosting方法中,通过专业人群生成群智优化算法的初始解,让未经训练的人群评估解的质量是人机结合群体智能的良好尝试.但是,这种形式的人机结合群体智能仅将人类智能作为机器群体智能的补充,如何利用机器学习方法来进一步增强人机结合的能力,以及如何将半自动化人机结合群体智能转化为全自动化人机结合群体智能仍需要进一步研究.

此外,人机群体智能需要的人的知识可以通过建立知识库来实现基于群体智能的软件工程方法的自动化.人群知识库对绝大部分软件工程问题都应该有借鉴意义,因此在建立普遍适用的人群知识库时,知识的表示和共享要能满足复用知识库的要求,并遵循一定的规范.在此基础上,复用人群知识库能缩减软件工程问题求解所需时间,提高软件生产力.不同知识库的相互补充还能使得全自动群体智能的性能不断提升,提高解决问题的性能.

8.4 基于群体智能的软件工程的应用和推广

基于群体智能的软件工程为解决复杂软件工程问题提供了新的思路,已成为现代软件工程方法的重要组成部分.为了进一步推广该方法,还有必要开展更多的经验研究,并与传统软件工程方法进行系统对比.同时,还需要针对云平台系统、移动应用、智能软件系统等新兴软件系统的特点开展基于群体智能的软件工程的实证研究和示范性应用.此外,需要开发全面的基于群体智能的软件工程工具集,为该领域的推广奠定基础.

9 总 结

基于群体智能的软件工程是当前软件工程领域研究的前沿和热点,具有良好的发展和应用前景.本文首先提出了涵盖机器群体智能、人类群体智能以及人机结合群体智能的基于群体智能的软件工程方法统一框架,随后从软件生命周期中的需求分析、设计、构造、测试和维护5个方面总结了上述3个层次群体智能方法在软件工程领域的应用,最后讨论了不同层次群体智能未来的发展方向和可能的挑战.

参 考 文 献

[1] Jacobson I. Object-Oriented Software Engineering: A Use Case Driven Approach [M]. Chennai, India: Person Education, 1993

[2] Zoran S, Dahanayake A. Service-Oriented Software System Engineering: Challenges and Practices [M]. Hershey: Igi Global, 2005

[3] Lü Jian, Tao Xianping, Ma Xiaoxing, et al. Research on agent-based network architecture software model [J]. China Science E: Information Sciences, 2005, 35(12): 1233-1253 (in Chinese)
(吕建, 陶先平, 马晓星, 等. 基于 Agent 的网构软件模型研究[J]. 中国科学 E 辑: 信息科学, 2005, 35(12): 1233-1253)

[4] Hasselbring W, Reussner R. Toward trustworthy software system [J]. Computer, 2006, 39(4): 91-92

[5] Madhavji N H, Miranskyy A, Kontogiannis K. Big picture of big data software engineering: With example research challenges [C] //Proc of the 2015 Int Workshop on Big Data Software Engineering. Piscataway, NJ: IEEE, 2015: 11-14

[6] He Xiaoxian, Zhu Yunlong, Wang Mei. Overview of knowledge emergence and complex adaptability in swarm intelligence [J]. Information and Control, 2005, 34(5):560-566 (in Chinese)
(何小贤, 朱云龙, 王玫. 群体智能中的知识涌现与复杂适应性问题综述研究[J]. 信息与控制, 2005, 34(5): 560-566)

[7] Koza J R, Koza J R. Genetic Programming: On the Programming of Computers by Means of Natural Selection [M]. Cambridge, MA: MIT, 1992

[8] Harman M, Jones B F. Search-based software engineering [J]. Information and Software Technology, 2001, 43(14): 833-839

[9] Howe J. Crowdsourcing: A definition [OL]. [2019-01-28]. http://crowdsourcing.typ-epad.com/cs/2006/06/crowdsourcing_a.html

[10] Harman M, Mansouri S A, Zhang Yuanyuan. Search-based software engineering: Trends, techniques and applications [J]. ACM Computing Surveys, 2012, 45(1): Article No.11

[11] Mao Ke, Capra L, Harman M, et al. A survey of the use of crowdsourcing in software engineering [J]. Journal of Systems and Software, 2017, 126: 57-84

[12] Heylighen F. Collective intelligence and its implementation on the Web: Algorithms to develop a collective mental map [J]. Computational & Mathematical Organization Theory, 1999, 5(3): 253-280

[13] Krause S, James R, Faria J J, et al. Swarm intelligence in humans: Diversity can trump ability [J]. Animal Behaviour, 2011, 81(5): 941-948

[14] Kaplan C A. Collective intelligence: A new approach to stock price forecasting [C] //Proc of the 2001 IEEE Int Conf on Systems, Man and Cybernetics. Piscataway, NJ: IEEE, 2001, 2893-2898

[15] Krause J, Ruxton G D, Krause S. Swarm intelligence in animals and humans [J]. Trends in Ecology & Evolution, 2010, 25(1): 28-34

[16] Winston M L. Bee work: The wisdom of the hive [J]. Science, 1996, 272(5264): 967-967

[17] Schutter G D, Theraulaz G, Deneubourg J L. Animal - robots collective intelligence [J]. Annals of Mathematics and Artificial Intelligence, 2001, 31(1): 223-238

[18] Colorni A, Dorigo M, Maniezzo V. Distributed optimization by ant colonies [C] //Proc of the 1992 European Conf on Artificial Life. Cambridge, MA: MIT Press, 1992: 134-142

