Vol.2 Issue.3, Pg.: 136-140
March 2014

INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

CLUSTERING BASED ON SOFTWARE EVOLUTION USING GENETIC ALGORITHM

Christabel Williams[1], Bright Gee Varghese R[2]

[1] PG Student, Department of Computer Science and Engineering, Karunya University, Tamil Nadu,India, **christabel.williams99@gmail.com**
[2] Assistant professor, Department of Computer Science and Engineering, Karunya University, Tamil Nadu,India, **brightfsona@yahoo.co.in**

Abstract

Clustering deals with grouping up of similar objects. Unlike classification, clustering tries to group a set of objects and find whether there is some relationship between the objects whereas in classification a set of predefined classes will be known and it is enough to find which class a object belongs. Simply, classification is a supervised learning technique and clustering is an unsupervised learning technique. Clustering has many applications in various fields. In Software engineering it helps in reverse engineering, software maintenance and for re-building systems. It aims at breaking a larger problem into small pieces of understanding elements. So far many clustering methods were carried out to achieve better clustering result. Clustering techniques apply when there is no class to be predicted but rather when the instances are to be divided into natural groups. These clusters presumably reflect some mechanism at work in the domain from which instances are drawn, a mechanism that causes some instances to bear a stronger resemblance to each other than they do to the remaining instances. It naturally requires different techniques to the classification and association learning methods. This paper aims at including the evolutionary data for carrying out clustering. Evolutionary data is nothing but when two classes are often changed together by a single transaction then the two classes are said to be evolutionarily dependent. By considering these data into account better results can be achieved. Due to its various advantages genetic algorithm is preferred compared to other traditional methods.

Index Terms: Clustering, Classification, Software Engineering, Traditional, Evolutionary, Genetic algorithm.

1. Introduction

Clustering is a technique for finding similarity groups in data, called clusters. It groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters. Clustering is often called an unsupervised learning task as no class values denoting an a priori grouping of the data instances are given, which is the case in supervised learning. Clustering can be used for understanding complex software systems (Mancoridis et al, 1999) , to restructure software architectures (Anquetil et al, 1999), to identify reusable components (Maarek et al, 1991) and also for detecting misplaced software artefacts (Vanya et al, 2008). They can be used in software maintenance and re-use. In order to maintain software, understanding the source code is mandatory. Therefore source code should be clustered. The main purpose is to understand the system, artefacts recovery and identifying the relationships among the source code.

Software clustering is an important discipline in software engineering especially in reverse engineering and software maintenance. Whenever a software is developed and distributed for its applications it should be monitored and maintained. In order to maintain a software every details about that same should be known. For this purpose clustering is carried out. Various factors are taken into account for carrying out clustering. In order to cluster source code various types of dependencies are taken into account.

1.1 Evolutionary data in clustering

Various software clustering algorithm rely on the information about various software artifacts to calculate similarity measure and yield better clustering output. So far various algorithms used only structural code dependencies to carry out clustering. The main objective of our work is to include evolutionary data to achieve better results. Evolution includes various changes underwent by the source code during the phase of development.

Clustering-cantered approaches enrich structural data with some evolutionary aspects: Andritsos and Tzerpos (2005) integrated the file ownership information which is a simple evolutionary data source and found that the results seemed to be much improved comparatively with the structural data. However, they also found by adding the timestamps of the files which is also an evolutionary data that it tends to decrease the clustering quality. Wierda et al, (2006) ,combined the current version of the software system with the initial version and achieved better decomposition. Sindhgatta et al, (2007) introduced a transaction-based clustering approach that is they added some evolutionary transactions to structural transactions and obtained a better clustering result.

