doi:10.11920/xnmdzk.2025.03.007

依赖关联的多目标缺陷分派优化模型

崔梦天,罗婕妤,王　锡
(西南民族大学计算机与人工智能学院,四川 成都 610041)

摘　要:随着开源软件规模和复杂度的不断提升,软件设计和开发的难度显著增加,软件缺陷的数量也迅速增长.缺陷分派是软件开发中确保质量的关键任务,近年来深度学习领域提出了多种自动化或半自动化的分派方法.然而,这些方法仍存在诸多局限.传统分派方法主要挖掘缺陷报告的文本特征,忽视了缺陷间的依赖关系及其引发的阻塞问题,而缺陷阻塞不仅会降低修复效率,甚至可能导致修复进程停滞.为解决缺陷阻塞问题并降低跨项目的认知成本,提出了一个依赖关联的多目标缺陷分派优化模型.该模型构建缺陷依赖树并更新缺陷优先级,调整缺陷修复顺序.同时,模型引入非支配排序遗传算法,平衡认知成本与修复偏好两个目标,向开发人员推荐排序后的最佳缺陷集合.实验结果表明,该方法在 Eclipse 和 Mozilla 数据集上的部分阻塞缺陷修复时间方面取得了显著优化效果.

关键词:软件缺陷分派;缺陷依赖;多级目标搜索
中图分类号:TP311　　**文献标志码**:A　　**文章编号**:2095-4271(2025)03-0288-10

Dependency-associated multi-objective bug assignment optimization model

CUI Mengtian, LUO Jieyu, WANG Xi
(School of Computer Science and Artificial Intelligence, Southwest Minzu University, Chengdu 610041, China)

Abstract: As the scale and complexity of open-source software continue to grow, the challenges in software design and development have significantly increased, leading to a rapid rise in the number of software bugs. Bug assignment is a critical task in software development to ensure quality, and in recent years, various automated or semi-automated assignment methods have been proposed in the field of deep learning. However, these methods still have several limitations. Traditional assignment approaches primarily focus on extracting textual features from bug reports while overlooking the dependencies between bugs and the blocking issues they may cause. Bug blocking not only reduces the efficiency of bug-fixing but can even lead to stagnation in the fixing process. To address the issue of bug blocking and reduce cross-project cognitive costs, this paper proposed a dependency-aware multi-objective bug assignment optimization model. The model constructed a bug dependency tree, updated bug priority, and adjusted the bug-fixing order. Additionally, it incorporated a non-dominated sorting genetic algorithm to balance cognitive cost and fixing preference, recommending an optimized set of prioritized bugs to developers. Experimental results demonstrated that the proposed approach achieved significant improvements in the fixing time of certain blocked bugs in the Eclipse and Mozilla datasets.

Keywords: bug triage; bug dependency; multi-objective search

开源软件(Open Source Software, OSS)以其可伸缩性、灵活性和重用性在工业自动化领域迅速发展,已成为推动自动化技术的重要趋势.然而,随着软件规模和复杂度的快速增长,开源软件的设计和开发难

收稿日期:2025-03-08
作者简介:崔梦天(1972-),女,教授,博士,研究方向:信息安全技术.E-mail:mengtian.cui@swun.edu.cn
通信作者:王锡(2000-),男,研究方向:软件缺陷分派. E-mail:220854112004@stu.swun.edu.cn
基金项目:人社部高端外国专家引进计划项目(H20240672);四川省国际科技创新合作项目(2025YFHZ0178);中央高校基本科研业务费专项资金优秀学生培养工程项目(2023NYXXS043)

度显著增加,软件缺陷也呈现爆发式增长,严重威胁软件质量[1].为此,许多开源软件项目采用缺陷报告和缺陷跟踪系统来记录和管理软件缺陷,以确保软件的持续稳定运行[2].缺陷分派是软件缺陷管理中的关键环节,缺陷分类者通过缺陷跟踪系统将缺陷报告分派给最合适的开发人员进行修复[3].然而,如果缺陷被错误分派,不仅会导致修复延误,还会增加修复的成本和复杂性.

随着深度学习技术的发展,其强大的特征学习和数据处理能力为缺陷自动分派的研究提供了新的契机. 已有研究中,Uddin 等[4]提出的 DevSched 模型使用余弦相似度将缺陷报告与开发者进行匹配,动态分配开发者的工作任务,提升了效率.Jahanshahi 等[5]的 ADPTriage 模型结合马尔可夫决策过程和动态规划,优化缺陷分派中的不确定性.Zeng 和 Zhang[6]利用循环神经网络和卷积神经网络提取开发者活动与缺陷报告特征,提升分派准确性.这些方法融入缺陷自动分派模型,通过评估开发人员的技术专长,为缺陷报告匹配合适的修复者.该过程无须分类者深入了解所有开发人员的技能范围,不仅降低了工作压力,还提高了分派的精准度,对优化开源软件开发成本具有重要价值.

目前主流的软件缺陷分派研究中,缺陷报告通常被孤立地处理并单独分派给开发人员进行修复.这些方法未考虑缺陷报告之间的依赖关系[7-8].然而,研究表明大多数软件缺陷并非孤立存在,而是相互关联的[9].在推荐缺陷修复列表时,如果这些缺陷共享相同产品组件中的潜在文件,这将有助于缩短开发人员在不同包或文件之间切换时所需的认知时间和努力.缺陷阻塞(Bug Blocking)指的是由于缺陷之间存在依赖关系,某些缺陷的修复必须依赖于其他缺陷的先行修复,导致缺陷报告长时间处于未解决状态,进而延长了整体修复周期[10].为解决缺陷阻塞问题、降低认知成本,本文重点关注缺陷报告的依赖关系和开发人员的修复偏好,将缺陷分派视为多目标优化问题,提出了一种依赖关联的多目标缺陷分派优化模型.模型通过挖掘缺陷报告间的依赖关系,构建缺陷依赖树,以此更新缺陷报告的优先级,调节不同缺陷的修复顺序.使用非支配排序遗传算法(NSGA-Ⅱ)平衡开发人员的认知时间成本和开发人员的修复偏好这两个属性的权重.模型实现了二者的均衡最优,最终向开发人员推荐一组修复优先级排序后的最佳缺陷报告集合,以此得到一种合理且高效的软件缺陷分派方案.