[19] Mao Chengying, Yu Xinxin, Xue Yunzhi. Algorithm design and empirical analysis for particle swarm optimization-based test data generation [J]. Journal of Computer Research and Development, 2014, 51(4): 824-837 (in Chinese)
(毛澄映, 喻新欣, 薛云志. 基于粒子群优化的测试数据生成及其实证分析[J]. 计算机研究与发展, 2014, 51(4): 824-837)

[20] Karaboga D. An idea based on honey bee swarm for numerical optimization [R/OL]. Kayseri, Turkey: Computer Engineering Department, Engineering Faculty, Erciyes University, 2005 [2019-08-01]. https://www.researchgate.net/publication/255638348_An_Idea_Based_on_Honey_Bee_Swarm_for_Numerical_Optimization_Technical_Report_-_TR06

[21] Engelbart D C. Augmenting Human Intellect: A Conceptual Framework (1962) [M]. Menlo Park, CA: Stanford Research Institute, 1997

[22] Sobel D. Longitude: The True Story of a Lone Genius Who Solved the Greatest Scientific Problem of His Time [M]. London: Macmillan, 2005

[23] Zhang Wei, Mei Hong. Software development based on collective intelligence on the Internet: Feasibility, state-of-the-practice, and challenges [J]. SCIENTIA SINICA Informationis, 2017, 47(12): 1601-1622

[24] Harinarayan V, Rajaraman A, Ranganathan A. Hybrid machine human computing arrangement: U.S. Patent 7,197,459 [P]. 2007-03-27

[25] Khatib F, Dimaio F, Cooper S, et al. Crystal structure of a monomeric retroviral protease solved by protein folding game players [J]. Nature Structural & Molecular Biology, 2011, 18(10): 1175-1177

[26] Lin Du, Robles A J, King J B, et al. Crowdsourcing natural products discovery to access uncharted dimensions of fungal metabolite diversity [J]. Angewandte Chemie Int Edition, 2015, 53(3): 804-809

[27] Miller W, Spooner D L. Automatic generation of floating-point test data [J]. IEEE Transactions on Software Engineering, 1976, 2(3): 223-226

[28] Harman M, Clark J. Metrics are fitness functions too [C] // Proc of the 2004 Int Symp on Software Metrics. Piscataway, NJ: IEEE, 2004: 58-69

[29] Sayyad A S, Ammar H. Pareto-optimal search-based software engineering (POSBSE): A literature survey [C] // Proc of the 2013 Int Workshop on Realizing Artificial Intelligence Synergies in Software Engineering. Piscataway, NJ: IEEE, 2013: 21-27

[30] Boussaïd I, Siarry P, Ahmed-Nacer M. A survey on search-based model-driven engineering [J]. Automated Software Engineering, 2017, 24(2): 233-294

[31] Howe J. The rise of crowdsourcing [J]. Wired Magazine, 2006, 14(6): 1-4

[32] Li Wei, Wu Wenjun, Wang Huaimin, et al. Crowd intelligence in AI 2.0 era [J]. Frontiers of Information Technology & Electronic Engineering, 2017, 18(1): 15-43

[33] Yuen M C, King I, Leung K S. A survey of crowdsourcing systems [C] //Proc of the 2011 IEEE Int Conf on Privacy, Security, Risk and Trust and 2011 IEEE Int Conf on Social Computing. Piscataway, NJ: IEEE, 2011: 766-773

[34] Zhao Yuxiang, Zhu Qinghua. Evaluation on crowdsourcing research: Current status and future direction [J]. Information Systems Frontiers, 2014, 16(3): 417-434

[35] Kittur A, Nickerson J V, Bernstein M, et al. The future of crowd work [C] //Proc of the 2013 Conf on Computer Supported Cooperative Work. New York: ACM, 2013: 1301-1318

[36] Chittilappilly A I, Chen Lei, Amer-Yahia S. A survey of general-purpose crowdsourcing techniques [J]. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(9): 2246-2266

[37] Srivastava P R, Ramachandran V, Kumar M, et al. Generation of test data using meta heuristic approach [C] // Proc of the TENCON 2008—2008 IEEE Region 10 Conf. Piscataway, NJ: IEEE, 2008: Article No.305

[38] Huang Yicheng, Wang Chun-I, Hsu J. Leveraging the crowd for creating wireframe-based exploration of mobile design pattern gallery [C] //Companion Publication of the 2013 Int Conf on Intelligent User Interfaces Companion. New York: ACM, 2013: 17-20

[39] Lim S L, Quercia D, Finkelstein A. StakeNet: Using social networks to analyse the stakeholders of large-scale software projects [C] //Proc of the 2010 ACM/IEEE Int Conf on Software Engineering-Volume 1. New York: ACM, 2010: 295-304

[40] Lim S L, Quercia D, Finkelstein A. StakeSource: Harnessing the power of crowdsourcing and social networks in stakeholder analysis [C] //Proc of the 2010 ACM/IEEE Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2010: 239-242

[41] Hartmann B, MacDougall D, Brandt J, et al. What would other programmers do: Suggesting solutions to error messages [C] //Proc of the 2010 SIGCHI Conf on Human Factors in Computing Systems. New York: ACM, 2010: 1019-1028

[42] Pohl K. Requirements Engineering: Fundamentals, Principles, and Techniques [M]. Berlin: Springer, 2010

