# 基于多目标优化算法 NSGA－II 推荐相似缺陷报告

樊田田[1) ] 许 蕾[1),2)] 陈 林[1),2)]

[1)](南京大学计算机科学与技术系 南京 210023)

[2)](南京大学计算机软件新技术国家重点实验室 南京 210023)

**摘 要** 在软件开发过程中，开发人员会收到用户提交的大量缺陷报告. 若修复缺陷报告中问题涉及到的相同源代码文件数目超过一半，则称这些缺陷报告为相似缺陷报告. 给开发人员推荐相似缺陷报告能够有效节约开发人员修复缺陷的时间. 该文提出一种基于多目标优化算法 NSGA-II 推荐相似缺陷报告的方法，即在推荐尽可能少的相似缺陷报告情况下，使得缺陷报告间的相似度尽可能大. 为此，利用缺陷报告的摘要和描述信息，该文采用 TF-IDF 和 Word Embedding 两种方法，从历史缺陷报告中找出相似的缺陷报告，并采用基于搜索的多目标优化算法 NSGA-II 来保证推荐的相似缺陷报告数目尽可能少. 实验数据集是 6 个开源项目(AspectJ、Birt、Eclipse UI、JDT、SWT 和 Tomcat). 与采用单目标算法相比，该文方法在推荐相似缺陷报告的准确率、平均准确率均值、平均序位倒数均值都有提高，其中，在 Top@1 准确率、平均准确率均值、平均序位倒数均值上分别比 Yang 方法提高 125.5%、67.7%和 62.75%.

**关键词** 相似缺陷报告推荐；多目标优化；空间向量模型；词嵌入模型；NSGA-II 算法；软件工程

**中图法分类号** TP311 **DOI 号** 10.11897/SP.J.1016.2019.02175

# Recommending Similar Bug Reports Based on Multi-Targets Optimization Algorithm NSGA-II

FAN Tian-Tian[1)] XU Lei[1),2)] CHEN Lin[1),2)]

[1)](*Department of Computer Science and Technology, Nanjing University, Nanjing* 210023)

[2)](*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023)

**Abstract** During the process of software development, developers usually receive a large number of bug reports, submitted by users, and they must handle these bug reports. Bug report is a description of the information about the problems encountered by software users. Developers fix bugs according to bug reports. Among them, if the fixing issues in bug reports are concerned with the same source code files (more than half), then these bug reports are called as the similar ones. Recommending similar bug reports for developers is quite important, since it can save a lot of time and improve the efficiency. At present, the existing methods usually recommend similar bug reports based on information retrieval methods, and handle the title and description information of bug reports. Considering the product and component information in the bug report, the similarity between the query bug report and the historical bug report is calculated, and then the recommendation list is returned for developer's reference. And the correlation between the previous and current bug reports is not necessarily typed. This paper proposes a recommendation method for similar bug reports based on a multi-targets optimization algorithm (NSGA-II), and the number of recommendations is as small as possible, with the maximum similarities among bug

**收稿日期**：2017-05-08；**在线出版日期**：2018-01-05. **本课题得到国家“九七三”重点基础研究发展规划项目**(2014CB340702)、**国家自然科学基金**(61272080，91418202，61403187)**资助**. **樊田田**，硕士，中国计算机学会(CCF)学生会员，主要研究领域为缺陷报告分析. E-mail：ftiantianlucky@163.com. **许 蕾**(通信作者)，博士，副教授，中国计算机学会(CCF)会员，主要研究领域为 Web 程序分析测试. E-mail：xlei@nju.edu.cn. **陈 林**，博士，副教授，主要研究领域为程序分析、实证软件工程.

reports. NSGA-II algorithm is a genetic algorithm, based on multi-objective optimization, which is used to solve the shortages of existing methods. The main challenges of using genetic algorithm to solve problems in recommending similar bug reports are: (1) transferring the problem of recommending similar bug reports as the problem using genetic algorithm, which need to consider how to represent individuals and how to form a population; (2) selecting appropriate fitness function to evaluate the quality of individuals in the population, since the fitness function is greater, the individual is better, and the probability of being inherited to the next generation is greater. In order to avoid local optimization, we need to select suitable selection operator, crossover operator and mutation operator to ensure the diversity of population in genetic algorithm. Therefore, based on the summary and descriptions of bug reports, after doing natural language pre-processing, such as participle, stop words, and word roots processing, this paper uses the TF-IDF and Word Embedding models to find out the similar bugs in the history reports. The key technique is using the multi-targets optimization algorithm based on NSGA-II, in order to ensure the recommended number of similar bug reports is minimum. The experimental data set includes six open-source projects, namely AspectJ, Birt, Eclipse UI, JDT, SWT, and Tomcat. Compared with single target algorithms, the accuracy of our method, the mean average precision and mean reciprocal rank have improved, and the accuracy in Top@1, Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) are increased to 125.5%, 67.7% and 62.75%, when comparing with the method of Yang. Finally, this paper puts forward the experimental validity threats, including construction validity, internal validity, external validity, also summarizes the paper, analyzes the advantages and disadvantages of this paper, and puts forward the possible improvements in the future.

**Keywords** similar bug report recommendation; multi-objective optimization; vector space model; word embedding; NSGA-II algorithm; software engineering

## 1 引 言

程序缺陷(Bug)是计算机程序中存在的、会破坏程序正常工作的问题或错误,是对系统所需要实现功能的失效或违背.在软件开发和维护的过程中,程序缺陷很难避免,而且会经常出现.随着软件规模的增长,软件缺陷也会相应增加.

当软件使用者遇到一个缺陷时,通常会撰写一份缺陷报告(bug report)来描述所遇到的问题以及相关信息,并反馈给开发人员.目前有一些专门的缺陷追踪系统来管理缺陷,比如 Bugzilla① 或者 Jira②.对于提交的每一份缺陷报告,开发人员需要耗费大量的时间和精力对其进行处理,尽可能修复缺陷报告中涉及到的程序缺陷.

为了帮助开发者高效地处理缺陷报告,Rocha 等人[1]提出了相似缺陷报告推荐的研究.如果两份缺陷报告是相似的,那么这两份缺陷报告涉及到的源代码文件基本相同;因此,如果开发者知道当前缺陷报告的相似缺陷报告,就可以类似处理,从而提高缺陷报告的分配、修复速率.但是该方法没有充分利用缺陷报告提供的信息,所以在此基础上,Yang 等人[2]提出一种将标准的信息检索技术和 Word Embedding 技术相结合的方法.这些方法均返回给开发人员一个缺陷报告的推荐列表,但是往往推荐列表中的缺陷报告和当前缺陷报告相关程度不是很大,而且数目较多,开发人员需要依次排查推荐列表中缺陷报告所对应的源代码文件,工作量很大.由于推荐精度不够高,现有方法会浪费开发人员的时间,导致修复缺陷效率的低下.

因此,本文提出一种基于多目标优化算法 NSGA-II[3]的相似缺陷报告推荐方法,使推荐列表返回的缺陷报告和当前缺陷报告相关程度最大且推荐列表数目最少.该方法结合了标准的信息检索技术和 Word Embedding 技术,并考虑了缺陷报告的标题和描述信息.另外,还为当前缺陷报告和历史缺陷报告分别建立 TF-IDF 向量,并且训练出当前缺陷报

① https://www.bugzilla.org/
② https://www.atlassian.com/software/jira/

告和历史缺陷报告的 Word Embedding 向量. 在此基础上，分别基于 TF-IDF 向量和 Word Embedding 向量对当前缺陷报告和历史缺陷报告计算余弦相似度，进而将这两种模型的相似度结合起来，得出最终相似度，然后采用多目标优化方法 NSGA-II，在尽可能最大化查询缺陷报告和历史缺陷报告库中缺陷报告之间相关度和尽可能最小化推荐列表数目这两个相互竞争的目标之间找到一个平衡.

实验结果表明：与采用单目标方法推荐的缺陷报告列表相比，使用多目标优化方法推荐的缺陷报告列表在 Top@$k$ 准确率、平均准确率均值、序位倒数均值上都有显著提升.

本文第 2 节介绍相关工作，主要包括相似缺陷报告、目前已有的相似缺陷报告推荐方法以及本文所面临的挑战；第 3 节是论文的主要部分，着重介绍多目标优化算法推荐相似缺陷报告的整体框架、NSGA-II 算法的应用、推荐相似缺陷报告方法表示；第 4 节是本文的实验部分，主要介绍本文实验评价指标以及提出的两个研究问题，根据实验结果对其进行回答，并对实验进行讨论；第 5 节提出本文的实验效度威胁，包括构建效度、内部效度、外部效度；第 6 节总结全文，包括本文方法的优缺点分析和可改进的地方.

## 2　相关工作及面临挑战

### 2.1　相似缺陷报告

相似缺陷报告是指历史缺陷报告库中与查询缺陷相关的、需要处理的源代码文件至少有 50% 是相同的缺陷报告（数字“50%”是文献 [1] 中给出的经验值，并且进行了实证研究，得到了开发人员的确认）. 具体计算方法如下：对于查询缺陷报告 $x$ 和历史缺陷报告 $y$，$F_x$ 和 $F_y$ 分别代表修复 $x$ 和 $y$ 所修改的源代码文件列表，采用如式 (1) 计算 $x$ 和 $y$ 相似的概率 $Mutual(x,y)$[2]：

$$Mutual(x,y)=\frac{|F_x \cap F_y|}{\min(|F_x|,|F_y|)} \quad (1)$$

在式 (1) 中，$|F_x|$ 代表修复查询缺陷报告 $x$ 所修改源代码文件列表的数目，$|F_y|$ 代表修复历史缺陷报告 $y$ 所修改源代码文件列表的数目；$F_x \cap F_y$ 表示修复查询缺陷报告 $x$ 的源代码文件列表和修复历史缺陷报告 $y$ 的源代码文件列表的交集（通过文件列表中源代码文件的绝对路径以及文件名来判断两个源代码文件是否相等），$|F_x \cap F_y|$ 代表交集集合中源代码文件的数目. 本文设置 $Mutual(x, y)$ 的阈值为 0.5，即如果修复两个缺陷所修改的源代码文件超过一半是相同的，那么就认为两个缺陷是相似的，否则认为它们是不相似的.