1.2 Genetic algorithm

A **genetic algorithm** (or **GA**) is a search technique used in computing to find true or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. One way of understanding intelligence is as the capability of a creature to adapt itself to an ever-changing environment. Adaptation is also change in the *characteristics of a species*, over the generations, in response to environmental change . An individual creature is in *competition* with other individuals of the same species for resources, mates etc. There is also *rivalry* from other species which may be a *direct* (predator) or *indirect* (food, water, land, etc.) threat. In nature, evolution operates on populations of organisms, ensuring by *natural selection* that characteristics that serve the members well tend to be passed on to the next generation, while those that don't die out .
Evolution can be seen as a process leading to the *optimization of a population's ability to survive* and thus reproduce in a specific environment. *Evolutionary fitness* **-** the measure of the ability to respond adequately to the environment, is the quantity that is actually optimized in natural life .Consider a normal population of rabbits. Some rabbits are naturally faster than others. Any characteristic has a *range of variation* that is due to i) sexual reproduction and ii) mutation. We may say that the faster rabbits possess superior fitness, since they have a greater chance of avoiding foxes, surviving and then breeding. If two parents have superior fitness, there is a good chance that a combination of their genes will produce an offspring with even higher fitness. We say that there is *crossover* between the parents' genes. Over successive generations, the entire population of rabbits tends to become faster to meet their environment challenges in the face of foxes. GA uses three main operations to create the next generation from the current population (at each step).

- ***Selection rules*** select the individuals, called *parents* that contribute to the population at the next generation.
- ***Crossover rules*** combine two parents to form children for the next generation.
- ***Mutation rules*** apply random changes to individual parents to form children.

GA is capable of finding solutions for many problems for which no usable algorithmic solutions exist. Optimization searches a solution space consisting of a large number of possible solutions. Continuous improvement is the key idea. The evolution usually starts from a population of randomly generated individuals. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are selected from the current population (based on their fitness), and modified (recombined and possibly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

2. Proposed method:

The proposed method uses genetic algorithm for achieving better results. So far many experiments were carried out and the result seems to be not promising. Previously the work carried out using agglomerative clustering algorithm is not satisfactory since it suffers with many drawbacks. It takes a lot of time in carrying out the clustering process. No provision can be made for a relocation of objects that may have been 'incorrectly' grouped at an early stage. The result should be examined closely to ensure it makes sense. Use of different distance metrics for measuring distances between clusters may generate different results. Performing multiple experiments and comparing the results is recommended to support the veracity of the original results.

The proposed method includes Genetic Algorithm (GA) along with the evolutionary data that is obtained during the process of development of the software system. The following are the steps involved in the proposed method: (i) Class identification ii) Structural and evolutionary dependency extraction iii) Module Dependency graph iv) Clustering

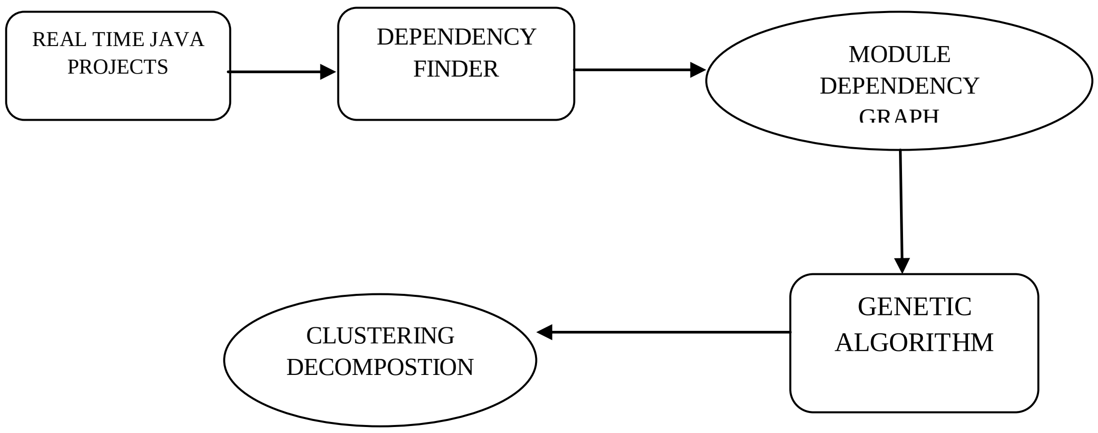


Fig 2.1 Block Diagram

2.1 Class Identification

Class identification is the initial phase in this work where the source package is loaded first and each class in the package is observed. Then each class in the package is analyzed and they are used for extracting the necessary information. Various java projects are taken into consideration and they are analysed for further extraction.