[43] Finkelstein A, Harman M, Mansouri S A, et al. "Fairness analysis" in requirements assignments [C] //Proc of the 2008 IEEE Int Requirements Engineering Conf. Piscataway, NJ: IEEE, 2008: 115-124

[44] Zhang Yuanyuan, Harman M. Search based optimization of requirements interaction management [C] //Proc of the 2010 Int Symp On Search Based Software Engineering. Piscataway, NJ: IEEE, 2010: 47-56

[45] De Souza J T, Maia C L B, Do Nascimento Ferreira T, et al. An ant colony optimization approach to the software release planning with dependent requirements [G] //LNCS 6956: Proc of Int Symp on Search Based Software Engineering. Berlin: Springer, 2011: 142-157

[46] Brasil M M A, Silva T G N D, Freitas F G D, et al. A multiobjective optimization approach to the software release planning with undefined number of releases and interdependent requirements [C] //Proc of the 2011 Int Conf on Enterprise Information Systems. Berlin: Springer, 2011: 300-314

[47] Zhang Yuanyuan, Harman M, Lim S L. Empirical evaluation of search based requirements interaction management [J]. Information & Software Technology, 2013, 55(1): 126-152

[48] Sureka A. Requirements prioritization and next-release problem under non-additive value conditions [C] //Proc of the 2014 Australian Software Engineering Conf. Piscataway, NJ: IEEE, 2014: 120-123

[49] Zhang Yuanyuan, Harman M, Finkelstein A, et al. Comparing the performance of metaheuristics for the analysis of multi-stakeholder tradeoffs in requirements optimisation [J]. Information and Software Technology, 2011, 53(7): 761-773

[50] Kumari A C, Srinivas K, Gupta M P. Software requirements selection using quantum-inspired multi-objective differential evolution algorithm [C] //Proc of the 2012 CSI Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2012: Article No.23

[51] Harman M, Krinke J, Ren Jian, et al. Search based data sensitivity analysis applied to requirement engineering [C] // Proc of the 2009 Annual Conf on Genetic and Evolutionary Computation. New York: ACM, 2009: 1681-1688

[52] Chaves-Gonzalez J M, Perez-Toledano M A, Navasa A. Software requirement optimization using a multiobjective swarm intelligence evolutionary algorithm [J]. Knowledge-Based Systems, 2015, 83: 105-115

[53] Del Sagrado J, Del Águila I M, Orellana F J. Multi-objective ant colony optimization for requirements selection [J]. Empirical Software Engineering, 2015, 20(3): 577-610

[54] DeVries B, Cheng B H C. Automatic detection of incomplete requirements using symbolic analysis and evolutionary computation [G] //LNCS 10452: Proc of Int Symp on Search Based Software Engineering. Berlin: Springer, 2017: 49-64

[55] Lim S L, Damian D, Finkelstein A. StakeSource2.0: Using social networks of stakeholders to identify and prioritise requirements [C] //Proc of the 2011 Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2011: 1022-1024

[56] Lim S L, Finkelstein A. StakeRare: Using social networks and collaborative filtering for large-scale requirements elicitation [J]. IEEE Transactions on Software Engineering, 2011, 38(3): 707-735

[57] Wang Hao, Wang Yasha, Wang Jiangtao. A participant recruitment framework for crowdsourcing based software requirement acquisition [C] //Proc of the 2014 IEEE Int Conf on Global Software Engineering. Piscataway, NJ: IEEE, 2014: 65-73

[58] Hosseini M, Shahri A, Phalp K, et al. Configuring crowdsourcing for requirements elicitation [C] //Proc of the 2015 IEEE Int Conf on Research Challenges in Information Science. Piscataway, NJ: IEEE, 2015: 133-138

[59] Adepetu A, Khaja A A, Al Abd Y, et al. Crowdrequire: A requirements engineering crowdsourcing platform [C] //Proc of the 2012 AAAI Spring Symp Series. Menlo Park, CA: AAAI, 2012: Article No.1

[60] Snijders R, Dalpiaz F, Hosseini M, et al. Crowd-centric requirements engineering [C] //Proc of the 2014 IEEE/ACM Int Conf on Utility and Cloud Computing. Piscataway, NJ: IEEE, 2014: 614-615

[61] Breaux T D, Schaub F. Scaling requirements extraction to the crowd: Experiments with privacy policies [C] //Proc of the 2014 IEEE Int Requirements Engineering Conf. Piscataway, NJ: IEEE, 2014: 163-172

[62] Nascimento P, Aguas R, Schneider D, et al. An approach to requirements categorization using Kano's model and crowds [C] //Proc of the 2012 IEEE Int Conf on Computer Supported Cooperative Work in Design. Piscataway, NJ: IEEE, 2012: 387-392

[63] Liang Peng, Avgeriou P, He Keqing, et al. From collective knowledge to intelligence: Pre-requirements analysis of large and complex systems [C] //Proc of the 2010 Workshop on Web 2.0 for Software Engineering. New York: ACM, 2010: 26-30

[64] Bagnall A J, Rayward-Smith V J, Whittley I M. The next release problem [J]. Information & Software Technology, 2001, 43(14): 883-890

[65] Nebro A J, Durillo J J, Luna F, et al. MOCell: A cellular genetic algorithm for multiobjective optimization [J]. International Journal of Intelligent Systems, 2009, 24(7): 726-746

[66] Robinson W N, Pawlowski S D, Volkov V. Requirements interaction management [J]. ACM Computing Surveys, 2003, 35(2): 132-190