相似缺陷报告和重复缺陷报告[4-8] 的概念相近，但是它们有所区别：相似缺陷报告的条件比重复缺陷报告的条件宽松，对于一个缺陷报告，通常能在历史缺陷追踪系统中找到一些与其相似的缺陷报告，但不一定能找到与其重复的缺陷报告，即，重复缺陷报告一定是相似缺陷报告，但相似缺陷报告不一定是重复缺陷报告.

例如，图 1 和图 2 是 Eclipse 缺陷报告追踪系统中的两份相似缺陷报告. 尽管它们都是和断点相关的缺陷报告，但是它们不是重复缺陷报告，因为单从语法或语义相似来看，这两份缺陷报告的相似度并不高. 但由于在修复这两份缺陷报告中缺陷时修改的一部分源代码文件是相同的，并且比例超过了 50%，因此它们是相似缺陷报告.

Bug ID: 396934
Title: [debugger] do not send ‘add breakpoint’ command in a synchronous way
Product: LDT
Component: LuaDevelopmentTools
Reported Time: 2012.12.19
Description: This might be a user error, or a “don’t do it that way”, so feel free to close this issue

图 1　Eclipse 缺陷追踪系统中的 Bug 396934

Bug ID: 378535
Title: Path mapping not done for “Run to line” breakpoints
Product: DLTKComponent: Commom-Debug
Reported Time: 2011-08-24
Description: When using “Run to line”, the created breakpoint contains the path for original file without any mapping.
The proposed patch just adds this mapping before creating the breakpoint.

图 2　Eclipse 缺陷追踪系统中的 Bug 378535

### 2.2　相似缺陷报告推荐已有工作

近几年来，现有方法通常基于信息检索方法推荐相似缺陷报告，通过对缺陷报告的摘要和描述信息进行处理，并考虑缺陷报告的产品和组件信息来计算查询缺陷报告和历史缺陷报告的相似度大小，然后返回推荐列表供开发人员参考. 但是返回的推

荐列表数目多，而且排在列表前面的缺陷报告和当前缺陷报告的相关性不一定大.

例如，NextBug 方法考虑缺陷报告的描述信息、标题信息来查找相似的缺陷报告，并使用较为复杂的排序方法来推荐相似缺陷报告；除此之外，该方法还进行实证研究来询问开发人员是否会考虑采用该方法推荐的相似缺陷报告. 另外，Yang 等人提出的推荐相似缺陷报告方法考虑缺陷报告的标题和描述信息，还考虑了缺陷报告的组件和产品信息，并把标准的信息检索技术和 Word Embedding 技术结合起来推荐相似缺陷报告.

### 2.3 多目标优化问题

最优化问题是科学研究中主要问题形式之一，在最优化问题中，如果只有一个目标函数则称该类优化问题为单目标优化问题，如果目标函数数目超过一个并且需要目标函数同时达到最优则称该类最优化问题为多目标优化问题. 对于多目标优化问题，各个目标函数的优化极有可能存在竞争，即无法使各个目标函数同时达到最优，需要对多个目标函数进行折中处理. 即一个解对于某个目标来说可能是较好的，但是对于其它目标来讲可能是最差的，因此，存在一个折中解的集合，称为 Pareto 最优解集或非支配解集[3,9].

在本文中，对于一份查询缺陷报告，与其相似的历史缺陷报告不止一份，所以要推荐多份数量有限的(数目不确定但是尽可能少)并且和查询缺陷报告相似度尽可能大的历史缺陷报告给开发人员. 显然，该问题存在两个相互竞争的目标，即尽可能最大化查询缺陷报告和推荐列表中缺陷报告的相似度，同时尽可能最小化推荐列表中的缺陷报告数目.

为了找到上述两个目标的折中平衡，本文把该问题抽象为多目标优化问题. 其中，最大化查询缺陷报告和推荐列表中缺陷报告的相似度有两个子目标：一是最大化查询缺陷报告和推荐列表中缺陷报告的词法相似度；二是最大化查询缺陷报告和推荐列表中缺陷报告的语义相似度.

### 2.4 面临的挑战

基于搜索的软件工程经常使用遗传算法来解决软件工程领域的优化问题[10]. 许多遗传算法已经被用到软件测试中[11-13]，例如测试用例生成、变异测试、回归测试和可测试性转换上.

目前，推荐相似缺陷报告的问题还尚未见到采用遗传算法来解决的先例. 和本文工作相关的多目标优化算法在缺陷定位上有应用[14]并且在缺陷优化问题[15-16]上也有应用. 当软件工程任务被抽象成搜索问题，通过定义该问题的个体、适应值函数和方法变更操作，就可以采用遗传算法来解决该问题.

使用遗传算法来解决相似缺陷报告推荐所面临的主要困难有：(1) 把推荐相似缺陷报告问题表示成可以采用遗传算法来解决的问题，需要考虑如何表示个体以及个体如何组成种群；(2) 为了评价种群中个体的优劣，需要选取合适的适应值函数，因为个体适应值函数的值越大，个体越好，被遗传到下一代的概率越大；(3) 在遗传算法中，为了避免局部最优，需要选择合适的选择算子、交叉算子、变异算子以保证种群个体的多样化.

为此，本文拟使用 NSGA-II[3] 来解决该问题，NSGA-II 算法是受达尔文理论影响的自然选择算法，该算法是 Srinivas 和 Deb 于 2000 年在 NSGA 算法的基础上提出的.

另外，为了说明采用遗传算法能够更加有效地解决本文所提出的基于多目标优化推荐相似缺陷报告的问题，本文需要对比单目标算法和多目标优化算法之间的差异. 其中，目前流行的单目标算法有 TF-IDF 算法、Word Embedding 算法、TF-IDF 和 Word Embedding 算法的结合、LDA 主题模型等. 单目标算法是使得推荐的相似缺陷报告和当前缺陷报告相关程度最大，从而减少开发人员修复缺陷的时间. 但是单目标算法一次只能针对一个目标(仅保证推荐的缺陷报告和查询缺陷报告相关程度最大)，而现实中需要同时针对两个相互竞争的目标，即“相关程度尽可能大且推荐列表数目尽可能少”.

## 3 采用 NSGA-II 算法推荐相似缺陷报告

本节介绍多目标优化算法在相似缺陷报告推荐中的使用情况：首先介绍本文方法的整体框架，然后将推荐相似缺陷报告形式化为多目标优化问题，再详细介绍多目标优化算法的过程和细节.

### 3.1 方法概览

对于一份新缺陷报告，开发者需要根据其标题和描述信息在搜索空间中查找相似的缺陷报告. 搜索空间不仅关系到最有可能被推荐的缺陷报告数目，也关系到它们被提供给开发者的顺序. 对每一份新缺陷报告，要修复其缺陷所依赖的相关代码文档不止一份. 本文采用多目标优化算法的主要目的是为了权衡两个相互竞争的目标，即：尽可能最大化查询缺陷报告和历史缺陷报告库中缺陷报告推荐列表的相似度，同时尽可能最小化推荐列表中缺陷报告

的数目.

图 3 是本文所提出方法的整体框架. 首先对历史缺陷报告库中的缺陷报告和查询缺陷报告做自然语言处理，即对缺陷报告的标题和描述部分做分词、去停止符、去词根处理；然后对处理好的缺陷报告信息建立 TF-IDF 向量；其次对处理好的缺陷报告构建 Word Embedding 词向量（图 3 中简称为 WE 词向量）；分别对查询缺陷报告和历史缺陷报告库中的缺陷报告采用 TF-IDF 向量、Word Embedding 词向量计算余弦相似度；将查询缺陷报告和历史缺陷报告的相似度输入到 NSGA-II 算法中，并采用遗传算法的 3 个算子（即选择算子、交叉算子、变异算子）对种群个体进行演变，采用多目标优化算法给查询缺陷报告推荐相关度最大而且数目最少的相似缺陷报告列表，最后将相似缺陷报告推荐列表返回给开发人员.