1　相关工作

开源软件项目广泛使用缺陷管理系统来收集缺陷信息,并将其分派给开发者进行修复,这一过程称为缺陷分派或开发者推荐[2].早期研究主要通过提取缺陷报告中的文本特征(如元数据、摘要、描述、评论等)来训练机器学习模型,以推荐合适的开发者.例如,Xuan 等[11]利用标题和描述训练 SVM 和 MNB 分类器进行开发者推荐,而 Jonsson 等[12]通过集成学习和堆栈泛化技术整合多种文本分类器.同时,部分学者也开始关注基于开发者合作关系的推荐方法.例如,刘海洋等[13]通过 LDA 模型刻画开发者技能,结合开发者合作网络进行推荐. 崔梦天等[14]将 LDA 与 Bert 模型相结合,融合了主题与上下文识别的优点,同时使用二级特征向量再检测方法节省了重复缺陷检测的时间.Bhattacharya 等[15-16]基于机器学习和抛掷图,通过附加属性和内折叠更新提高推荐准确率.吴克奇等[17]通过挖掘缺陷历史相似性,构建历史相似推荐网络,利用协同过滤算法对新缺陷进行推荐.

近年来,深度学习技术的应用进一步推动了缺陷分派研究的发展.Xi 等[18]使用双向神经网络提取文本特征,并通过单向循环神经网络捕捉开发者活跃度.Xi 等[19]继续提出了 ITriage 模型,结合序列到序列模型学习缺陷报告文本与抛掷序列特征.此外,图神经网络(GNN)的引入也推动了基于图模型的缺陷分派方法的发展.Zaidi 等[20]通过图卷积网络(GCN)捕捉异构图中的词-词边和词-缺陷文档关系,提升了缺陷分派的准确性.李元香等[21]则基于余弦相似性构建缺陷报告关系网络并提取特征.崔梦天等[22]从复杂网络角度出发,提出了一种基于图神经网络的软件缺陷预测框架,也为复杂网络思想在缺陷分派的应用提供了新思路.后来该团队将拓扑图引入特征选择算法中,利用对称不确定性作为特征关联度,构建特征全连接图,以此来解决复杂的多边关系[23].

在优化修复时间和开发者工作负载方面,Jahanshahi 等[24]提出了依赖性感知的缺陷分类方法(DABT),通过优先处理阻塞缺陷简化依赖关系.

Kashiwa 和 Ohira[25]将缺陷分派视为多背包问题,限制每位开发者在特定时间内的任务量以缓解任务分布不均. Park 等[26]通过 LDA 模型估算修复时间,优化修复成本.崔梦天等[27]考虑到缺陷报告类不平衡问题,采用了混合采样 SMOTEENN 方法. Mani 等[28]提出了基于 word2vec 嵌入的双向递归神经网络(DBRNN-A),能够自动学习句法与语义特征.

其他研究也不断提出创新方法. Zaidi 等[29]结合了上下文感知技术改进了卷积神经网络. Liu 等[30]提出了基于深度强化学习(BT-RL)的自动缺陷分派模型.该模型通过深度多语义特征提取高质量的文本特征,并结合在线动态匹配(ODM)分析缺陷报告和开发人员活动,进一步提升了推荐的精度和效率. Fang 等[31]通过加权图卷积网络预测缺陷修复优先级,而 Akbarinasaji 等[32]提出的部分可观察马尔可夫决策过程模型,考虑了依赖结构的不确定性. Dai 等[33]提出基于图协同过滤的缺陷分类框架,通过构建缺陷-开发者二分图和时空图卷积策略,提升分类精度.此外,在特征增强领域, Alazzam 等[34]通过研究图邻域关系提升摘要的特征向量优化缺陷分类效果. Sepahvand 等[35]通过 CNN 模型预测是否需将缺陷分派给设计人员, Dong 等[36]则提出基于邻域对比学习的图神经网络框架(NCGBT),通过对缺陷报告与开发者的关系建模预测匹配结果.

2 依赖关联的多目标缺陷分派优化模型

为解决缺陷阻塞的问题,同时降低开发人员修复过程中的认知时间成本.本文提出了一种基于搜索的软件工程方法:首先,通过挖掘缺陷报告的依赖关系,构建缺陷依赖树并更新缺陷报告的优先级,以调节阻塞缺陷的修复次序.接下来,计算缺陷报告跨组件切换成本和开发人员匹配度,将缺陷分派任务视为一个使用非支配排序遗传算法(NSGA-Ⅱ, Non-dominated Sorting Genetic Algorithm Ⅱ)的多目标优化问题.目标实现开发人员能在处理相匹配缺陷的同时,也专注于相同文件下的缺陷,减少因切换缺陷任务类型带来的认知成本.最后,结合更新后的缺陷报告优先级,在大搜索空间确定最佳组合将相匹配的缺陷报告按需推荐给开发人员,从而有效缩短缺陷定位和修复的时间.模型结构如图 1 所示.