[67] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-Ⅱ [J]. IEEE Transactions on Evolutionary Computation, 2002, 6(2): 182-197

[68] Tonella P, Susi A, Palma F. et al. Interactive requirements prioritization using a genetic algorithm [J]. Information & Software Technology, 2013, 55(1): 173-187

[69] Wever M, Van Rooijen L, Hamann H. Active coevolutionary learning of requirements specifications from examples [C] //Proc of the 2017 Genetic and Evolutionary Computation Conf. New York: ACM, 2017: 1327-1334

[70] Sultanov H, Hayes J H, Kong W K. Application of swarm techniques to requirements tracing [J]. Requirements Engineering, 2011, 16(3): 209-226

[71] Yue Tao, Ali S. Applying search algorithms for optimizing stakeholders familiarity and balancing workload in requirements assignment [C] //Proc of the 2014 Annual Conf on Genetic and Evolutionary Computation. New York: ACM, 2014: 1295-1302

[72] Jiang Yi, Wang Shijun, Fu Kai, et al. A collaborative conceptual modeling tool based on stigmergy mechanism [C] //Proc of the 2016 Asia-Pacific Symp on Internetware. New York: ACM, 2016: 11-18

[73] Nebeling M, Leone S, Norrie M C. Crowdsourced Web engineering and design [G] //LNCS 7387: Proc of Int Conf on Web Engineering. Berlin: Springer, 2012: 31-45

[74] Lagerström R, Johnson P, Ekstedt M. Search-based design of large software systems-of-systems [C] //Proc of the 2015 Int Workshop on Software Engineering for Systems-of-Systems. Piscataway, NJ: IEEE, 2015: 44-47

[75] Andrade S S, De Araújo Macêdo R J. Toward systematic conveying of architecture design knowledge for self-adaptive systems [C] //Proc of the 2013 IEEE Int Conf on Self-Adaptation and Self-Organizing Systems Workshops. Piscataway, NJ: IEEE, 2013: 23-24

[76] Andrade S S, Macêdo R J A. A search-based approach for architectural design of feedback control concerns in self-adaptive systems [C] //Proc of the 2013 IEEE Int Conf on Self-Adaptive and Self-Organizing Systems. Piscataway, NJ: IEEE, 2013: 61-70

[77] Li Rui, Chaudron M R V, Ladan R C. Towards automated software architectures design using model transformations and evolutionary algorithms [C] //Proc of the 2010 Annual Conf Companion on Genetic and Evolutionary Computation. New York: ACM, 2010: 2097-2098

[78] Féderle É L, Do Nascimento Ferreira T, Colanzi T E, et al. OPLA-Tool: A support tool for search-based product line architecture design [C] //Proc of the 2015 Int Conf on Software Product Line. New York: ACM, 2015: 370-373

[79] Mariani T, Vergilio S R, Colanzi T E. Search based design of layered product line architectures [C] //Proc of the 2015 IEEE Annual Computer Software and Applications Conf. Piscataway, NJ: IEEE, 2015: 270-275

[80] Wu Zhiqiao, Tang Jiafu. Designing and reporting on computational experiments of multi-objective component selection algorithm [J]. International Journal of Information Technology & Decision Making, 2015, 14(2): 375-394

[81] Colanzi T E, Vergilio S R. A feature-driven crossover operator for product line architecture design optimization [C] //Proc of the 2014 IEEE Annual Computer Software and Applications Conf. Piscataway, NJ: IEEE, 2014: 43-52

[82] Guizzo G, Colanzi T E, Vergilio S R. A pattern-driven mutation operator for search-based product line architecture design [G] //LNCS 8636: Proc of Int Symp on Search Based Software Engineering. Berlin: Springer, 2014: 77-91

[83] Colanzi T E, Vergilio S R. Representation of software product line architectures for search-based design [C] //Proc of the 2013 Int Workshop on Combining Modelling and Search-Based Software Engineering. Piscataway, NJ: IEEE, 2013: 28-33

[84] Ardagna D, Gibilisco G P, Ciavotta M, et al. A multi-model optimization framework for the model driven design of cloud applications [G] //LNCS 8636: Proc of the 2014 Int Symp on Search Based Software Engineering. Berlin: Springer, 2014: 61-76

[85] Fanjiang Y, Syu Y. Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach [J]. Information and Software Technology, 2014, 56(3): 352-373

[86] Simons C L, Parmee I C, Gwynllyw R. Interactive, evolutionary search in upstream object-oriented class design [J]. IEEE Transactions on Software Engineering, 2010, 36(6): 798-816

[87] Räihä O. A survey on search-based software design [J]. Computer Science Review, 2010, 4(4): 203-249

[88] Harman M, Jia Y, Krinke J, et al. Search based software engineering for software product line engineering: A survey and directions for future work [C] //Proc of the 2014 Int Software Product Line Conf-Volume 1. New York: ACM, 2014: 5-18

[89] Wu Zhiqiao, Tang Jiafu, Kwong C K, et al. An optimization model for reuse scenario selection considering reliability and cost in software product line development [J]. International Journal of Information Technology & Decision Making, 2011, 10(5): 811-841

[90] Canfora G, Di Penta M, Esposito R, et al. An approach for QoS-aware service composition based on genetic algorithms [C] //Proc of the 2005 Annual Conf on Genetic and Evolutionary Computation. New York: ACM, 2005: 1069-1075