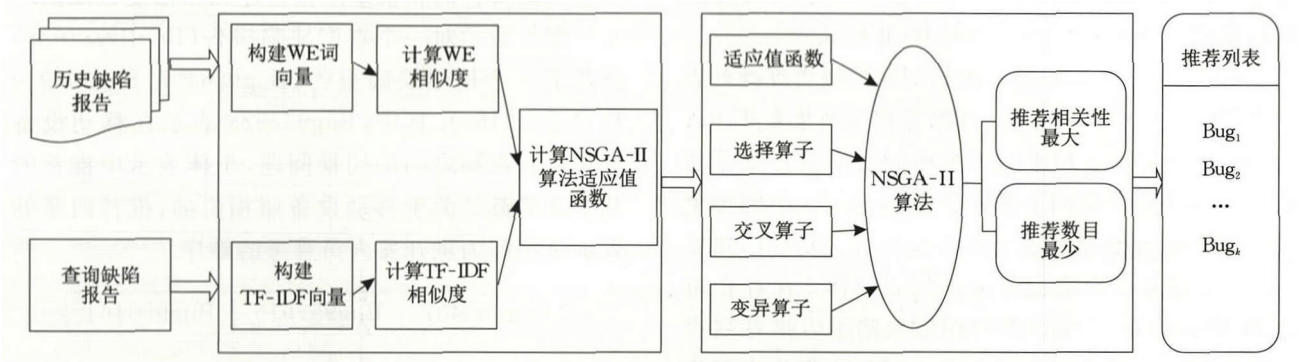


图 3　基于多目标优化算法推荐相似缺陷报告

### 3.2　NSGA-II 算法

如前所述，我们把推荐相似缺陷报告的问题抽象为多目标优化问题，并采用目前流行的多目标遗传算法 NSGA-II（Non-dominated Sorting Genetic Algorithm）来解决此问题.

单目标优化问题通常只存在一个目标函数，只需寻找一个解使该目标函数达到全局最优；而多目标优化问题通常有多个目标函数，多个目标之间会存在竞争，所以使多个目标同时达到最优几乎是不可能的，即一个解对于某些目标可以使其达到最优，但是对于其它目标并非可以使其达到最优，所以对于多目标优化问题，通常对其求得一个解集，解集的特点是无法优化任何一个目标函数，同时也不恶化其它至少一个目标函数，这些解集称为非支配解或者 Pareto 最优解.

NSGA-II 算法详细过程如算法 1 所示：第 1 行表示对查询缺陷报告随机产生相似缺陷报告推荐列表初始种群 $P_0$，第 2 行表示非支配排序后通过遗传算法的 3 个算子（即选择算子、交叉算子、变异算子 3 个基本操作）进行变更操作得到查询缺陷报告的第一代子种群 $Q_0$，第 5 行表示将查询缺陷报告的父代种群 $P_t$ 与其子代种群 $Q_t$ 合并得到大小为 $N$ 的初始化种群 $R_t$，对于输入的一份缺陷报告的标题和描述，该种群代表一系列将要被查看的候选缺陷报告排列；第 6 行对查询缺陷报告所包括的 $N$ 个相似缺陷报告推荐列表个体的种群进行快速非支配排序，在第 9 行对查询缺陷报告的每个非支配层中的相似缺陷报告推荐列表个体进行拥挤度计算，第 10 行选择合适个体 $F_i$ 组成新父种群 $P_{t+1}$，第 13，14 行根据查询缺陷报告的非支配关系及相似缺陷报告推荐列表个体的拥挤度选取合适的个体组成新的父代种群，第 15 行对查询缺陷报告通过遗传算法的基本变更操作产生新的子代种群；以此类推，直到满足第 4 行程序结束的条件（即遗传进化代数），退出循环、结束程序.

**算法 1.**　NSGA-II 算法伪代码

Input：Bug ID，Fitness function，Population size

Output：Pareto bug report list

1. create an initial population $P_0$
2. create an offspring population $Q_0$
3. $t = 0$
4. WHILE stopping criteria not reached DO
5. $R_t = P_t \cup Q_t$
6. $F$ = fast-non-dominated-sort( $R_t$ )
7. $P_{t+1} = \varnothing$ and $i = 1$
8. WHILE $|P_{t+1}| + |F_i| \leq N$ do
9. apply crowding-distance-assignment ($F_i$)
10. $P_{t+1} = P_{t+1} \cup F_i$
11. $i = i + 1$
12. END WHILE
13. sort ($F_i$, $\prec n$)
14. $P_{t+1} = P_{t+1} \cup F_i[N - |P_{t+1}|]$
15. $Q_{t+1}$ = create-new pop($P_{t+1}$)
16. $t = t + 1$
17. END WHILE

NSGA-II 算法中通过快速非支配排序算法和拥挤度的计算来保证种群多样性，同时也可以折中

本文提出的两个相互竞争的目标.

快速非支配排序算法[3,17-18]为:假设查询缺陷报告的相似缺陷报告推荐列表种群为 $P$,则快速非支配排序算法需要计算种群 $P$ 中每个相似缺陷报告推荐列表个体 $p$ 的两个参数 $n_p$ 和 $s_p$,其中 $n_p$ 为种群中支配个体 $p$ 的个体数,$s_p$ 为查询缺陷报告推荐相似缺陷报告列表种群中被个体 $p$ 支配的个体集合.对查询缺陷报告的相似缺陷报告推荐列表种群进行快速非支配排序的具体操作如下:

(1)找到查询缺陷报告的相似缺陷报告推荐列表种群中所有 $n_p=0$ 的个体,并保存在当前集合 $F_l$ 中;(2)对于当前集合 $F_l$ 中的每个相似缺陷报告推荐列表个体 $i$,其所支配的个体集合为 $s_i$,遍历 $s_i$ 中的每个相似缺陷报告推荐列表个体 $l$,执行 $n_l=n_l-1$,如果 $n_l=0$,则将相似缺陷报告推荐列表个体 $l$ 保存在集合 $H$ 中;(3)记 $F_l$ 中得到的相似缺陷报告推荐列表个体为第一个非支配层的个体,并以 $H$ 作为当前集合,重复上述操作,直到查询缺陷报告的整个种群被分级.

通过对查询缺陷报告的种群进行非支配排序操作,对种群中的个体分级,分级之后需要对每个非支配排序中的相似缺陷报告推荐列表个体进行拥挤度计算,然后根据非支配关系以及相似缺陷报告推荐列表个体的拥挤度选择合适的个体,组成新的父代种群.拥挤度是指查询缺陷报告种群中给定相似缺陷报告推荐列表个体的周围个体的密度,直观上可表示为个体.拥挤度算法具体操作如下:(1)令拥挤度 $n_d=0$,总共有 $N$ 个个体,每个个体用 $n=1,2,\cdots,N$ 表示;(2)对于每个目标函数 $f_m$,首先基于该目标函数对查询缺陷报告的种群进行排序,其次令边界的两个相似缺陷报告推荐列表个体拥挤度为无穷,即 $1_d=N_d=\infty$,最后计算其它每个个体的拥挤度为 $n_d=n_d+(f_m(i+1)-f_m(i-1))$,其中 $n$ 取值为 $2,3,\cdots,N-1$.

通过快速非支配排序以及拥挤度的计算之后,查询缺陷报告种群中的每个相似缺陷报告推荐列表个体得到两个属性:非支配序 $n_{rank}$ 和拥挤度 $n_d$.利用这两个属性,可以区分查询缺陷报告种群中任意两个相似缺陷报告推荐列表个体的支配和非支配的关系,进而可以比较两个个体的优劣[3,17-18].如果定义拥挤度比较算子为 $\geq_n$,那么相似缺陷报告推荐列表个体优劣的比较依据为 $i\geq_n j$,即相似缺陷报告推荐列表个体 $i$ 优于个体 $j$,当且仅当 $i_{rank}\leq j_{rank}$ 并且 $i_d>j_d$.

### 3.3 方法表示

#### 3.3.1 个体初始化及编码方式

为了采用 NSGA-II 遗传算法给查询缺陷报告推荐相似缺陷报告,本文用向量来表示遗传个体.对于每一份缺陷报告,个体向量的每个维度表示推荐给查询缺陷报告的相似缺陷报告.因此,每个个体代表了推荐给查询缺陷报告的相似缺陷报告序列,开发者根据推荐的相似缺陷报告序列来修复该缺陷.

图 4 表示对一个查询缺陷报告(BugID937928)推荐了 3 个相似缺陷报告(Bug956407、Bug957910 和 Bug959464).其中,Bug937928 表示在移动设备上相机和视频之间的切换问题,个体表示中推荐的几个缺陷都是关于移动设备照相机的;推荐向量的表示顺序即为供开发人员参考的顺序.

| Bug956407 | Bug957910 | Bug959464 |
|---|---|---|

图 4 个体表示的例子

一般地,对于一份缺陷报告,其推荐列表个体初始化方式是从历史缺陷报告库中随机选择历史缺陷报告来生成,个体列表长度随机确定.每个个体是由随机数目历史缺陷报告的 ID 号来表示的.由于遗传算法不能直接处理历史缺陷报告的 ID 号,因此需要将历史缺陷报告的 ID 号转换成遗传空间内由基因按一定结构组成的个体,即对个体做编码操作.对相似缺陷报告推荐列表个体进行编码需要满足以下规范①:(1)完备性,问题空间中的所有点(即为候选解,相似缺陷报告推荐列表集合)都能作为遗传空间中的点(染色体)表现;(2)健全性,遗传空间中的染色体能对应所有问题空间中的候选解(相似缺陷报告推荐列表集合);(3)非冗余性,染色体和候选解(相似缺陷报告推荐列表集合)一一对应.在本文中,由于个体是由历史缺陷报告 ID 号构成的,历史缺陷报告 ID 号由整数表示,所以对个体进行编码采用的方式是二进制编码,这是目前遗传算法中最常用的编码方法.

二进制编码是由二进制字符 0 和 1 产生的字符串,用来表示问题空间的候选解.本文中将历史缺陷报告 ID 用二进制编码来表示,因为历史缺陷报告 ID 号在不同项目中的值大小不一致,所以编码长度需要根据实验中具体项目的历史缺陷报告 ID 号的大小来确定.