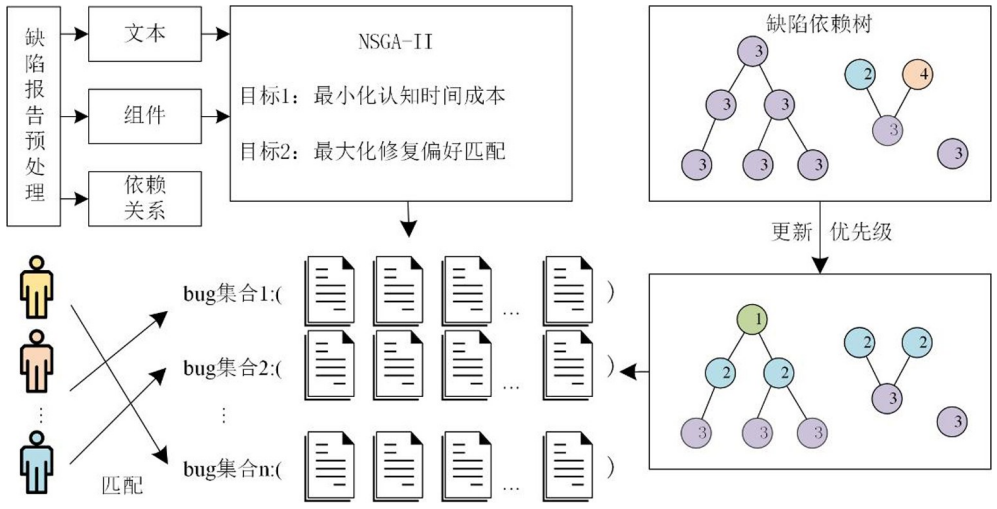

图 1 依赖关联的多目标缺陷分派优化模型

Fig.1 Dependency-associated multi-objective bug assignment optimization model

2.1 构建缺陷依赖树

每个缺陷报告都包含一组属性,例如缺陷的摘要、描述、报告日期、报告者信息、缺陷的严重性和优先级.根据 Bugzilla 的定义,缺陷报告的优先级用于确定缺陷处理的顺序和紧急程度.严重性则反映了问题的影响程度,其范围从“阻塞”(即“应用程序不可用”)到“微不足道”(即“轻微的外观问题”). 缺陷报告的优先级分为 P1(最高)至 P5(最低)五个等级,而严重性则包括以下几种类型:阻塞、关键、增强、主要、次要、正常和微不足道.

阻塞缺陷对整个软件缺陷维护过程具有不利影响,它们不仅延迟了其他缺陷的修复进程,还可能导

致修复进程停滞.阻塞缺陷的发生通常是因为它们影响了系统的关键功能或其他缺陷的修复,而不是仅仅由于缺乏足够的资源来修复它们.如果在这些依赖于阻塞缺陷修复的缺陷仍未解决之前,就将其优先分派给开发人员,会导致项目的修复进度受到阻碍.相比之下,没有依赖关系的缺陷报告可以立即开始修复,因此其修复时间通常较短.

Eclipse 和 Mozilla 两个项目中阻塞缺陷报告和非阻塞缺陷报告的数量如表 1 所示,其中 Mozilla 项目中阻塞缺陷报告的比例高达 12.5%,对整个项目的修复进程造成了较大影响 . Eclipse 和 Mozilla 项目中阻塞和非阻塞缺陷报告的修复时间中位数如表 2 所示.可以看出,非阻塞缺陷报告的修复时间普遍短于阻塞缺陷报告.由此可见,考虑缺陷依赖和阻塞关系对实现软件缺陷自动分派和整个项目修复进程都有非常重要的作用.

表 1　阻塞缺陷报告

Table 1　Blocked defect report

项目	阻塞报告数	非阻塞报告数	总数
Mozilla	4 705[2.8%]	163 319[97.2%]	168 024
Eclipse	5 861[12.5%]	41 023[87.5%]	46 884

表 2　修复时间中位数

Table 2　Median time to repair

项目	阻塞报告/天	非阻塞报告/天
Mozilla	146	51
Eclipse	100	42

因此,为了解决缺陷报告阻塞问题,调节缺陷报告的分派顺序,本文通过以下方法构建代表缺陷依赖层次关系的缺陷依赖树,并更新相应缺陷的优先级:

1)识别阻塞性缺陷报告:首先,从缺陷数据库中检索所有严重性标记为“阻塞”的缺陷报告,记录对应的缺陷报告 ID 和初始优先级.

2)挖掘依赖缺陷:对缺陷描述进行语义分析,结合历史修复记录和代码变更信息,识别可能存在依赖关系的缺陷报告.这些关联缺陷可能与被标记为“阻塞”的缺陷报告在相同的组件或模块中.对于每个具有依赖关系缺陷,提取其关联缺陷报告 ID 和优先级,并递归分析其相关缺陷,直到无法找到新的依赖关系为止.

3)构建缺陷报告依赖树:基于缺陷间的依赖关系,构建缺陷依赖树状图.在树结构中,所有的“阻塞”缺陷报告 ID 将作为子节点,关联的缺陷报告 ID 将作为父节点,依次构建层次化的依赖关系.

4)更新缺陷报告优先级:依据依赖树的层次结构,逐层更新缺陷报告的优先级,更新遵循以下规则:上层(父节点)缺陷优先级提高,以确保其优先修复,从而解除下层(子节点)缺陷的阻塞状态,提高整体修复效率.

在构建缺陷依赖树之后,父节点代表被依赖的缺陷,而子节点则表示依赖于父节点的缺陷.在实际的软件修复过程中,大多数具有依赖关系的缺陷报告通常来自同一产品或组件.当父节点的缺陷被修复后,修复其子节点的缺陷将变得更加便捷 . 如果没有遵循这一顺序,可能会导致被阻塞的缺陷报告优先分派给开发人员,从而造成修复时间的浪费.

以 PC 产品下 ID 为 42795 的缺陷报告为例构建的缺陷依赖树.缺陷报告 42795 号处于“阻塞”状态,且依赖于缺陷报告 42602 号;同时,缺陷报告 42602 号又依赖于缺陷报告 46545 号和 43371 号.在这四个缺陷报告中,所有报告的优先级均为 P3.经过优先级更新后,缺陷报告 46545 号和 43371 号的优先级由 P3 提升为 P1,意味着这两个缺陷报告将会被优先分派给开发人员进行修复.缺陷依赖树更新过程如图 2 所示.