[91] Simons C L, Smith J, White P. Interactive ant colony optimization (iACO) for early lifecycle software design [J]. Swarm Intelligence, 2014, 8(2): 139-157

[92] Lasecki W S, Kim J, Rafter N, et al. Apparition: Crowdsourced user interfaces that come to life as you sketch them [C] //Proc of the 2015 Annual ACM Conf on Human Factors in Computing Systems. New York: ACM, 2015: 1925-1934

[93] LaToza T D, Chen M, Jiang Luxi, et al. Borrowing from the crowd: A study of recombination in software design competitions [C] //Proc of the 2015 IEEE/ACM IEEE Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2015, 1: 551-562

[94] Fast E, Steffee D, Wang L, et al. Emergent, crowd-scale programming practice in the IDE [C] //Proc of the 2014 SIGCHI Conf on Human Factors in Computing Systems. New York: ACM, 2014: 2491-2500

[95] Mujumdar D, Kallenbach M, Liu B, et al. Crowdsourcing suggestions to programming problems for dynamic Web development languages [C] //Proc of the 2011 Int Conf on Human Factors in Computing Systems. New York: ACM, 2011: 1525-1530

[96] Chen Fuxiang, Kim S. Crowd debugging [C] //Proc of the 2015 Joint Meeting on Foundations of Software Engineering. New York: ACM, 2015: 320-332

[97] Auler R, Borin E, De Halleux P, et al. Addressing javascript JIT engines performance quirks: A crowdsourced adaptive compiler [G] //LNCS 8409: Proc of the 2014 Int Conf on Compiler Construction. Berlin: Springer, 2014: 218-237

[98] Fast E, Bernstein M S. Meta: Enabling programming languages to learn from the crowd [C] //Proc of the 2016 Annual Symp on User Interface Software and Technology. New York: ACM, 2016: 259-270

[99] Adriano C M, Van Der Hoek A. Exploring microtask crowdsourcing as a means of fault localization [J]. arXiv preprint arXiv:1612.03015, 2016

[100] Fryer H, Simperl E. Web science challenges in researching bug bounties [C] //Proc of the 2017 ACM on Web Science Conf. New York: ACM, 2017: 273-277

[101] Cody-Kenny B, Fenton M, Ronayne A, et al. A search for improved performance in regular expressions [C] //Proc of the 2017 Genetic and Evolutionary Computation Conf. New York: ACM, 2017: 1280-1287

[102] Langdon W B, Harman M. Optimizing existing software with genetic programming [J]. IEEE Transactions on Evolutionary Computation, 2015, 19(1): 118-135

[103] Langdon W B, Lam B Y H, Petke J, et al. Improving CUDA DNA analysis software with genetic programming [C] //Proc of the 2015 Annual Conference on Genetic and Evolutionary Computation. New York: ACM, 2015: 1063-1070

[104] Petke J, Harman M, Langdon W B, et al. Using genetic improvement and code transplants to specialise a C++ program to a problem class [G] //LNCS 8599: Proc of European Conf on Genetic Programming. Berlin: Springer, 2014: 137-149

[105] Orlov M, Sipper M. Flight of the FINCH through the Java wilderness [J]. IEEE Transactions on Evolutionary Computation, 2011, 15(2): 166-182

[106] White D R, Arcuri A, Clark J A. Evolutionary improvement of programs [J]. IEEE Transactions on Evolutionary Computation, 2011, 15(4): 515-538

[107] Weimer W, Nguyen T V, Le Goues C, et al. Automatically finding patches using genetic programming [C] //Proc of the 2009 Int Conf on Software Engineering. Piscataway, NJ: IEEE Computer Society, 2009: 364-374

[108] Le Goues C, Dewey-Vogt M, Forrest S, et al. A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each [C] //Proc of the 2012 Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2012: 3-13

[109] Nguyen T V, Weimer W, Le Goues C, et al. Using execution paths to evolve software patches [C] //Proc of the 2009 Int Conf on Software Testing, Verification, and Validation Workshops. Piscataway, NJ: IEEE, 2009: 152-153

[110] Schulte E, Forrest S, Weimer W. Automated program repair through the evolution of assembly code [C] //Proc of the 2010 IEEE/ACM Int Conf on Automated Software Engineering. New York: ACM, 2010: 313-316

[111] Fast E, Le Goues C, Forrest S, et al. Designing better fitness functions for automated program repair [C] //Proc of the 2010 Annual Conf on Genetic and Evolutionary Computation. New York: ACM, 2010: 965-972

[112] Qi Yuhua, Mao Xiaoguang, Lei Yan. Efficient automated program repair through fault-recorded testing prioritization [C] //Proc of the 2013 IEEE Int Conf on Software Maintenance. Piscataway, NJ: IEEE, 2013: 180-189

[113] Cochran R A, D'Antoni L, Livshits B, et al. Program boosting: Program synthesis via crowd-sourcing [J]. ACM SIGPLAN Notices, 2015, 50(1): 677-688

[114] Nguyen H D T, Qi Dawei, Roychoudhury A, et al. Semfix: Program repair via semantic analysis [C] //Proc of the 2013 Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2013: 772-781

[115] Kim D, Nam J, Song J, et al. Automatic patch generation learned from human-written patches [C] //Proc of the 2013 Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2013: 802-811

[116] Terragni V, Liu Yepang, Cheung S C. CSNIPPEX: Automated synthesis of compilable code snippets from Q&A sites [C] //Proc of the 2016 Int Symp on Software Testing and Analysis. New York: ACM, 2016: 118-129

