International Journal of Data Warehousing and Mining
Volume 16 • Issue 2 • April-June 2020

Data Mining in Programs:
Clustering Programs Based on Structure Metrics and Execution Values

TianTian Wang, School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
KeChao Wang, School of Information Engineering, Harbin University, Harbin, China
XiaoHong Su, School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
Lin Liu, School of Information Engineering, Harbin University, Harbin, China

ABSTRACT

Software exists in various control systems, such as security-critical systems and so on. Existing program clustering methods are limited in identifying functional equivalent programs with different syntactic representations. To solve this problem, firstly, a clustering method based on structured metric vectors was proposed to quickly identify structurally similar programs from a large number of existing programs. Next, a clustering method based on similar execution value sequences was proposed, to accurately identify the functional equivalent programs with code variations. This approach has been applied in automatic program repair, to identify sample programs from a large pool of template programs. The average purity value is 0.95576 and the average entropy is 0.15497. This means that the clustering partition is consistent with the expected partition.

KEYWORDS

Clustering, Data Mining, Program Repair, Structural Metrics, Value Sequence

INTRODUCTION

Clustering is a widely used data mining technology. A cluster generated by a clustering algorithm is a set of data objects, which are similar to the objects in the same cluster, but different from the objects in other clusters. Many clustering methods have been proposed, such as hierarchical clustering, fuzzy clustering, K-Means clustering etc.

Program clustering refers to the clustering technology applied to the software field. It identifies and groups programs with similar attributes. It is widely used in program analysis and software maintenance, such as comprehension complex software systems (Mahmoud & Niu, 2013; Timothy & Nicolas, 2002), optimizing software architecture (Lung et al. 2006; Demme & Sethumadhavan, 2012), predicting software defects (Seliya & Khoshgoftaar, 2007), software evolution (Scanniello & Marcus, 2011), to improve the quality of software. Because software exists in various control systems, such as fuzzy systems, security-critical systems and so on, program clustering algorithm has received extensive attention.

DOI: 10.4018/IJDWM.2020040104

We tried to apply program clustering in automatic repair of student programs. With the development of MOOCs and online educations, how to provide students with automatic feedback and assistance has become an urgent problem to be solved. Especially for the practical programming courses, the problem is more urgent. In the process of programming practice, students often encounter various errors that make their programs unable to run correctly. However, the current on-line programming system cannot prompt students the specific location of errors or the fixing scheme. Therefore, it is of great value to analyze student programs automatically and provide suggestions for fixing bugs.

Correct template programs written by teachers can provide useful guidance for bug fixing. However, writing templates increases the workload of teachers, and the templates provided may not be comprehensive. In the process of programming exercises or examination, a large number of student programs can be obtained. Many of these programs are correct and can be used as potential template programs. But student programs that implement the same programming task may have similar implementation forms or may be implemented in different ways. This requires analysis of these potential templates, removing redundant similar programs, and retaining multiple different implementations as templates. In addition, in the process of fixing bugs, the template program similar to that of the buggy program, can provide better reference for the revision of the buggy program, and avoid unnecessary modifications. The template programs identification problem can be solved by program clustering.

Most of the existing program clustering methods only analyze the static properties of programs but lack the consideration of execution information. This limits these methods in identifying the functional equivalent programs with code variations. Therefore, the purpose of this paper is to solve this problem and apply program clustering in automatic repairing of programs.

The contributions are as follows. A program clustering method considering both the structural metrics and execution value sequences is proposed. Not only the programs with similar structure can be identified, but also the programs with similar execution behavior can be recognized. By doing this, whether two functional equivalent programs adopt the same algorithm can be judged by the similarity of execution sequences.

RELATED WORK

State of the Art of Automatic Program Repair

At present, the main research object of existing program repairing methods is industrial software (Gazzola et al., 2017). The methods can be divided into the following types: formal specification based methods e.g. AutoFix (Pei et al., 2014); stochastic evolution methods e.g. GenProg (Goues et al., 2012), AE (Weimer et al., 2013), RSRepair (Qi et al., 2014), CapGen (Wen et al., 2018); template-based methods e.g. Prophet (Long & Rinard, 2016); constraint solving and programming synthesis, such as Angelix (Mechtaev & Roychoudhury, 2017), ACS (Xiong et al., 2017), S3 (Le et al., 2017), SemGraft (Mechtaev et al., 2018). However, most of these methods are based on the assumption of "sophisticated programmers", i.e. there are usually a few of simple bugs in software. However, there may be several complex bugs in a student program.

Yi et al. studied the performance of current popular industrial software repair tools in repairing student programs (Yi et al., 2017). Four tools i.e. GenProg, AE, Prophet, Angelix were analyzed. The results showed that the repair rate of these tools is low. The reason is that the types of defects considered by these tools are limited. These tools typically modify one error at a time; however, student programs often have more errors in their programs, requiring more complex modifications. In addition, the test only takes into account the execution results of the program, without considering the behavior of the program and the programming specification, so that the generated patches are likely to sacrifice other important requirements. Therefore, these tools are not suitable for application in student program repair.