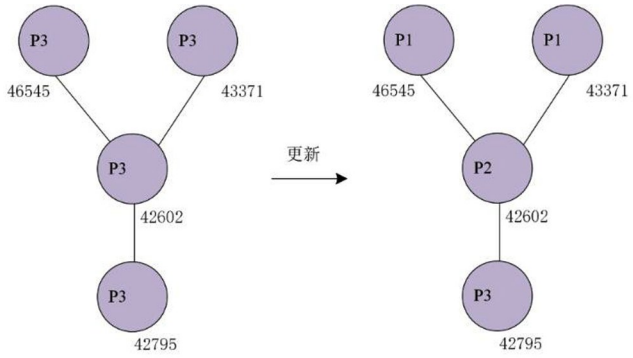

图 2　缺陷依赖树

Fig.2　Defect dependency tree

2.2　跨组件切换成本

若将涉及不同文件的多个缺陷报告分派给同一开发人员,会增加其处理这些孤立缺陷的认知负担.相较而言,开发人员更倾向于处理相互依赖的缺陷,这样他们可以专注于同一组文件,而无须在多个不相关的缺陷报告之间频繁切换.例如,Bugzilla 上报告的

4 个来自 Eclipse Birt 项目的缺陷报告列表,如表 3 所示.这些缺陷报告在两天内被提交.通过查看 GitHub 上的漏洞描述和解决方案,这些缺陷报告均与软件的核心组件或模块密切相关,并且需要检查几乎相同的文件或目录才能完成定位和修复.这表明,考虑缺陷报告之间的关联性在缺陷分派中具有重要意义,可有效降低开发人员的认知负担并提升修复效率.

表 3 缺陷报告文件位置

Table 3 Defect report file location

缺陷报告 ID	摘要	时间	文件
Bug 66733	org.eelipse.birt.core.utilIOUtil doesn′t cheek EOF	2019-01-05	core/org.eelipse birt.core/sre/org/eclipse/birveore! . . /l0Util.java
Bug 66735	Optimize the performance of ULocale.forLocale	2019-01-05	core/org.eelipse,birt.core/sre/org/eclipse/birteore,. . /Local Util,java
Bug 66740	Missing default value in initializing seriptContext	2019-01-05	co re/org.eelipse,birt.core/sre/org/eclipse/birt/core! . . /SeriptContext,java
Bug 66857	BirtDateTime function in chart′s onRender funetioncauses render failure	2019-01-06	core/org.ee lipse,birt.core/src/org/ec lipse/birt/eore! . . /CategoryW′rapper,java

产生的认知时间成本是指从一个任务切换到另一个任务时所需的额外时间和资源.当开发人员跨组件修复缺陷时,不仅要在不同模块间切换任务,还需要重新理解新组件的代码结构和逻辑.各个组件可能具有不同的架构和编码风格,开发人员需要花时间学习这些细节.另外,组件间的跨切换可能还涉及与新组件相关的团队成员的沟通和协作,进一步增加了认知和沟通的成本.因此,跨组件的切换通常会导致开发人员花费更多的时间和精力,影响修复效率和准确性.

相比之下,在同一组件内处理多个缺陷时,认知时间成本较低.开发人员对组件内的信息更加熟悉,能够快速识别和定位问题,投入大量精力去理解代码或系统背景.例如,开发人员可能已了解该组件的常见缺陷模式,或者与该组件的相关团队成员有较好的协作关系.因此,同一组件内的切换成本较低,开发人员能够更高效地修复缺陷.

在修复缺陷的过程中,一个缺陷报告并不总是只涉及一种组件,尤其在复杂的系统中,缺陷可能跨多个组件、模块.为了更全面记录缺陷报告的组件信息,本文不仅考虑了缺陷报告初始时所属的组件,还整合了修复过程中涉及的其他组件信息,形成一个更加全面的组件向量表示.再将这些组件信息构建成一个 bug-component 矩阵作为本文使用的缺陷报告组件特征,矩阵的每一行对应一个缺陷报告,每一列对应一个组件,矩阵中的元素表示该缺陷报告是否涉及该组件.为确保特征的可比性,矩阵经过归一化处理,使得不同缺陷报告之间的组件特征具有一致的尺度.

本文使用余弦相似度计算组件特征向量间的相似度,以此量化缺陷报告间的跨组件关系,进而衡量开发人员修复缺陷报告时的认知时间成本.余弦相似度通常用于衡量两个向量之间的相似度,尤其是在高维向量空间中能更好地处理不同维度之间的差异 C_i,C_i 和 C_j 为缺陷报告 i 和缺陷报告 j 其特征向量,如公式(1)所示.

$$Similarity(C_i,C_j)=\frac{C_i\times C_j}{\|C_i\|\|C_j\|}. \tag{1}$$

2.3 NSGA-Ⅱ

在现有的软件缺陷分派研究中,研究者通常只关注孤立缺陷报告的静态文本特征或开发人员的修复特征,直接进行模型训练并实现缺陷分派.然而,本文同时考虑缺陷报告集内的依赖关系和开发人员的修复偏好,目标解决缺陷阻塞问题,减少开发人员的认知时间成本,以此优化阻塞缺陷的定位与修复时间.通过综合考虑开发人员的修复偏好和认知时间成本,实现更精确地分配修复任务,从而提高整个修复过程的效率和质量.传统的单目标优化方法往往无法平衡多个目标的问题,因此本文选择了多目标优化算法 NSGA-Ⅱ(Non-dominated Sorting Genetic Algorithm Ⅱ)来处理该问题.

NSGA-Ⅱ是一种基于遗传算法的多目标优化方法,旨在寻找一组最优解,称为非支配解,也称 Pareto 解集[37].在多目标优化中,Pareto 最优解是指在目标空间中没有其他解在所有目标上同时优于该解集.NSGA-Ⅱ的核心思想是通过非支配排序和拥挤度距

离来引导解集的搜索,使一群候选解向近最优解演化,使得解决方案在多个目标之间实现良好的平衡并保持解集的多样性,而不会忽略其中任何一个目标.

2.3.1　优化目标

在本文缺陷分派任务中,NSGA-Ⅱ通过对缺陷报告和开发人员之间的分配方案进行优化,以满足以下冲突目标的需求:最小化开发人员上下文切换成本、最大化开发人员修复的匹配度.二者相互制约,相互对抗,因此需要通过 NSGA-Ⅱ在解空间中找到最佳的平衡解.