[117] Bruce B R, Petke J, Harman M. Reducing energy consumption using genetic improvement [C] //Proc of the 2015 Annual Conf on Genetic and Evolutionary Computation. New York: ACM, 2015: 1327-1334

[118] Manotas I, Pollock L, Clause J. SEEDS: A software engineer's energy-optimization decision support framework [C] //Proc of the 2014 Int Conf on Software Engineering. New York: ACM, 2014: 503-514

[119] Li Ding, Tran A H, Halfond W G J. Making Web applications more energy efficient for OLED smartphones [C] //Proc of the 2014 Int Conf on Software Engineering. New York: ACM, 2014: 527-538

[120] Wu Fan, Weimer W, Harman M, et al. Deep parameter optimisation [C] //Proc of the 2015 Annual Conf on Genetic and Evolutionary Computation. New York: ACM, 2015: 1375-1382

[121] Malhotra R, Khari M. Heuristic search-based approach for automated test data generation: A survey [J]. International Journal of Bio-Inspired Computation, 2013, 5(1): Article No.1

[122] Zhang Xiaofang, Feng Yang, Liu Di, et al. Research progress of crowdsourced software testing [J]. Journal of Software, 2018, 29(1): 69-88 (in Chinese)
(章晓芳, 冯洋, 刘頔, 等. 众包软件测试技术研究进展[J]. 软件学报, 2018, 29(1): 69-88)

[123] Pham R, Singer L, Liskin O, et al. Creating a shared understanding of testing culture on a social coding site [C] //Proc of the 2013 Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2013: 112-121

[124] Pham R, Singer L, Schneider K. Building test suites in social coding sites by leveraging drive-by commits [C] //Proc of the 2013 Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2013: 1209-1212

[125] Chen Ning, Kim S. Puzzle-based automatic testing: Bringing humans into the loop by solving puzzles [C] //Proc of the 2012 IEEE/ACM Int Conf on Automated Software Engineering. Piscataway, NJ: IEEE, 2012: 140-149

[126] Pastore F, Mariani L, Fraser G. Crowdoracles: Can the crowd solve the oracle problem? [C] //Proc of the 2013 IEEE Int Conf on Software Testing, Verification and Validation. Piscataway, NJ: IEEE, 2013: 342-351

[127] Dolstra E, Vliegendhart R, Pouwelse J. Crowdsourcing GUI tests [C] //Proc of the 2013 IEEE Int Conf on Software Testing, Verification and Validation. Piscataway, NJ: IEEE, 2013: 332-341

[128] Vliegendhart R, Dolstra E, Pouwelse J. Crowdsourced user interface testing for multimedia applications [C] //Proc of the ACM Multimedia 2012 Workshop on Crowdsourcing for Multimedia. New York: ACM, 2012: 21-22

[129] Musson R, Richards J, Fisher D, et al. Leveraging the crowd: How 48 000 users helped improve Lync performance [J]. IEEE Software, 2013, 30(4): 38-45

[130] Schneider C, Cheung T. The Power of the Crowd: Performing Usability Testing Using an On-Demand Workforce [M]. Information Systems Development. Berlin: Springer, 2013: 551-560

[131] Mahajan M, Kumar S, Porwal R. Applying genetic algorithm to increase the efficiency of a data flow-based test data generation approach [J]. ACM SIGSOFT Software Engineering Notes, 2012, 37(5): Article No.14

[132] Rathore A, Bohara A, Prashil R G, et al. Application of genetic algorithm and tabu search in software testing [C] // Proc of the 2011 Annual ACM Bangalore Conf on Compute. New York: ACM, 2011: Article No.23

[133] Li Kewen, Zhang Zilu, Liu Wenying. Automatic test data generation based on ant colony optimization [C] //Proc of the 2009 Int Conf on Natural Computation. Piscataway, NJ: IEEE, 2009: 216-220

[134] Srivastava P R, Baby K. Automated software testing using metahurestic technique based on an ant colony optimization [C] //Proc of the 2010 Int Symp on Electronic System Design. Piscataway, NJ: IEEE, 2010: 235-240

[135] Rao K K, Raju G, Nagaraj S. Optimizing the software testing efficiency by using a genetic algorithm: A design methodology [J]. ACM SIGSOFT Software Engineering Notes, 2013, 38(3): Article No.13

[136] Singh Y, Kaur A, Suri B. Test case prioritization using ant colony optimization [J]. ACM SIGSOFT Software Engineering Notes, 2010, 35(4): Article No.14

[137] Just R, Schweiggert F. Automating unit and integration testing with partial oracles [J]. Software Quality Journal, 2011, 19(4): 753-769

[138] Bauersfeld S, Wappler S, Wegener J. A metaheuristic approach to test sequence generation for applications with a GUI [G] //LNCS 6956: Proc of Int Symp on Search Based Software Engineering. Berlin: Springer, 2011: 173-187

[139] Bauersfeld S, Wappler S, Wegener J. An approach to automatic input sequence generation for GUI testing using ant colony optimization [C] //Proc of the 2011 Annual Conference Companion on Genetic and Evolutionary Computation. New York: ACM, 2011: 251-252

[140] Gross F, Fraser G, Zeller A. EXSYST: Search-based GUI testing [C] //Proc of the 2012 Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2012: 1423-1426