① http://www.baike.com/wiki/%E9%81%97%E4%BC%A0%E7%AE%97%E6%B3%95

3.3.2　适应值函数

在遗传算法中，适应值函数非常重要. 推荐相似缺陷报告列表的目标是使推荐的相似缺陷报告列表和查询缺陷的相关性尽可能最大，而且列表的数目尽可能最少. 在本文中，缺陷报告相关性由 TF-IDF 相似度(TS)、Word Embedding 词向量相似度(WS)决定. 考虑到 $TS$ 和 $WS$ 是互补的[2]，我们用两者的和来衡量缺陷报告的相似度，两者和越大，缺陷报告的相似度越大，所以采用 $TS$ 和 $WS$ 两者的和作为适应函数，具体表示如下：

$$Fitness = TS + WS \tag{2}$$

余弦相似度：算法中计算相似度采用的是余弦相似度，基于查询缺陷和历史缺陷报告的标题和描述信息建立的向量来计算它们的余弦相似度. 计算公式如下：

$$\mathrm{sim}(r_1, r_2) = \cos(\boldsymbol{r}_1, \boldsymbol{r}_2) = \frac{\boldsymbol{r}_1 \cdot \boldsymbol{r}_2}{|\boldsymbol{r}_1| \cdot |\boldsymbol{r}_2|} \tag{3}$$

其中 $r_1$ 和 $r_2$ 分别代表查询缺陷报告和历史缺陷报告的词列表，$\boldsymbol{r}_1$ 和 $\boldsymbol{r}_2$ 分别代表其所对应的词向量.

$TS$ 是采用 TF-IDF(Term Frequency-Inverse Document Frequency)来对一份缺陷报告向量化，然后采用余弦相似度来计算相似度. TF-IDF 是目前最流行的信息检索技术之一，其主要思想是：如果某个词或短语在一份缺陷报告中出现的频率 $TF$ 高，并且在其它缺陷报告中很少出现，则认为此词或者短语具有很好的区分辨别能力. TF-IDF 的公式表示如下：

$$TF(t,d) = \frac{Num_{t,d}}{Num_d} \tag{4}$$

$$IDF(t) = \log \frac{N}{N_t + 1} \tag{5}$$

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t) \tag{6}$$

其中，$Num_{t,d}$ 表示词语 $t$ 在 bug 报告 $d$ 中出现的次数，$Num_d$ 表示 bug 报告 $d$ 中词语的数目，$N$ 表示语料库中所有 bug 报告的数目，$N_t$ 表示文档中包括词语 $t$ 的文档.

WS 是采用 Word Embedding 来对一份缺陷报告表示向量化，然后采用余弦相似度来对其计算相似度. Word Embedding 模型包括 CBOW 模型和 skip-gram 模型[19-20]. 在本文中，实现 Word Embedding 采用的是 skip-gram 模型，之前有研究表明该模型解决软件工程任务比 CBOW 模型效果要好[21-22]. 在 skip-gram 模型中，给一个字词 $w$，$w$ 的上下文定义为 $C_w$，其目标函数 $J$ 如下所示：

$$J = \sum_{i=1}^{n} \sum_{w_j \in C_{w_i}} \log p(w_j \mid w_i) \tag{7}$$

在式(7)中，$n$ 代表字词序列的长度，$p(w_j \mid w_i)$ 是使用下面的 software 函数定义的条件概率：

$$p(w_j \in C_{w_i} \mid w_i) = \frac{\exp(\boldsymbol{v}_{w_j}^{\mathrm{T}} \upsilon_{w_i})}{\sum_{w \in W} \exp(\boldsymbol{v}_w^{\mathrm{T}} \upsilon_{w_i})} \tag{8}$$

在式(8)中，$\boldsymbol{v}_w$ 是词 $w$ 的向量表示，$W$ 是所有字词的词汇表. 通过训练 wiki-english 语料库，语料库中词汇表的所有词可以表示为 $d$ 维向量，根据经验 $d$ 一般取值为 100. 每个词被表示为词向量，那么一份文档用一个矩阵表示，矩阵每一行表示一个词的向量，不同的文档有不同数目的词，所以不能采用文档矩阵计算文档相似度. 本文中为了把文档矩阵转换成向量，对文档中包含的所有词向量求平均，$n$ 表示文档矩阵的行数，$r_i$ 表示矩阵的行向量，公式表示如下：

$$\upsilon_d = \frac{\sum_i r_i}{n} \tag{9}$$

采用以上公式，可把文档矩阵转换成文档向量，便于采用余弦相似度计算文档余弦相似度.

3.3.3　遗传算法的算子

(1) 选择算子

在基于搜索的遗传算法中，变更操作在搜索空间中扮演着很重要的角色，其主要目的是使搜索空间中尽可能留下好的个体. 我们使用轮盘赌选择[23]又称比例选择算子来对个体进行交叉和变异，其选择算子的基本思想是个体被选中的概率与其适应度函数值成正比.

具体操作如下：根据适应值函数计算出每个个体推荐列表的适应度 $f(i=1,2,\cdots,n)$，$n$ 为种群大小；然后根据每个推荐列表个体的适应度在种群中所占比例计算出每个个体被遗传到下一代群体的概率. 计算公式为如式(10)所示，其中 $f(x_i)$ 表示种群个体 $x_i$ 的适应值：

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)} \tag{10}$$

(2) 交叉算子

每次迭代过程中，选择第 $i$ 次迭代的一半个体，用这些被选择的个体采用交叉算子，得到第 $i+1$ 次迭代的另一半新个体. 我们选择单点交叉算子，可以从选择的两个父代个体 $P_x$ 和 $P_y$ 得到两个后代个体 $P'_x$ 和 $P'_y$.

具体操作如下：随机选择一个位置 $k$，$P_x$ 中的前

$k$ 个缺陷报告成为 $P'_x$ 的前 $k$ 个缺陷报告，$P_y$ 中位置 $k$ 后面的缺陷报告成为 $P'_x$ 从位置 $k$ 开始后面的缺陷报告；同样地，$P_y$ 的前 $k$ 个缺陷报告成为 $P'_y$ 的前 $k$ 个缺陷报告，$P_x$ 中位置 $k$ 后面的缺陷报告成为 $P'_y$ 从位置 $k$ 后面的缺陷报告. 这种交叉算子得到的子代可能包含冗余的缺陷报告，所以需要检测是否包含冗余的缺陷报告，如果包含，需要从历史缺陷报告库中随机选择一份缺陷报告来替换冗余的缺陷报告.

(3) 变异算子

变异算子是对缺陷报告推荐列表个体中某些缺陷报告进行变动，采用变异操作是因为变异操作使遗传算法具有局部的随机搜索能力，而且可以使遗传算法维持种群多样性，避免局部最优.

具体操作如下：对于选择的缺陷报告推荐列表，随机选择向量的一个或更多维度点；对于每个选择的维度点，每个维度对应一个缺陷报告，可以删除该缺陷报告或者被其它缺陷报告替换. 我们使用同样方法来解决变异操作造成的冗余.

#### 3.3.4 数值归一化处理

由于对缺陷报告采用不同方法计算的相似度是在不同量纲上的，所以需要对不同方法得到的相似度值进行规范化处理，通过函数变换将其数值映射到某个数值区间. 数值归一化处理是将数据按比例缩放，使之落入一个特定小的区间.

本文采用如下公式对数值进行归一化处理：

$$N(x)=\frac{x-x_{\min}}{x_{\max}-x_{\min}} \tag{11}$$

其中，$x_{\min}$ 和 $x_{\max}$ 分别代表数据集 $X$ 中的最小和最大的值，而 $x$ 代表数据集 $X$ 中任意一个给定的元素. 特别地，在本文中，$x_{\min}$ 代表一份查询缺陷报告和历史缺陷报告库所有相似度的最小值；相应地，$x_{\max}$ 表示一份查询缺陷报告和历史缺陷报告库所有相似度值的最大值；$x$ 代表每一份查询缺陷和历史缺陷库中每一份缺陷报告的相似度值.

## 4 实验评估

### 4.1 数据集

#### 4.1.1 数据集收集

为了验证实验结果的正确性，本文使用 Ye 等人[24]提供的缺陷定位标准数据集来验证本文提出的基于多目标优化的相似缺陷报告推荐方法.

如表 1 所示，数据集包含了 6 个开源项目的缺陷报告：AspectJ 是 Eclipse 平台下的一个子项目，扩展 Java 语言的面向切面编程；Birt 基于 Eclipse 平台，是一个为 Web 应用程序开发的、基于 Eclipse 的开源报表系统；JDT 是一套针对 Eclipse 平台的 Java 开发框架；SWT 是一个 Java 平台下的开源 GUI 编程框架；Eclipse UI 为 Eclipse 的用户界面提供一个基本的构建模块；Tomcat 实现了 J2EE 标准化，是免费开放源码的 Web 应用轻量级服务器. 这六个项目都使用 Bugzilla 作为它们的缺陷追踪系统，并使用 Git 管理软件的开发过程，早期的软件版本已经从 CVS/SVN 转移到 Git. 该数据集总共收集了这 6 个开源项目的 22747 份被标记为 resolved fixed、verified fixed、closed fixed 的缺陷报告，所收集数据集的每份缺陷报告包括的信息有：缺陷报告 ID、标题、描述、报告提交时间、报告者、修复者、状态、产品、组件、优先级、修复缺陷提交的提交版本号、提交日志信息、提交所修复每个 Java 文件的绝对路径. 之所以需要从Github中提取每份缺陷报告的提交信息，是因为要根据缺陷报告和提交日志文件列表判断两份缺陷报告是否是相似的.