1)跨组件切换成本

当开发人员需要在多个组件间处理缺陷时,频繁的上下文切换会增加认知负担和修复成本.开发人员处理的缺陷类型越多(如 UI、数据库、后端逻辑),其切换成本就越高.因此本文将上下文切换成本作为优化目标之一,尽量让开发人员能够专注于某一组件或某一类缺陷.T_{switch} 是一个常数,表示每次组件切换所需的基本时间成本.如果缺陷报告的组件特征相似,相似度越高(接近 1),切换时间成本越低,如公式(2)所示.

$$Switching\_Cost(i,j)=\begin{cases}T_{switch}\times(1-similarity(C_i,C_j)), & \text{if}C_i\neq C_j\\ 0, & \text{if}C_i=C_j\end{cases}. \quad (2)$$

2)开发人员匹配度

在缺陷报告分派中,每个开发人员对不同类型的缺陷报告具有不同的经验、能力和偏好,特别是在特定组件或主题上可能表现出更强的优势.例如,一些开发人员擅长处理涉及 UI 模块的问题,而另一些开发人员可能更熟悉底层服务或数据库相关的缺陷.为了保障缺陷报告分派给合适能力的开发人员,本文通过分析开发人员的历史缺陷修复记录提取上下文特征,结合开发人员修复的对应组件属性,通过权重调整突出其在某些模块或主题上的偏好.采用加权匹配方法,将缺陷的关键特征与开发人员的对应能力进行放大和匹配.

此外,本文在优化过程中,加入每位开发人员的工作负载约束,确保高优先级缺陷能够被多个开发人员分配,以避免高优先级缺陷集中分配给少数开发人员,从而导致任务积压.每位开发人员的最大工作负载为 W_{max},每位开发人员 d_i 的工作负载定义为分配其缺陷报告的优先级加权和,$Priority\_Weight(k)$ 是缺陷报告 k 的优先级权重,如公式(3)所示.对于每位开发人员都应该满足 $W_d\leq W_{max}$,避免工作过载.

$$W_d=\sum_{k\in Assigned\_Bugs(d_i)}Priority\_Weight(k). \quad (3)$$

2.3.2　实现过程

NSGA-II 算法通过非支配排序对种群分类,在每次迭代中利用拥挤度比较操作,确保选择的解决方案能够均匀分布在 Pareto 前沿上.通过精英策略,将评估得到的当代的最优个体保留到下一代,从而逐代提高解的质量.其主要流程包括种群初始化、非支配排序、拥挤度计算、个体选择、染色体交叉和变异,并通过精英策略确保优秀个体始终得以保留,算法流程如图 3 所示.

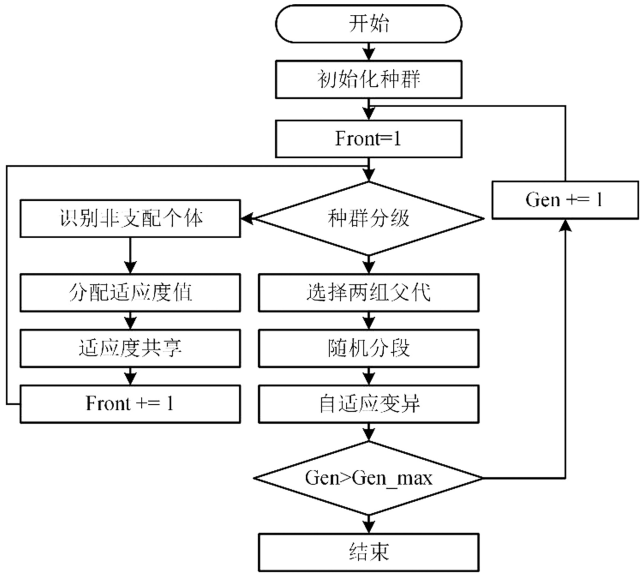

图 3　NSGA-II 算法流程

Fig.3　NSGA-Ⅱ algorithm process

模型以开发人员的历史行为记录、缺陷报告列表中的公共文件及待检查缺陷报告的文本描述信息作为输入,挖掘组件特征的组合关系,并通过文本主题特征扩展开发人员的特征向量,以构建缺陷推荐的目标函数,旨在为每位开发人员生成接近最优的缺陷报告序列.NSGA-II 初始化时,将开发人员的推荐方案视为种群个体,其基因序列长度对应缺陷报告总数,基因值表示推荐给开发人员的缺陷报告编号.为确保种群多样性,基因序列在决策空间 Ω 中随机生成,并设定种群规模 N 以构建初始种群.随后,基于多目标函数进行适应度评估,并通过非支配排序和拥挤距离优先保留高质量的缺陷推荐方案.在交叉操作中,以概率

P_c 随机选择两个父代个体,并在基因序列中随机选取断点交换片段,从而生成新个体,以保留优质推荐信息并探索新方案.变异操作则以较低概率 P_m 随机修改基因值,以增加种群多样性,避免局部最优停滞.

3 实验设置与结果分析

3.1 实验设置

1)实验数据集

本文使用来自大型开源软件项目 Eclipse 和 Mozilla 的缺陷报告,两个项目均采用 Bugzilla 系统管理缺陷信息.本课题选择状态标记为“已验证”和“已修复”的缺陷报告作为实验数据,共提取超过200 000个报告,涉及 2 000 多名开发人员,如表 4 和 5 所示.筛选出拥有初始优先级的缺陷报告作为本文实验的数据.数据中还包括缺陷报告的产品和组件等预定义字段,并将摘要、描述和评论整合为文本信息.

表 4 Eclipse 数据集

Table 4 Eclipse dataset

产品	组件数量/个	缺陷报告数量/个
Platform	22	37 775
JDT	6	19 814
CDT	20	9 104
PDE	5	8 655

表 5 Mozilla 数据集

Table 5 Mozilla dataset

产品	组件数量/个	缺陷报告数量/个
Core	137	74 292
Firefox	47	69 879
Thunderbird	23	19 237
Bugzilla	21	4 616

2)模型参数设置

种群规模 N 的选择影响解的多样性与计算成本,本文在保证解的多样性的同时,选择 150 作为折中方案.迭代次数为 100 能确保本文模型在保持计算成本可控的同时 Pareto 前沿收敛. 交叉概率 P_c 设为 0.7,以保持搜索能力并避免过早收敛. 变异概率 P_m 设为 0.2,以增强全局搜索能力并维持种群多样性.当连续 10 代 Pareto 解集的变化低于 0.01 时,模型终止.本文模型所设置的参数如表 6 所示.