[141] Salvesen K, Galeotti J P, Gross F, et al. Using dynamic symbolic execution to generate inputs in search-based GUI testing [C] //Proc of the 2015 Int Workshop on Search-Based Software Testing. Piscataway, NJ: IEEE, 2015: 32-35

[142] Gu Yuanyan, Ge Yujia. Search-based performance testing of applications with composite services [C] //Proc of the 2009 Int Conf on Web Information Systems and Mining. Piscataway, NJ: IEEE, 2009: 320-324

[143] Wegener J, Bühler O. Evaluation of different fitness functions for the evolutionary testing of an autonomous parking system [G] //LNCS 3103: Proc of Genetic and Evolutionary Computation Conf. Berlin: Springer, 2004: 1400-1412

[144] Bühler O, Wegener J. Evolutionary functional testing [J]. Computers & Operations Research, 2008, 35(10): 3144-3160

[145] Wagner M. Maximising axiomatization coverage and minimizing regression testing time [C] //Proc of the 2014 IEEE Congress on Evolutionary Computation. Piscataway, NJ: IEEE, 2014: 2885-2892

[146] Alba E, Chicano F. Ant colony optimization for model checking [G] //LNCS 4739: Proc of Int Conf on Computer Aided Systems Theory. Berlin: Springer, 2007: 523-530

[147] Mahanti P K, Banerjee S. Automated testing in software engineering: Using ant colony and self-regulated swarms [C] //Proc of the 2006 IASTED Int Conf on Modelling and Simulation. Calgary, AB, Canada: ACTA Press, 2006: 443-448

[148] Katz G, Peled D. Genetic programming and model checking: Synthesizing new mutual exclusion algorithms [G] //LNCS 5311: Proc of Int Symp on Automated Technology for Verification and Analysis. Berlin: Springer, 2008: 33-47

[149] Katz G, Peled D. Model checking-based genetic programming with an application to mutual exclusion [G] // LNCS 4963: Proc of Int Conf on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Springer, 2008: 141-156

[150] Mcminn P, Stevenson M, Harman M. Reducing qualitative human oracle costs associated with automatically generated test data [C] //Proc of the 2010 Int Workshop on Software Test Output Validation. New York: ACM, 2010: Article No.1

[151] Afshan S, McMinn P, Stevenson M. Evolving readable string test inputs using a natural language model to reduce human oracle cost [C] //Proc of the 2013 IEEE Int Conf on Software Testing, Verification and Validation. Piscataway, NJ: IEEE, 2013: 352-361

[152] Dietl W, Dietzel S, Ernst M D, et al. Verification games: Making verification fun [C] //Proc of the 2012 Workshop on Formal Techniques for Java-like Programs. New York: ACM, 2012: 42-49

[153] Li Wenchao, Seshia S A, Jha S. CrowdMine: Towards crowdsourced human-assisted verification [C] //Proc of the 2012 Annual Design Automation Conf. New York: ACM, 2012: 1254-1255

[154] Lakhotia K, McMinn P, Harman M. Automated test data generation for coverage: Haven't we solved this problem yet? [C] //Testing: Academic and Industrial Conf-Practice and Research Techniques. Piscataway, NJ: IEEE, 2009: 95-104

[155] Memon A, Banerjee I, Nagarajan A. GUI ripping: Reverse engineering of graphical user interfaces for testing [C] // Proc of the 2003 Working Conf on Reverse Engineering. Piscataway, NJ: IEEE, 2003: 260-269

[156] Harman M, Jia Yue, Zhang Yuanyuan. Achievements, open problems and challenges for search based software testing [C] //Proc of the 2015 IEEE Int Conf on Software Testing, Verification and Validation. Piscataway, NJ: IEEE, 2015: 11-22

[157] Ali R, Solis C, Salehie M, et al. Social sensing: When users become monitors [C] //Proc of the 2011 ACM SIGSOFT Symp and the European Conf on Foundations of Software Engineering. New York: ACM, 2011: 476-479

[158] He Huihong, Ma Zhiyi, Chen Hongjie, et al. How the crowd impacts commercial applications: A user-oriented approach [C] //Proc of the 2014 Int Workshop on Crowd-based Software Development Methods and Technologies. New York: ACM, 2014: Article No.1

[159] Nebeling M, Norrie M C. Context-aware and adaptive Web interfaces: A crowdsourcing approach [G] //LNCS 7059: Proc of Int Conf on Web Engineering. Berlin: Springer, 2011: 167-170

[160] Schulte E. Neutral Networks of Real-World Programs and Their Application to Automated Software Evolution [M]. Albuquerque, NM: The University of New Mexico, 2015

[161] Cooper K D, Schielke P J, Subramanian D. Optimizing for reduced code space using genetic algorithms [J]. ACM SIGPLAN Notices, 1999, 34(7): Article No.1

[162] Kebir S, Borne I, Meslati D. A genetic algorithm-based approach for automated refactoring of component-based software [J]. Information and Software Technology, 2017, 88: 17-36

[163] Mkaouer M W, Kessentini M, Cinnéide M Ó, et al. A robust multi-objective approach to balance severity and importance of refactoring opportunities [J]. Empirical Software Engineering, 2017, 22(2): 894-927

[164] Mancoridis S, Mitchell B S, Rorres C, et al. Using automatic clustering to produce high-level system organizations of source code [C] //Proc of the 1998 Int Workshop on Program Comprehension. Piscataway, NJ: IEEE, 1998: 45-52