表 1 实验数据集

| 项目 | 开发周期 | Bug报告数目 | 每份Bug报告相关文件数目 |
|---|---|---|---|
| AspectJ | 2002—2014 | 593 | 2 |
| Birt | 2005—2013 | 4178 | 1 |
| JDT | 2001—2014 | 6274 | 2 |
| SWT | 2002—2014 | 4151 | 3 |
| Eclipse UI | 2001—2014 | 6495 | 2 |
| Tomcat | 2002—2014 | 1056 | 1 |

#### 4.1.2 数据集处理

因为本文根据查询缺陷报告和历史缺陷报告的标题和描述信息来为查询缺陷报告推荐相似缺陷报告，所以本文采用自然语言处理方法来对缺陷报告的标题和描述信息做文本处理. 具体的自然语言处理操作如下：分词、去停止符、词根还原、拆分复合词. 其中，去停止符操作是去掉自然语言中常见的、如连接词和限定词之类的停止符，英文常见的停止符有“a”、“the”；词根还原操作采用 Porter Stemmer 方法为列表中所有的词取词根，例如“except”和“exception”两个词含义相同，对“exception”做取词根操作变为“except”；因为缺陷报告中的标题和描述部分会有编程语言，所以有些词是使用驼峰命名法的复合词，需要对其做拆分复合词操作，例如“HelloWorld”拆分为“Hello”和“World”.

由于缺陷追踪系统不记录缺陷发生在源代码中的哪些位置，需要找出和每个缺陷相关的源代码文件. 可以根据项目在用 git 提交中产生的提交日志

来获取，用代码快照对应的提交标号将缺陷报告和对应的代码快照关联. 根据 git diff 命令找出该代码快照和前一个代码快照所修改的源代码文件，这些源代码文件即为修改该缺陷报告所关联的源代码文件.

#### 4.1.3　相似缺陷报告基准数据集

为了构建本文实验中的相似缺陷报告基准数据集，本文采用 Rocha 等人[1]研究中所提出的判断相似缺陷报告方法. 相似缺陷报告是指历史缺陷报告库中与查询缺陷相关的、需要处理的源代码文件至少有 50%是相同的缺陷报告. 历史缺陷报告数据集中有每份缺陷报告修复时所修改的源代码文件，采用上述 2.1 节中方法可以找出每份历史缺陷报告的相似缺陷报告，建立相似缺陷报告的基准数据集.

### 4.2　研究问题和评价指标

为了评估基于多目标优化方法推荐相似缺陷报告的有效性，我们主要考虑以下研究问题：

*RQ*1：本文提出的基于多目标优化算法 NSGA-II 是否比单目标算法效果好？

*RQ*2：本文所提出的方法是否比已有推荐相似缺陷报告方法好？

为了对实验结果进行评价，我们采用 Top@$k$ 准确率(Accuracy@$k$)、平均准确率均值、序位倒数均值 3 种评价指标来评价本文所提出的基于多目标优化方法推荐相似缺陷报告的有效性，这 3 种评价指标被广泛用来评价软件工程领域中的推荐任务[1,20,24-26]：

(1) Top@$k$ 准确率(Accuracy@$k$). Top@$k$ 准确率表明对推荐相似缺陷报告方法生成的推荐列表前 $k$ 个缺陷报告进行审查时，推荐缺陷报告是相似缺陷报告的概率. 在本文的评估中，对于一个待推荐的缺陷报告，如果缺陷报告方法生成的推荐列表的前 $k$ 个缺陷报告中至少有一个和给定的缺陷报告相似，则认为推荐相似缺陷报告成功. 计算公式如下：

$$\text{Accuracy@}k = \frac{|R_k|}{n} \tag{12}$$

在上述公式中，$n$ 表示评估过程中使用缺陷报告的总数；$|R_k|$ 表示推荐相似缺陷报告方法进行 Top@$k$ 推荐时，$n$ 个缺陷报告中推荐相似缺陷报告成功的数量. Accuracy@$k$ 的值越大，推荐相似缺陷报告方法的性能就越好.

(2) 平均准确率均值(MAP). 平均准确率均值(MAP)被广泛应用于信息检索领域，用来度量推荐方法的性能，平均准确率均值越大，推荐方法的性能越好.

$$\text{Prec@}k = \frac{\sum_{i=1}^{k} IsRelevant(k)}{k} \tag{13}$$

$$AvgP_j = \frac{1}{|K_j|}\sum_{k\in K_j}\{\text{Prec@}k\} \tag{14}$$

$$\text{MAP} = \frac{1}{n}\sum_{j=1}^{n}\text{Avg}P_j \tag{15}$$

在上式中，对于第 $j$ 份查询缺陷报告，推荐方法为其生成推荐列表 $l$；Prec@$k$ 表示推荐列表 $l$ 的前 $k$ 个缺陷报告和查询缺陷报告的相似比例，其中 $IsRelevant(k)$ 为 $l$ 表示推荐列表 $l$ 中第 $k$ 个缺陷报告和查询缺陷报告相关、为 0 表示不相关；$K_j$ 表示推荐列表 $l$ 中所有和查询缺陷报告相似的缺陷报告在推荐列表中位置的集合；MAP 的值越大，推荐相似缺陷报告方法的性能就越好.

(3) 序位倒数均值(MRR). 序位倒数均值根据推荐相似缺陷报告方法生成的推荐列表中第一个和查询缺陷报告相似的缺陷报告在推荐列表中位置的倒数来判断推荐相似缺陷报告方法的性能，序位倒数均值越大，推荐相似缺陷报告方法的性能越好.

$$\text{MRR} = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{first_j} \tag{16}$$

式(16)中，$first_j$ 表示第 $j$ 个缺陷报告对应的推荐列表中第一个和缺陷报告相似的缺陷报告位置. MRR 的值越大，推荐缺陷报告方法的性能越好.

### 4.3　实验结果

#### 4.3.1　*RQ*1 的实验结果

为了回答 *RQ*1，我们使用如节 4.2 所述的 3 大评价指标对多目标优化方法 NSGA-II 推荐相似缺陷报告和单目标算法推荐相似缺陷报告的方法进行对比.

在实验过程中，我们发现：因为多目标优化算法推荐相似缺陷报告返回的结果是一个相似缺陷报告推荐列表的非支配个体集合，而单目标算法推荐相似缺陷报告返回的是一个相似缺陷报告推荐列表. 这导致不能直接对比使用多目标优化算法和单目标算法推荐得到的相似缺陷报告结果. 因此，本文采用多目标优化算法返回的相似缺陷报告推荐列表集合中的最接近 Knee 点[3]的个体作为候选列表(非支配解集中最优的个体)来和单目标算法的候选结果进行对比[14].

实验中采用多目标优化算法的进化次数分别设置为 30 次、50 次和 100 次，实验结果发现在 30 次

以后进化结果基本稳定，所以本文实验中多目标优化算法的进化次数设置为 30 次.

我们在 6 个开源项目上分别对单目标算法（基于 TF-IDF、LDA、Word Embedding 和 TF-IDF＋Word Embedding 的算法）和基于多目标优化的算法 NSGA-II 进行对比，实验结果如图 5 所示.