表 6 参数设置

Table 6 Parameter settings

参数名称	参数设置
种群规模 N	150
迭代次数 Gen_max	100
交叉概率 P_c	0.7
变异概率 P_m	0.2
早停阈值	0.01
耐受代数	10

3)评价指标

本文使用的评价指标为准确率,代表每个缺陷报告都有唯一的真实修复者.推荐的开发人员列表长度为 k 时对应的真实修复者命中率,公式如(4)所示:

$$Accuracy_k = \frac{T_k}{N}. \tag{4}$$

其中,T_k 为推荐开发人员列表长度为 k 时成功命中(推荐者含真实修复者)的缺陷报告数量,N 代表缺陷报告的总数.

3.2 实验问题设计

为解决软件缺陷修复过程中因缺陷阻塞带来的挑战,本文构建了缺陷依赖树,综合考虑缺陷报告的依赖关系、开发人员的认知成本以及修复偏好,利用 NSGA-Ⅱ寻找软件缺陷分派的最优平衡解集.模型旨在优化缺陷分派的准确性,降低因缺陷依赖导致的修复延迟.本文的研究将重点关注以下关键问题:

Q1:与基准方法相比,本文模型在缺陷分派准确率上的表现如何?

为验证本文模型在缺陷分派准确率上的效果,实验采用推荐列表 Top-10 准确率作为评估指标,统计每个缺陷报告的实际修复者是否出现在推荐前十名中.实验中,本文模型分别与四个基准方法进行对比测试:BOW + MNB(Bag-of-Words + Multinomial Naïve Bayes)、DBRNN-A[28]、DABT[24]、GRCNN[38].

Q2:与单目标方法相比,本文模型的效果如何?

为了进一步测试本文模型在缺陷分派任务中的效果,本实验分别计算仅考虑跨组件切换成本(S_Time)和仅考虑开发人员匹配度(S_Match)两种单目标方法下的缺陷分派准确率,并与本文的多目标优化方法进行对比实验.

Q3:构建缺陷依赖树是否能够有效缓解缺陷阻塞问题?

本文将关注部分阻塞缺陷的平均修复时间,通过模型推荐缺陷集并将其合理分派给开发人员,使其按照推荐顺序逐个修复缺陷,以验证本文模型在加速阻塞缺陷修复方面的有效性.为了进行深入分析,本文设计了以下四组实验进行对比验证:

1)本文模型(OURS)

使用多目标优化方法推荐缺陷集,并基于缺陷依赖树调整缺陷的优先级.开发人员优先修复匹配缺陷集中高优先级的缺陷.

2)多目标优化方法(Multi)

仍采用本文的多目标优化方法推荐缺陷集,但不应用缺陷依赖树.即不调整缺陷的优先级,忽视缺陷报告的依赖和阻塞关系,开发人员按初始推荐顺序依次修复缺陷.

3)基于跨组件切换成本的单目标方法(S_Time)

采用"跨组件切换成本"作为单一优化目标,得到开发人员匹配的缺陷集,开发人员仍按推荐顺序修复缺陷.

4)基于开发人员匹配度的单目标方法(S_Match)

采用"开发人员匹配度"作为单一优化目标,得到开发人员匹配的缺陷集,开发人员仍按推荐顺序修复缺陷.

实验过程中,开发人员以修复单个缺陷的平均修复时间 T 为单位,按序修复特定阻塞缺陷,以此对比不同方法下阻塞缺陷的平均修复时间,验证本文模型在解决缺陷阻塞问题上的有效性.

3.3　实验结果分析

对于 Q1,在 Eclipse 和 Mozilla 数据集上,本文对比了不同缺陷分派方法的 TOP-10 推荐准确率,以评估本文模型在缺陷分派任务中的表现.实验结果表明,传统方法(BOW+MNB)在准确率上表现较差,而深度学习方法(DBRNN-A、DABT、GRCNN)在一定程度上提升了推荐质量,但仍存在优化空间.本文提出的多目标优化模型(OURS)在 Eclipse 和 Mozilla 数据集上准确率分别达到了 87.2% 和 75.7%,明显优于 DABT 和 DBRNN-A,并与当前最优方法 GRCNN 接近,展现了较强的竞争力.特别是在 Mozilla 数据集上,OURS 方法的准确率比 DABT 高 10.7%,比传统方法高 34.1%,验证了该模型在不同数据集上的适应性和稳定性.具体结果如图 4 所示.

对于 Q2,为了验证多目标优化策略的有效性,本文通过与单目标优化方法的消融实验对比,分别计算了仅考虑跨组件切换成本的 S_Time 方法和仅考虑开发人员匹配度的 S_Match 方法的 TOP-10 推荐准确率.消融实验结果如图 4 所示,在 Eclipse 和 Mozilla 数据集下 S_Match 方法的准确率分别为 84.8% 和 77.2%,S_Time 方法的准确率仅为 51.9% 和 44.4%.从实验结果可以看出,S_Time 方法的推荐准确率远低于 OURS,表明仅优化开发人员跨组件切换成本不足以提供高质量的推荐.而 S_Match 方法虽然在 Mozilla 数据集上的准确率略高于 OURS,但在 Eclipse 数据集上仍略逊于 OURS.这表明,仅考虑开发人员匹配度可以在特定场景下提高推荐效果,但结合开发人员匹配度后能够提供更全面的优化.因此,本文模型通过权衡开发人员匹配度与跨组件切换成本,提高了缺陷分派的合理性.