[165] Mancoridis S, Mitchell B S, Chen Y, et al. Bunch: A clustering tool for the recovery and maIntenance of software system structures [C] //Proc of the 1999 IEEE Int Conf on Software Maintenance. Piscataway, NJ: IEEE, 1999: 50-59

[166] Harman M, Hierons R M, Proctor M. A new representation and crossover operator for search-based optimization of software modularization [C] //Proc of the 2002 Genetic and Evolutionary Computation Conf. New York: ACM, 2002: 1351-1358

[167] Ghannem A, El Boussaidi G, Kessentini M. Model refactoring using interactive genetic algorithm [G] //LNCS 8084: Proc of Int Symp on Search Based Software Engineering. Berlin: Springer, 2013: 96-110

[168] Mariani T, Vergilio S R. A systematic review on search-based refactoring [J]. Information and Software Technology, 2017, 83: 14-34

[169] Jiau H C, Yang Fengpu. Facing up to the inequality of crowdsourced API documentation [J]. ACM SIGSOFT Software Engineering Notes, 2012, 37(1): 45-53

[170] Exton C, Wasala A, Buckley J, et al. Micro crowdsourcing: A new model for software localisation [J]. Localisation Focus, 2009, 8(1): 81-89

[171] Sahraoui H, Valtchev P, Konkobo I, et al. Object identification in legacy code as a grouping problem [C] // Proc of the 2002 Annual Int Computer Software and Applications. Piscataway, NJ: IEEE, 2002: 689-696

[172] Tufano M, Palomba F, Bavota G, et al. When and why your code starts to smell bad [C] //Proc of the 2015 Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2015: 403-414

[173] Sharifi M, Fink E, Carbonell J G. Smartnotes: Application of crowdsourcing to the detection of Web threats [C] //Proc of the 2011 IEEE Int Conf on Systems, Man, and Cybernetics. Piscataway, NJ: IEEE, 2011: 1346-1350

[174] Chilana P K. Supporting users after software deployment through selection-based crowdsourced contextual help [D]. Seattle: University of Washington Libraries, 2013

[175] Chilana P K, Ko A J, Wobbrock J O. LemonAid: Selection-based crowdsourced contextual help for Web applications [C] //Proc of the 2012 SIGCHI Conf on Human Factors in Computing Systems. New York: ACM, 2012: 1549-1558

[176] Chilana P K, Ko A J, Wobbrock J O, et al. A multi-site field study of crowdsourced contextual help: Usage and perspectives of end users and software teams [C] //Proc of the 2013 SIGCHI Conf on Human Factors in Computing Systems. New York: ACM, 2013: 217-226

[177] Leimeister J M, Huber M, Bretschneider U, et al. Leveraging crowdsourcing: Activation-supporting components for IT-based ideas competition [J]. Journal of Management Information Systems, 2009, 26(1): 197-224

[178] Chang C E, Chao Chikuang, Hsieh S Y, et al. Spmnet: A formal methodology for software management [C] //Proc of the 1994 Annual Int Computer Software and Applications Conf. Piscataway, NJ: IEEE, 1994: 57

[179] Kapur P, Ngo-The A, Ruhe G, et al. Optimized staffing for product releases and its application at chartwell technology [J]. Journal of Software MaIntenance and Evolution: Research and Practice, 2008, 20(5): 365-386

[180] Arellano C, Díaz O, Iturrioz J. Crowdsourced Web augmentation: A security model [G] //LNCS 6488: Proc of Int Conf on Web Information Systems Engineering. Berlin: Springer, 2010: 294-307

[181] Harman M, Lakhotia K, Singer J, et al. Cloud engineering is search based software engineering too [J]. Journal of Systems and Software, 2013, 86(9): 2225-2241

[182] Madhavji N H, Miranskyy A, Kontogiannis K. Big picture of big data software engineering: With example research challenges [C] //Proc of the 2015 Int Workshop on Big Data Software Engineering. Piscataway, NJ: IEEE, 2015: 11-14

[183] Stephenson M, Amarasinghe S, Martin M, et al. Meta optimization: Improving compiler heuristics with machine learning [J]. ACM SIGPLAN Notices, 2003, 38(5): 77-90

[184] Jia Yue, Cohen M B, Harman M, et al. Learning combinatorial interaction test generation strategies using hyperheuristic search [C] //Proc of the 2015 Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2015: 540-550

[185] Varshney L R. Participation in crowd systems [C] //Proc of the 2012 Annual Allerton Conf on Communication, Control, and Computing. Piscataway, NJ: IEEE, 2012: 996-1001

[186] Groen E C, Seyff N, Ali R, et al. The crowd in requirements engineering: The landscape and challenges [J]. IEEE Software, 2017, 34(2): 44-52

[187] Allahbakhsh M, Benatallah B, Ignjatovic A, et al. Quality control in crowdsourcing systems: Issues and directions [J]. IEEE Internet Computing, 2013, 17(2): 76-81

[188] Li Ke, Xiao Junchao, Wang Yongji, et al. Analysis of the key factors for software quality in crowdsourcing development: An empirical study on topcoder. com [C] // Proc of the 2013 IEEE Annual Computer Software and Applications Conf. Piscataway, NJ: IEEE, 2013: 812-817

Xu Lixin, born in 1996. Master candidate. His main research interests include collective intelligence based software engineering.

Wu Huayao, born in 1989. PhD, assistant researcher. His main research interests include software testing and analysis.