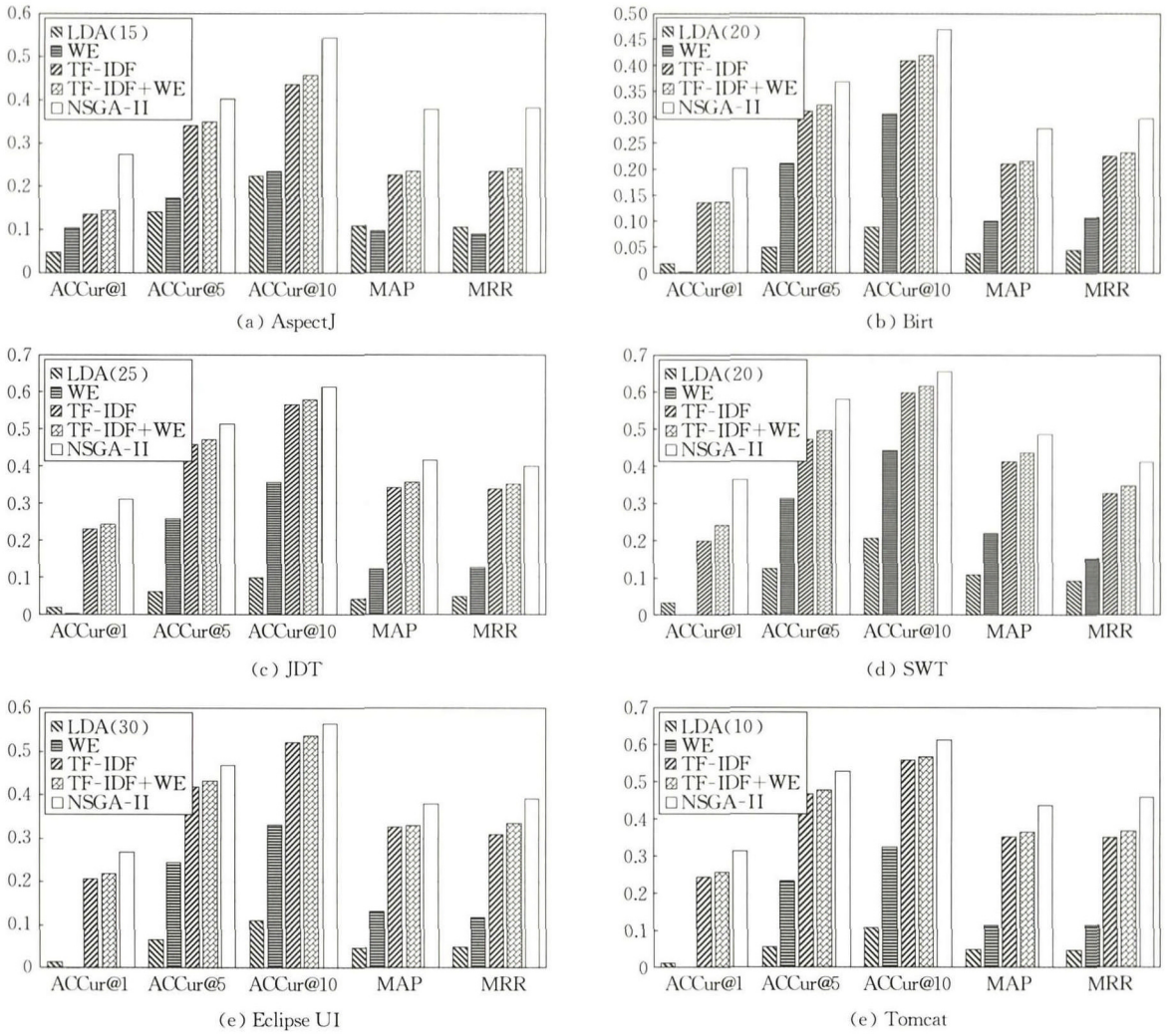


图 5 多目标 *NSGA-II* 算法和单目标算法 TF-IDF、LDA、WE、TF-IDF＋WE 在 AspectJ、Birt、JDT、SWT、Tomcat、Eclipse UI 六个项目上的对比

对图 5 中 AspectJ 项目实验结果进行分析，与 TF-IDF、LDA、Word Embedding 和 TF-IDF＋Word Embedding 相比，我们发现多目标优化算法：(1) 在 Accuracy@1 上分别提高 101.6%、463.2%、159.3% 和 87.7%；(2) 在 Accuracy@5 上分别提高 18.4%、184.7%、132.2% 和 15.0%，在 Accuracy@10 上分别提高 24.4%、140.4%、130.1% 和 18.5%，在 MAP 上分别提高 67.1%、240.8%、285.9% 和 61.0%，在 MRR 上分别提高 62.1%、254.8%、315.0% 和 57.2%.

由图 5 可知，除 AspectJ 项目以外的 5 个项目，其情况也基本类似，即本文所提出来的多目标优化算法和 TF-IDF、LDA、Word Embedding 和 TF-IDF＋Word Embedding 算法相比，在评价指标 Accuracy@1、Accuracy@5、Accuracy@10、MAP 和 MRR 上都有明显提高.

### 4.3.2 RQ2 的实验结果

为了回答 RQ2，我们对本文中所提出的多目标优化算法 NSGA-II 推荐相似缺陷报告方法和已经存在的推荐相似缺陷报告方法进行对比：

(1) NextBug. NextBug[1] 依赖于标准的信息检索技术，对一份预处理的查询缺陷报告 $q$（开发者感兴趣的缺陷报告的描述）和一些缺陷报告集合 $B$（缺陷追踪系统中和 $q$ 有相同组件的缺陷报告集合的描述）中的每一份缺陷报告 $b$ 计算余弦相似度，设置阈值，根据阈值大小对 $q$ 推荐相似缺陷报告 $b$.

在实验中对基于多目标优化算法 NSGA-II 算法和 NextBug 论文[1]中的算法进行对比，实验结果如图 6 所示. 实验结果表明本文方法对评价指标 Accuracy@1、MAP、MRR 提高较明显：NSGA-II 算法比 NextBug 在 6 个项目中 Accuracy@1、MAP 和 MRR 分别平均提高 99. 7%、64. 8%和 60. 7%.

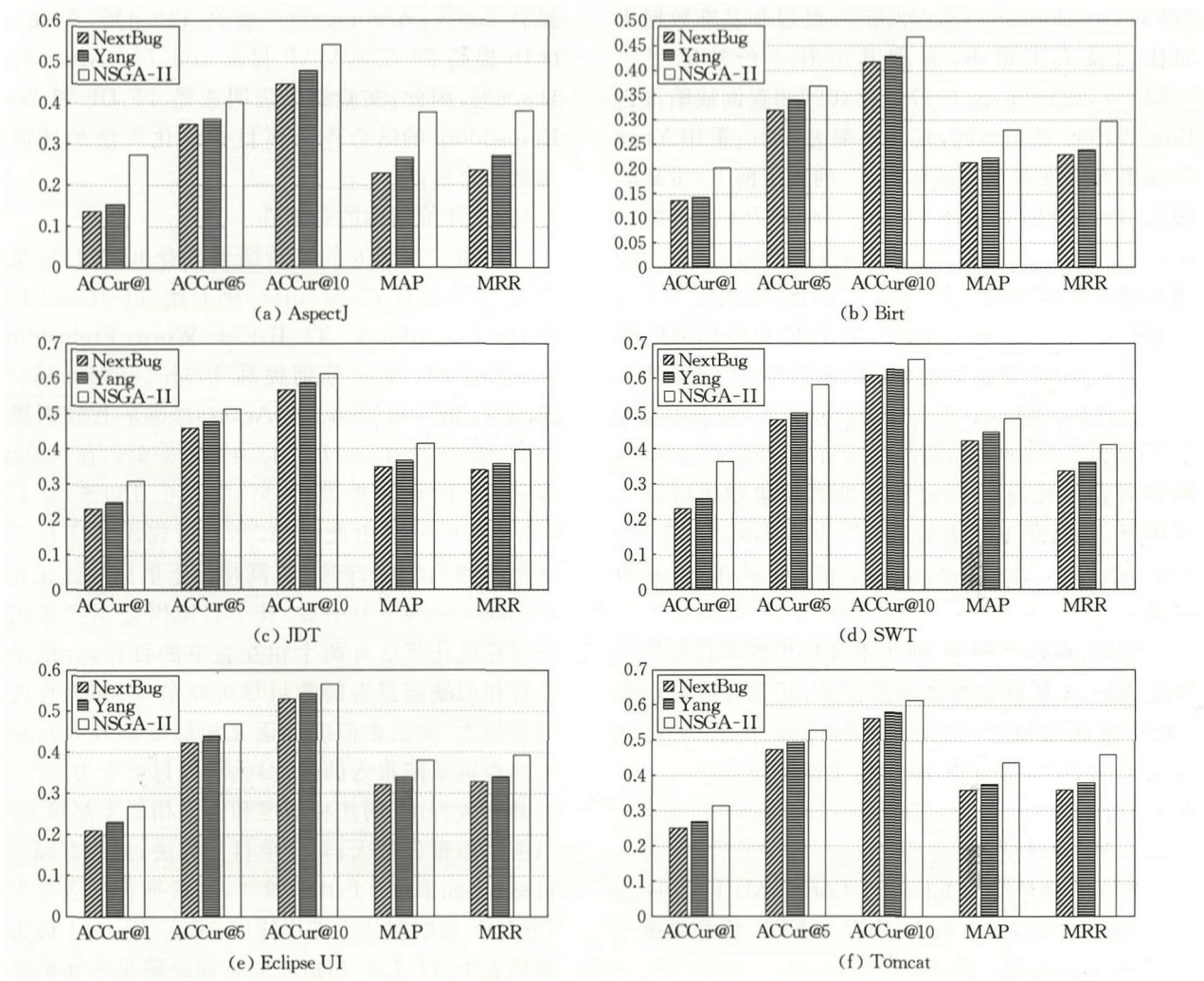


图 6　多目标优化算法 NSGA-II 算法和目前已有方法 NextBug、Yang 在 AspectJ、Birt、JDT、SWT、Eclipse UI、Tomcat 六个项目上进行对比

(2) Yang. Yang 等人[2]论文中所提出的方法结合传统的基于 TF-IDF 的信息检索技术和 Word Embedding 技术，并且考虑缺陷的标题和描述，还有缺陷的产品信息和组件信息. 在实验中对基于多目标优化算法 NSGA-II 算法和 Yang 论文中的算法进行对比，实验结果如图 6 所示. NSGA-II 算法比 Yang 中的方法在 6 个项目上 Accuracy@1、MAP 和 MRR 分别平均提高 125. 5%、67. 7%和 62. 75%.

4. 3. 3　案例分析

对于查询缺陷报告 Bug 399590①，其标题信息为“Bad generics signature generated”，其部分描述信息为“We were using AspectJ version 1. 7. 1 and got hit by this bug today. During investigation we have found that the problem does not occur with AspectJ versions 1. 6. 6—1. 6. 8 but it resurfaces with version 1. 6. 9 and can be reproduced all the way up to the current version 1. 7. 1…”. 对其标题和描述信息做自然语言处理操作(分词、去停止符、取词根)，然后采用本文的多目标优化算法 *NSGA-II* 对其推荐相似缺陷报告. 采用本文 2. 3 节所给方法，得出 Bug 399590 的相似缺陷报告基准数据集为“124 999，39 626，423 257，148 409，124 654，203 367，55 134，91 719”. 经过人工审查，发现这 8 份缺陷报告和 Bug 399590 是相似的，都是关于 Java 中一个很重要的特性——泛型相关的问题.