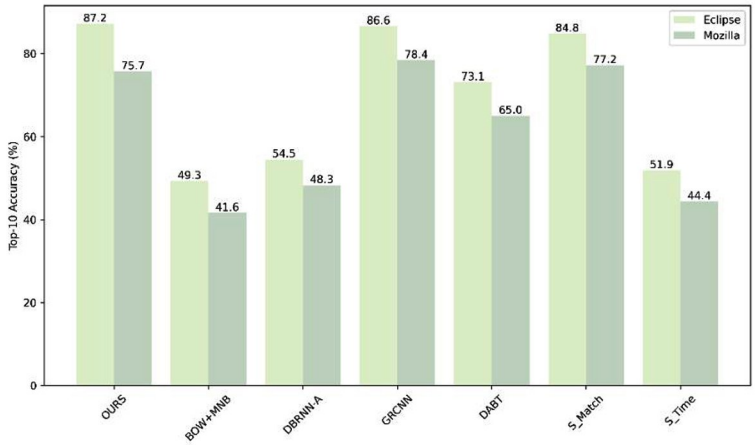

图 4　Top-10 准确率均值对比

Fig.4　Comparison of the mean accuracy of Top-10

对于 Q3,为了评估缺陷依赖树在解决缺陷阻塞方面的有效性,本文通过消融实验,对比未考虑缺陷依赖树的多目标优化方法(Multi)、仅考虑开发人员匹配度的方法(S_Match)和仅考虑跨组件切换成本方法(S_Time)分别在部分阻塞缺陷的平均修复时间上的表现.实验结果表明,OURS 方法在 Eclipse 和 Mozilla 数据集上的平均修复时间分别为 1.3 T 和 1.7 T,显著低于 Multi 方法的 2.8 T 和 3.6 T,也明显优于 S_Time 和 S_Match 两种单目标优化方法的平均修复时间.OURS 方法通过结合缺陷依赖树,更新缺陷修复优先级,调节缺陷修复次序,有效降低了缺陷阻塞对修复效率的影响,验证了缺陷依赖树在优化修复效率方面的有效性.此外,S_Time 方法的缩短平均修复时间效果也优于 Multi 方法和 S_Match 方法,说明优化跨组件切换成本可以使开发人员更多地修复到具有依赖关系的缺陷,也进一步验证了开发人员更适合处理同一文件或同一组件下的缺陷.实验具体结果如图 5 所示.

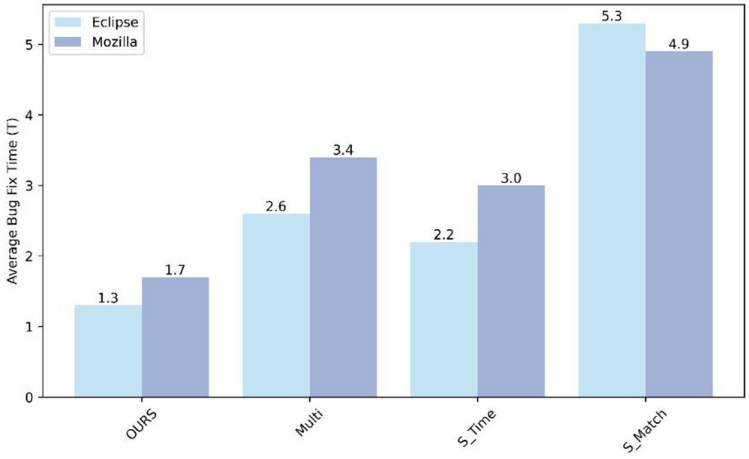

图 5 缺陷的平均修复时间

Fig.5 Mean time to fix bugs

4 结论

当前的软件缺陷分派研究通常孤立地将缺陷报告分派给开发人员,导致开发人员在修复过程中往往需要跨越多个文件和组件,增加了修复成本.此外,传统方法忽视了缺陷间的依赖和阻塞问题.因此,本文提出了一种依赖关联的多目标缺陷分派优化模型,通过构建缺陷依赖树并调整缺陷报告的修复优先级,调节阻塞缺陷的修复次序,加速整体修复进程.同时,本文使用 NSGA-Ⅱ,在开发人员认知成本和修复偏好之间实现优化平衡,从而向开发人员推荐一组经过排序的最佳缺陷报告集合.实验结果表明,该方法在 Eclipse 和 Mozilla 数据集上的部分阻塞缺陷修复平均时间上取得了明显的效果,能够显著减少开发人员在修复阻塞缺陷时的时间消耗.

未来的研究将进一步扩展到更多的开源项目数据集,以提升模型的学习和预测能力.同时,也将重点关注模型的泛化能力,计划在更多不同类型的工业和开源项目中进行测试,验证其适用性和效果.

参考文献

[1] CUI M T,ZHANG Y C.Application of Markov chain approach for multi-attributes dynamic software reliability assessment under both AHP and gray correlation methods[J].International Journal of Modern Physics C,2018,29(5):1840008.

[2] ANVIK J.Automating bug report assignment[C]//Proceedings of the 28th International Conference on Software Engineering.Shanghai:ACM,2006:937-940.

[3] ZHANG J,WANG X Y,HAO D,et al.A survey on bug-report analysis[J].Science China Information Sciences,2015,58(2):1-24.

[4] ASLAM UDDIN K M,SATU M S,RIYAD M M H,et al.DevSched:An efficient bug-triaging model for allocating and balancing developer tasks[J].Iran Journal of Computer Science,2024,7(1):1-11.

[5] JAHANSHAHI H,CEVIK M,MOUSAVI K,et al.ADPTriage:Approximate dynamic programming for bug triage[J].IEEE Transactions on Software Engineering,2023,49(10):4594-4609.

[6] ZENG L,ZHANG Y H.An approach for assigning bug reports based on developer activities[C]//ICETIS 2022 7th International Conference on Electronic Technology and Information Science.VDE,2022:1-5.

[7] ZHENG A X,JORDAN M I,LIBLIT B,et al.Statistical debugging:Simultaneous identification of multiple bugs[C]//Proceedings of the 23rd

International Conference on Machine Learning - ICML '06. Pittsburgh, Pennsylvania: ACM, 2006: 1105-1112.