2.2 Structural and evolutionary dependency extraction

The structural dependencies are the most widely used data source for software clustering. They represent the conventional approach and constitute control group in our study. Main types of structural class dependencies namely inheritance, aggregation and usage ie., method calls, method parameters, local variables are incorporated.

These types of dependencies can be defined according to Becket al, (2012) as follows

$$(c1; c2)\ \varepsilon\ E_{CIG} \Leftrightarrow c1 \text{ extends } c2$$
$$(c1; c2)\ \varepsilon\ E_{CAG} \Leftrightarrow c1 \text{ aggregates } c2$$
$$(c1; c2)\ \varepsilon\ E_{CUG} \Leftrightarrow c1 \text{ uses } c2$$
$$E_{SCDG} := E_{CIG} \cup E_{CAG} \cup E_{CUG}$$

Where CIG–Class Inheritance Graph, CAG-Class Aggregation Graph , CUG-Class Usage Graph and SCDG-Structural Code Dependency Graph. These structural code dependencies can be extracted using Dependency Finder which is able to extract all sorts of dependencies.

The evolutionary dependency of a software project is documented by the changes applied to its source files in the course of development. Class A depends on class B by evolution if class A has often been changed together with class B or else both classes have been the part of the same transactions then A and B are said to be evolutionarily dependent. Such types of classes are extracted in this module.

2.3 Module Dependency graph:

Module dependency graph represents modules as nodes and module dependencies as directed edges. They are traced in order to be given as an input for carrying out clustering. Since the terms module and dependency are not bound to a strict definition, we are allowed to consider the weighted dependency graphs.

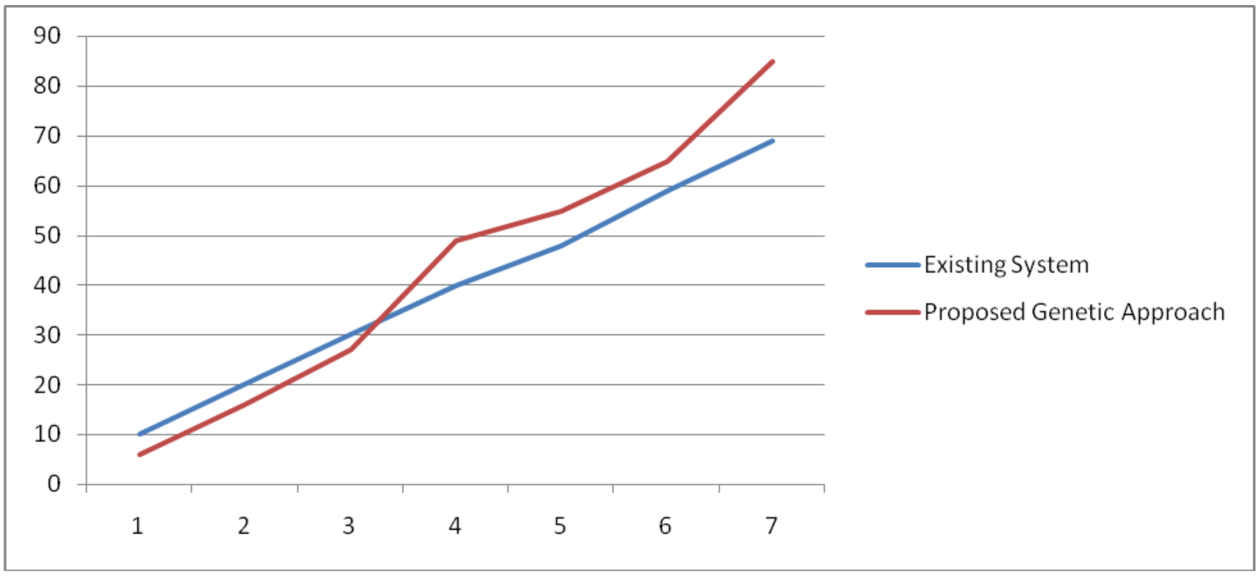
2.4 Clustering

Clustering is the final module in this project. Here clustering is carried out by combining both the structural and evolutionary dependencies between the classes used. Genetic Algorithm is used to carry out clustering. Repeated iterations help us to get a much better results comparatively.