① http://bugs. eclipse. org/bugs/show_bug. cgi ?id=399590

对于 Bug 399590，采用 NextBug 方法推荐相似缺陷报告，列表中前 10 份缺陷报告 ID 号分别为“423 257，33 474，124 999，153 845，84 333，148 409，124 654，100 260，49 743，203 367”，经过和基准数据集对比以及人工审查，发现其中有 4 份缺陷报告“423 257，124 999，203 367，148 409”和查询缺陷报告 Bug 399590 是相似的，其准确率为 40%；采用 Yang 所提出方法推荐相似缺陷报告，列表中前 10 份缺陷报告 ID 号分别为“318 397，36 564，46 298，423 257，124 654，129 298，100 260，39 626，122 248，55 134”，经过和基准数据集对比以及人工审查，发现其中有 3 份缺陷报告“39 626，124 654，55 134”和查询缺陷报告 Bug 399590 是相似的，其准确率为 30%；采用本文方法对 Bug 399590 推荐相似缺陷报告，列表中对应的ID号为“124 999，423 257，124 654，153 845，203 367”，推荐列表长度为 5，经过和基准数据集对比以及人工审查，发现有 4 份缺陷报告“124 999，423 257，124 654，203 367”和 Bug 399590 相似，其准确率为 80%.

另外，在该案例中，使用本文提出的多目标优化算法 NSGA-II 推荐相似缺陷报告，开发人员需审查 5 份历史缺陷报告；而采用 NextBug 方法和 Yang 论文中的方法，需审查 10 份历史缺陷报告.

### 4.4 讨 论

#### 4.4.1 实验环境

本文实验环境为 Inter(R) Core(TM) i7-4790 八核处理器，主频为 3.6 GHz，内存为 8 GB，操作系统为 64 位 Windows 7.

#### 4.4.2 适应值函数的合理性

本实验将 TF-IDF 和 Word Embedding 方法进行结合[2]，并将其作为多目标优化算法的目标函数. 这是因为研究发现 TF-IDF 是基于信息检索模型的，Word Embedding 是语义模型，这两个模型是互补的. 实验结果表明这两个模型的结合分别比 TF-IDF、Word Embedding 算法 Accuracy@1 提高 7%、30%，MAP 分别提高 4%、60%，MRR 分别提高 3%、60%.

另外，LDA 主题模型是一种能够实现从海量语料库中抽取对象核心语义或者特征的主题建模方法，是一个离散数据集的生成概率模型. 它能将语料库中的每一个文档对应到一组潜在主题的一种概率分布上，而每一个潜在主题又对应了词汇集合上词的一种概率分布. 所以对于 LDA 方法，由于每个项目的语料库不同，因此主题个数也不同(对 AspectJ、Birt、JDT、SWT、Eclipse UI和 Tomcat 项目，LDA 主题个数分别为 15、20、25、20、30 和 10). TF-IDF＋Word Embedding 的实验效果比 LDA 方法的 Accuracy@1 提高 200%，Accuracy@5 提高 140.4%，Accuracy@10 提高 89.5%，MAP 提高 102.7%，MRR 提高 125.6%. 因此，实验结果表明选择 TF-DF 和 Word Embedding 的结合作为多目标优化算法的适应值函数是合适的.

#### 4.4.3 评价指标的合理性

从图 5 和图 6 的实验结果可看出，多目标优化算法 NSGA-II 在 Accuracy@1 比 TF-IDF、LDA、Word Embedding、TF-IDF ＋ Word Embedding、NextBug 和 Yang 分别提高 103%、467%、158%、28.3%、26%和 25%，在 Accuracy@5 上分别提高 17%、182%、131%、11%、12.4%和 8.6%，在 Accuracy@10 上分别提高 25%、140%、130%、8.4%、9.7%和 6.5%. 由此可知，本文所提出的多目标优化算法的 Accuracy@1 提高相比于单目标优化算法在 Accuracy@1 上明显. 其主要原因是本文采用的多目标优化算法有两个相互竞争的目标，分别是使推荐相似缺陷报告的数目尽可能最少、相关程度尽可能最大(词法相似度和语义相似度都较大)，对于某些查询缺陷报告的推荐列表，数目最少为一个，但是其相关程度(词法相似度和语义相似度都较大)和当前缺陷报告最大；对于单目标算法，只能兼顾一个目标，比如采用 TF-IDF 算法，最后推荐列表是根据 TF-IDF 相似度从大到小排序，那么 Top@1 即为推荐列表中 TF-IDF 相似度和查询缺陷报告相似度最大的缺陷报告，根据 TF-IDF 得出的词法相似度，仅仅保证了词法相似度最大，没有考虑语义相似度. 所以，同时考虑词法和语义相似度，会使得实验结果中多目标优化算法的 Accuracy@1 比单目标算法的 Accuracy@1 有明显提高.

另外，既然 Accuracy@1 的实验结果比较好，为什么还需要对比 Accuracy@5 和 Accuracy@10 的实验结果？这是因为如图 5 所示，在针对指标Accuracy@1 的实验结果中，Word Embedding 算法在开源项目 JDT、Birt、Eclipse UI、SWT 和 Tomcat 上的值为 0，表明 Word Embedding 模型对Top@1的准确率提高程度比较低，但是该模型的 Accuracy@5 和 Accuracy@10 有很明显的提高. 因此，我们需要对 Accuracy@5 和 Accuracy@10 分别做实验.

多目标优化算法 NSGA-II 在 MAP 比 TF-IDF、LDA、Word Embedding、TF-IDF＋Word Embedding、

NextBug 和 Yang 分别提高 21.8%、83.4%、66.9%、19.1%、20.8%和 15.7%，在 MAP 分别提高 23.9%、85.4%、69.2%、20%、21.9%和 16.8%. MAP 和 MRR 提高较明显，是因为本文采用的多目标优化算法 NSGA-II 尽可能推荐相似性最大、数目最少的相似缺陷报告，所以推荐的相似缺陷报告尽可能靠前，使 MAP 和 MRR 较大. 因此，本实验结果中 MAP 和 MRR 实验结果提高较明显，也是合理的.

## 5　实验效度威胁

本节对本文实验过程中的影响因素进行分析，主要从实验研究的构建效度、内部效度、外部效度三个方面进行分析. 其中，构建效度表示实验中所选取的数据集对本文实验结果是否有明显的影响以及在未来工作中可改进的地方；内部效度表示实验中各个方法的可比较性以及实验结果的可信性；外部效度则表示本文实验研究结果是否能够推广到其它项目(新的项目或者是一些缺乏历史缺陷报告的项目).

### 5.1　构建效度

本文实验中，为了证实本文所提出的多目标优化算法能有效推荐相似缺陷报告，所选取的数据集是从目前应用最广泛的缺陷追踪系统 Bugzilla 中获得的，并且要求所选取的缺陷报告是已经被修复的且能够从项目版本控制系统的提交日志中获取修复该缺陷报告所关联的源代码文件. 另外，在实验过程中，如果发现一份缺陷报告和项目提交的多个代码快照关联，这表明我们不能确定缺陷报告采用哪个代码快照来确定修复该缺陷报告所修改的源代码文件，这会带来比较大的误差，所以我们认为这类缺陷报告是无效缺陷报告，将去除掉该份缺陷报告.

由于缺陷追踪系统中的缺陷报告没有明确记录修复缺陷报告所修改的源代码文件，尽管在实验过程中可能通过版本控制系统来获得该信息，但是不可能准确获取到修复缺陷报告所修改的源代码文件. 因此，在构建效度上存在基准数据集不是绝对准确的不足，但是我们已经尽最大努力提高基准数据集的准确性.

另外，由于实验数据集所限，本实验对缺陷报告的处理仅考虑了缺陷报告的标题和描述部分，没有考虑缺陷报告的其它信息；其次，本文中采用的多目标优化算法 NSGA-II 适应值函数是 TF-IDF 和 Word Embedding 的结合，本实验在训练 Word Embedding 模型时采用的是 wiki 数据集，wiki 数据集涵盖了各个领域的各种词汇，覆盖面广泛，目前 wiki 数据集在自然语言处理领域中有广泛应用，所以本文也采用 wiki 数据集来训练 Word Embedding 模型.

除此之外，因为缺陷报告中部分是用编程语言描述，编程语言中可能会有一些词是开发人员自己设置的变量、方法名或者类名，导致在采用 wiki 数据集训练的 Word Embedding 中找不到这类词. 简便起见，本文在实验过程中剔除了这类词. 但此操作会在一定程度上降低文本语义相似度，本文中已经采用了 TF-IDF(词法相似度)和 Word Embedding(语义相似度)相结合的适应值函数，可以部分减轻该操作带来的负面影响. 在未来工作中我们会收集和所选项目相关的文档，作为训练 Word Embedding 模型的语料库.

### 5.2　内部效度

本文实验主要针对单目标算法(TF-IDF，Word Embedding，TF-IDF＋Word Embedding，LDA)和多目标优化算法(NSGA-II)，在实验评价指标下进行对比. 由于不同方法的实验结果是在不同量纲上得到的，本文采用数值归一化处理来对每种方法的实验结果进行处理，尽可能让不同方法的实验结果在同一量纲上进行对比.

和其他学者所提出的推荐相似缺陷报告方法相比，因为其他学者没有公开提供出他们做实验所采用的数据集，所以本文所采用的数据集和其他学者方法采用的数据集有所不同，导致可能会在一定程度上和其他学者的实验结果存在差异. 但是，本文实验中针对各种现有方法和本文方法进行对比，都是在本文数据集上完成的实验，本文方法的实验对象一致. 本文对实验结果进行评价所采用的评价指标是该领域普遍采用的.