[8] LI Z M, TAN L, WANG X H, et al. Have things changed now?: An empirical study of bug characteristics in modern open source software [C]//Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability. San Jose California: ACM, 2006: 25-33.

[9] CANFORA G, CECCARELLI M, CERULO L, et al. How long does a bug survive? an empirical study [C]//2011 18th Working Conference on Reverse Engineering. Limerick: IEEE, 2011: 191-200.

[10] REN H, LI Y H, CHEN l. An empirical study on critical blocking bugs [C]//Proceedings of the 28th International Conference on Program Comprehension. Seoul Republic of Korea: ACM, 2020: 72-82.

[11] XUAN J F, JIANG H, HU Y, et al. Towards effective bug triage with software data reduction techniques [J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(1): 264-280.

[12] JONSSON L, BORG M, BROMAN D, et al. Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts [J]. Empirical Software Engineering, 2016, 21(4): 1533-1578.

[13] 刘海洋,马于涛. 一种针对软件缺陷自动分派的开发者推荐方法[J]. 小型微型计算机系统, 2017, 38(12): 2747-2753.

[14] 崔梦天,杨善矿,袁启航. 基于 LDA-BERT 重复缺陷报告检测模型研究[J]. 西南民族大学学报(自然科学版), 2023, 49(4): 414-423.

[15] BHATTACHARYA P, NEAMTIU i. Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging [C]//2010 IEEE International Conference on software maintenance. Romania: IEEE, 2010: 1-10.

[16] BHATTACHARYA P, NEAMTIU I, SHELTON C R. Automated, highly-accurate, bug assignment using machine learning and tossing graphs [J]. Journal of Systems and Software, 2012, 85(10): 2275-2292.

[17] 吴克奇,崔梦天,Mariani Manuel Sebastian,等. 面向软件缺陷数据的协同过滤抽样推荐算法[J]. 西南师范大学学报(自然科学版), 2021, 46(11): 46-55.

[18] XI S Q, YAO Y, XU F, et al. Bug triaging approach based on recurrent neural networks [J]. Journal of Software, 2018, 29(8): 2322-2335.

[19] XI S Q, YAO Y, XIAO X S, et al. Bug triaging based on tossing sequence modeling [J]. Journal of Computer Science and Technology, 2019, 34(5): 942-956.

[20] ZAIDI S F A, WOO H, LEE C-g. A graph convolution network-based bug triage system to learn heterogeneous graph representation of bug reports [J]. IEEE Access, 2022, 10: 20677-20689.

[21] 李元香,董夏磊,项正龙,等. 基于图卷积神经网络的软件缺陷分派方法[J]. 武汉大学学报(理学版), 2020, 66(3): 244-252.

[22] CUI M T, LONG S L, JIANG Y, et al. Research of software defect prediction model based on complex network and graph neural network [J]. Entropy, 2022, 24(10): 1373.

[23] 崔梦天,陈建英,徐智慧. 基于 TMFG 生成拓扑图的软件缺陷预测图特征选择方法[J]. 西南民族大学学报(自然科学版), 2024, 50(4): 418-427.

[24] JAHANSHAHI H, CHHABRA K, CEVIK M, et al. DABT: A dependency-aware bug triaging method [C]//Evaluation and Assessment in Software Engineering. Trondheim Norway: ACM, 2021: 221-230.

[25] KASHIWA Y, OHIRA M. A release-aware bug triaging method considering developers' bug-fixing loads [J]. IEICE Transactions on Information and Systems, 2020, 103 (2): 348-362.

[26] PARK J W, LEE M W, KIM J, et al. CosTriage: A cost-aware triage algorithm for bug reporting systems [J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2011, 25(1): 139-144.

[27] 崔梦天,吴克奇. 基于特征提取和 Stacking 集成学习的软件缺陷预测[J]. 计算机应用与软件, 2025, 42(1): 25-29+48.

[28] MANI S, SANKARAN A, ARALIKATTE R. DeepTriage: Exploring the effectiveness of deep learning for bug triaging [C]//Proceedings of the ACM India Joint International Conference on Data Science and Management of Data. Kolkata: ACM, 2019: 171-179.

[29] ZAIDI S F A, AWAN F M, LEE M, et al. Applying convolutional neural networks with different word representation techniques to recommend bug fixers [J]. IEEE Access, 2020, 8: 213729-213747.

[30] LIU Y, QI X, ZHANG J, et al. Automatic bug triaging via deep reinforcement learning [J]. Applied Sciences, 2022, 12(7): 3565.

[31] FANG S, TAN Y S, ZHANG T, et al. Effective prediction of bug-fixing priority *via* weighted graph convolutional networks [J]. IEEE Transactions on Reliability, 2021, 70(2): 563-574.

[32] AKBARINASAJI S, KAVAKLIOGLU C, BAŞAR A, et al. Partially observable Markov decision process to generate policies in software defect management [J]. Journal of Systems and Software, 2020, 163: 110518.

[33] DAI J, LI Q S, XUE H, et al. Graph collaborative filtering-based bug triaging [J]. Journal of Systems and Software, 2023, 200: 111667.

[34] ALAZZAM I, ALEROUD A, AL LATIFAH Z, et al. Automatic bug triage in software systems using graph neighborhood relations for feature augmentation [J]. IEEE Transactions on Computational Social Systems, 2020, 7(5): 1288-1303.

[35] SEPAHVAND R, AKBARI R, JAMASB B, et al. Using word embedding and convolution neural network for bug triaging by considering design flaws [J]. Science of Computer Programming, 2023, 228: 102945.

[36] DONG H Z, REN H M, SHI J L, et al. Neighborhood contrastive learning-based graph neural network for bug triaging [J]. Science of Computer Programming, 2024, 235: 103093.

[37] 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.

[38] WU H R, MA Y T, XIANG Z L, et al. A spatial - temporal graph neural network framework for automated software bug triaging [J]. Knowledge-Based Systems, 2022, 241: 108308.

(责任编辑:张阳,殷锋,付强,和力新,肖丽;英文编辑:周序林,郑玉才)