3 Result Analyses

The performance can be seen from the graph and it seemed to be promising. It is plotted for time and accuracy.

Accuracy



4 Conclusions

A clustering exclusively based on evolutionary dependencies, however, is only successful if substantial evolutionary data is available. Evolutionary dependencies often do not cover the set of classes sufficiently. This seems to be the main reason for the better performance of the aggregated structural dependencies. Thus, when clustering a system by only taking evolutionary dependencies into account, it is most important to rely on extensive historical data that covers the essential parts of the system. The main advantages over a purely structural based clustering are that also non-source files can be considered and that the approach works independent of the programming language. An integration of the two data sources unites the advantages of the approaches at the cost of a more complex data acquisition. In addition to a parsed system core, the clustering approach is still able to handle non-source files and non-parsed source files. The clustering quality increases in our experiments, especially when strengthening duplicate dependencies. In this work are four modules which are class identification, structural dependency and evolutionary dependency extraction, module dependency graph and clustering. The full implementation has been carried out and the performance has been analyzed. The result of the implementation shows that by including evolutionary dependency better clustering quality can be achieved.

REFERENCES

Andritsos P, Tzerpos V (2005) Information-theoretic software clustering. IEEE Transactions on Software Engineering 31(2):150-165

Anquetil N, Fourrier C, Lethbridge TC (1999) Experiments with clustering as a software remodularization method. In: WCRE '99: Proceedings of the 6th Working Conference on Reverse Engineering, IEEE Computer Society, Washington, DC, USA, pp 235-255.

Beck F, Diehl S (2010a) Evaluating the impact of software evolution on software clustering. In:WCRE '10: Proceedings of the 17thWorking Conference on Reverse Engineering, IEEE Computer Society, pp 99-108

Maarek YS, Berry DM, Kaiser GE (1991) An information retrieval approach for automatically constructing software libraries. IEEE Transactions on Software Engineering 17(8):800-813.

Mancoridis S, Mitchell BS, Chen Y, Gansner ER (1999) Bunch: A clustering tool for the recovery and maintenance of software system structures. In:ICSM '99: Proceedings of the IEEE International Conference on Software Maintenance, IEEE Computer Society, Washington, DC, USA, pp 50-59

Sindhgatta R, Pooloth K (2007) Identifying software decompositions by applying transaction clustering on source code. In: COMPSAC '07: 31st Annual International Computer Software and Applications Conference - Vol. 1, pp 317-326

Vanya A, Hoand L, Klusener S, van de Laar P, van Vliet H (2008) Assessing software archives with evolutionary clusters. In: ICPC '08: Proceedings of the 16th IEEE International Conference on Program Comprehension, IEEE Computer Society, Los Alamitos, CA, USA, pp 192-201.

Wierda A, Dortmans E, Somers LL (2006) Using version information in architectural clustering - a case study. In: CSMR '06: Proceedings of the Conference on Software Maintenance and Reengineering, IEEE Computer Society, Washington, DC, USA, pp 214-228

BIOGRAPHIES

Christabel Williams is pursuing MTech (Computer Science and Engineering) degree from Karunya University, Tamil Nadu, India. She received her BE (Computer Science and Engineering) degree from C.S.I College of Engineering, Ketti ,The Nilgiris. Tamil Nadu, India.

Bright Gee Varghese R completed his B.E (Computer Science and Engineering) from Sun College of Engineering and Technology, Nagercoil, Tamil Nadu, India in 2003. He started his career as Lecturer in Sun College of Engineering and Technology, Nagercoil, from June 2003. He completed his M.E from Vinayaka Mission university, Salem, Tamil Nadu in 2011.Currently he is working as an Assistant Professor in Computer Science Department in Karunya University and pursuing part time Ph.D in Karunya University, Tamil Nadu, India. His main research area is Software Engineering.