### 5.3　外部效度

本文实验研究所使用的缺陷报告来自于目前被广泛应用并且有一定规模的 6 个 Java 软件项目，是从 Eclipse 基金会(Eclipse Foundation)以及 Apache 软件基金会(Apache Software Foundation，ASF)下众多优秀的开源项目中选取到的. 这些软件项目的开发过程都比较规范，而且对开发过程及其过程中产生的缺陷都有详细的信息记录.

但是，这些项目都是 Eclipse 项目，并且都是开源的软件项目，所以本文中的方法可能不能完全推广到其它项目或者非开源形式开发的软件项目中去. 在软件工程领域有太多不能被完全控制的因素，这些影响是难以避免的，我们需尽最大努力进行更

加充分和完备的实验准备，以此来减小外部因素的影响.在未来工作中，我们打算收集更多不同类型项目的缺陷报告来验证本文所提出的方法的通用性以及可推广性.

除此之外，本文方法局限使用于历史缺陷报告丰富而且被修复的缺陷报告数量较大的项目，并不适用于新的项目或缺乏历史缺陷报告的项目.要给新的项目或者是缺乏历史缺陷报告的项目推荐相似缺陷报告，可以采用跨项目推荐相似缺陷报告方法，也可以采用已有历史缺陷报告数目充足的项目中的缺陷报告来作为历史缺陷报告，然后采用本文方法来推荐相似缺陷报告.

## 6 总 结

当开发者收到一个新的缺陷，为了帮助开发者更高效地修复缺陷，在重复缺陷的基础上，近几年来研究者提出给开发人员推荐相似缺陷.

本文提出一种基于多目标优化算法 NSGA-II 推荐相似缺陷报告的方法.首先对收集缺陷报告的标题和描述部分进行自然语言处理，然后对历史缺陷报告建立 TF-IDF 模型和 Word Embedding 模型，对每一份新的缺陷报告和历史缺陷报告计算余弦相似度，然后采用 NSGA-II 算法给新的缺陷报告推荐数目少而且和该缺陷报告相关程度最大的缺陷报告，返回一个相似缺陷报告的推荐列表给开发人员.

单目标算法推荐列表采用的方法是仅考虑词与词之间相似度的 TF-IDF 模型、目前比较流行的 LDA 主题模型或者是基于 Word Embedding 语义模型.本文对多目标优化算法 NSGA-II 推荐相似缺陷报告列表和单目标算法推荐列表进行对比，在六个开源项目上做实验，实验结果表明多目标优化算法相比于 TF-IDF、Word Embedding、TF-IDF + Word Embedding、LDA 单目标算法的 Top@K 准确率、平均准确率均值、序位倒数均值都有所提高.

**致 谢** 在此，我们向本文的审稿人和提出宝贵建议的同行表示由衷的感谢！

## 参 考 文 献

[1] Rocha H, Valente M T, Marques-Neto H, et al. An empirical study on recommendations of similar bugs//Proceedings of the International Conference on Software Analysis, Evolution, and Reengineering (SANER). Klagenfurt, Austria, 2016: 46-56

[2] Yang X L, Lo D, Xia X, et al. Combining word embedding with information retrieval to recommend similar bug reports// Proceedings of the International Symposium on Software Reliability Engineering (ISSRE). Changsha, China, 2016: 127-137

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

[4] Runeson P, Alexandersson M, Nyholm O. Detection of duplicatedefect reports using natural language processing// Proceedings of the International Conference on Software Engineering (ICSE). Minneapolis, USA, 2007: 499-510

[5] Wang X Y, Zhang L, Xie T, et al. An approach to detecting duplicate bug reports using natural language and execution information. Proceedings of the International Conference on Software Engineering (ICSE). Vancouver, Canada, 2008: 461-470

[6] Sun C N, Lo D, Wang X Y, et al. A discriminative model approach for accurate duplicate bug report retrieval// Proceedings of the International Conference on Software Engineering (ICSE). Cape Town, South Africa, 2010: 45-54

[7] Sun C, Lo D, Khoo S C, et al. Towards more accurate retrieval of duplicate bug reports//Proceedings of the IEEE International Conference on Automated Software Engineering (ASE). Lawrence, USA, 2011: 253-262

[8] Tian Y, Sun C, Lo D. Improved duplicate bug report identification//Proceedings of the Software Maintenance and Reengineering (CSMR). Szeged, Hungary, 2012: 385-390

[9] Xie Tao, Chen Huo-Wang, Kang Li-Shan. Evolutionary algorithms of multi-objective optimization problems. Chinese Journal of Computers, 2003, 26(8): 997-1003(in Chinese) (谢涛，陈火旺，康立山. 多目标优化的演化算法. 计算机学报，2003, 26(8): 997-1003)

[10] Harman M, Jones S A. Search-based software engineering. Information and Software Technology, 2001(12): 833-839

[11] Núñez A, Merayo M G, Hierons R M, et al. Using genetic algorithms to generate test sequences for complex timed systems. Soft Computing, 2013, 17(2): 301-315

[12] Henard C, Papadakis M, Traon Y L. Mutation-based generation of software product line test configurations//Proceedings of the International Symposium on Search Based Software Engineering (SSBSE). Fortaleza, Brazil, 2014: 92-106

[13] Shelburg J, Kessentini M, Tauritz D R. Regression testing for model transformations: A multi-objective approach// Proceedings of the International Symposium on Search Based Software Engineering (SSBSE). Petersburg, Russia, 2013: 209-223

[14] Almhana R, Mkaouer W, Kessentini M, et al. Recommending relevant classes for bug reports using multi-objective search//Proceedings of the IEEE International Conference on

Automated Software Engineering (ASE). Singapore, 2016: 286-295

[15] Dreyton D, Araújo A A, Dantas A, et al. Search-based bug report prioritization for kate editor bugs repository. Proceedings of the International Symposium on Search Based Software Engineering(SSBSE). Bergamo, Italy, 2015: 295-300

[16] Minelli R, Mocci A, Lanza M. I know what you did last summer: An investigation of how developers spend their time//Proceedings of the International Conference on Program Comprehension (ICPC). Marrakech, Morocco, 2015: 25-35

[17] Zheng Qiang. Elitist nondominated sorting genetic algorithm and its applications [MS dissertation]. Zhejiang University, Hangzhou, 2006(in Chinese)
(郑强. 带精英策略的非支配排序遗传算法的研究与应用[硕士学位论文]. 浙江大学,杭州,2006)

[18] Gao Yuan. Non-dominated sorting genetic algorithm (NSGA) and its applications [MS dissertation]. Zhejiang University, Hangzhou, 2006(in Chinese)
(高媛. 非支配排序遗传算法(NSGA-II)的研究与应用[硕士学位论文]. 浙江大学,杭州,2006)

[19] Mikolov T, Chen K, Corrado G. Efficient estimation of word representations in vector space. Computer Science, 2013(9): 1301-3781

[20] Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality//Proceedings of the Neural Information Processing Systems. Nevada, USA, 2013: 3111-3119

[21] Ye X, Shen H, Ma X, et al. From word embeddings to document similarities for improved information retrieval in software engineering//Proceedings of the International Conference on Software Engineering (ICSE). Texas, USA, 2016: 404-415

[22] Chen C, Gao S, Xing Z. Mining analogical libraries in Q&A discussions—Incorporating relational and categorical knowledge into word embedding//Proceedings of the International Conference on Software Analysis, Evolution, and Reengineering (SANER). Klagenfurt, Austria, 2016: 338-348

[23] Goldberg D E, Deb K. A comparative analysis of selection schemes used in genetic algorithms. Foundations of Genetic Algorithms, 1991(1): 69-93

[24] Ye X, Bunescu R, Liu C. Learning to rank relevant files for bug reports using domain knowledge//Proceedings of the International Symposium on Foundations of Software Engineering (FSE). Hong Kong, China, 2014: 689-699

[25] Zimmermann T, Zeller A, Weissgerber P, et al. Mining version histories to guide software changes. IEEE Transactions on Software Engineering (TSE), 2005, 31(6): 429-445

[26] Zhou J, Zhang H, Lo D. Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports//Proceedings of the IEEE International Conference on Software Engineering (ICSE). Zurich, Switzerland, 2012: 14-24

**FAN Tian-Tian**, MS. Her current research are mainly in bug report analysis.

**XU Lei**, associate professor. Her current research interests are Web app analysis and testing.

**CHEN Lin**, associate professor. His current research interests are program analysis and empirical software engineering.

**Background**

During the process of software development, developers usually receive a large number of bug reports submitted by users. Among them, if fixing issues in bug reports are concerned with the same source code files, these bug reports are called as similar ones. Recommending similar bug reports for developers is quite important, since it can save time for developers. However, the existed methods are usually short of precision, recall and efficiency. Therefore, we plan to improve the performance of recommending similar bug reports by adopting multiple objectives optimization.

The work in this paper is partially supported by the National Basic Research Program (973 Program) of China under grant No. 2014CB340702, the National Natural Science Foundation of China (61272080, 91418202, 61403187).

This paper proposes a recommendation method for similar bug reports based on a multi-targets optimization algorithm (NSGA-II), and the number of recommendations is as small as possible, with the maximum similarities among bug reports. Therefore, based on the abstract and descriptions of bug reports, this paper uses the VSM and Word Embedding models to find out the similar bugs in the history reports, together with the multi-targets optimization algorithm based on NSGA-II, in order to ensure the recommended number of similar bug reports is minimum.