Information Sciences 342 (2016) 96–110

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A similarity-based modularization quality measure for software module clustering problems

Jinhuang Huang, Jing Liu*

Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an 710071, China

ARTICLE INFO

Article history:
Received 10 February 2015
Revised 22 October 2015
Accepted 6 January 2016
Available online 19 January 2016

Keywords:
Software module clustering problem
Modularization quality measure
Reverse engineering
Similarity

ABSTRACT

Software systems tend to become larger and more complicated as the functional requirements increase gradually. Well-modularized software systems are widely believed to be understood, managed and evolved easily. Software module clustering problems (SMCPs) are significant and challenging problems in software engineering whose primary aim is to obtain perfect structures of software systems according to predefined rules. To solve SMCPs, it is important to design a measure which can appropriately evaluate the quality of the structures obtained. So far, the modularization quality (the basic *MQ*) is widely used as the conventional measure. However, this measure does not consider global modules and edge directions between two modules. Therefore, in this paper, we design a new modularization quality measure based on the similarity, labeled as *MS*, to automatically guide optimization algorithms to find a good partition of software systems which considers both global modules and edge directions. To systematically validate the performance of *MS*, three optimization algorithms, namely hill-climbing algorithm (HC), genetic algorithm (GA), and multi-agent evolutionary algorithm (MAEA), are adopted to optimize *MS* on 17 real-world software systems with varying sizes. A thorough comparison shows that *MS* outperforms the basic *MQ* in guiding the optimization algorithms to find more meaningful module clusters for software systems.

1. Introduction and motivation

Software systems develop and change gradually due to the increasingly changes of need in business, government and social cultures. It is a challenging task to understand and maintain large software systems, especially if they are not documented well [4]. Good structures of software systems tend to be easy to be understood, maintained and managed in the software development process [25]. The software module clustering, which can obtain good clusters for software systems according to predefined rules [18], is one of the effective ways to maintain the good structure of software systems.

The software module clustering is an approach that uses Module Dependency Graphs (MDGs) [23] to decompose the structure into several subsystems, which makes the complex software systems be more comprehensible and manageable. MDGs are directed graphs in which modules and their relationships are represented as vertexes and edges between the vertexes, respectively [32]. Using clustering algorithms to create partitioned MDGs is defined as the software module clustering problems (SMCPs) [4,19,20,23,25].

* Corresponding author. Tel.: +86 29 88202661.
E-mail addresses: 287309407@qq.com (J. Huang), neouma@163.com (J. Liu).
URL: http://see.xidian.edu.cn/faculty/liujing/ (J. Liu)

http://dx.doi.org/10.1016/j.ins.2016.01.030
0020-0255/

In general, the process of dealing with SMCPs can be transformed into minimizing the inter-connectivity and maximizing the intra-connectivity. The inter-connectivity means the connections between modules of two different clusters, while the intra-connectivity means the connections between modules in the same cluster [21]. In fact, the inter- and intra-connectivity correspond to coupling and cohesion in the software engineering, respectively, which are two primary principles in the process of software design [34]. SMCPs can be treated as a search-based optimization problem, which was first suggested by Mancoridis et al. [23]. To guide the process of searching, an objective function is needed to evaluate the quality of the clusters obtained. In existing work, the Modularization Quality (the basic *MQ*) [22], which is based on the trade-off between inter- and intra- connectivity, has been widely used as the objective function.

However, due to the large searching space, it is difficult to find the best partition for SMCPs using reasonable computational cost, which had been proved to be NP hard problems by Mitchell in [25]. So far, many algorithms have been proposed to solve SMCPs, such as hill-climbing algorithms, genetic algorithms, multi-objective evolutionary algorithms [3,4,7,13–15,20,24,25,27–30,32,35,37–39].

In 2002, Mitchell first used the hill-climbing approach as the search technique, which could quickly find an acceptable solution for SMCPs [25]. In this algorithm, it begins with a random partition and repeatedly searches better neighboring partitions until the value of *MQ* of the neighbor partition could not be improved. However, one of the most serious problems of this algorithm is it may be trapped in local optima easily and is hard to find the global optima [19]. To overcome this problem, a multiple hill-climbing approach was put forward in [19]. Doval [4] first employed a genetic algorithm to search for the best partition from all possible partitions which was demonstrated to perform well on a medium-size software system. Based on it, Parsa and Bushehrian used DAGC genetic algorithm [30], which evolves a new encoding scheme and a software environment, to overcome the limitation of large search space for SMCPs. In [32], two evolutionary algorithms, namely GGA and GNE, adopted the basic *MQ* as the single objective function were proposed by Praditwong and the experiments showed that GGA defeats a genetic algorithm with the string representation. Later, in [33], Praditwong et al. modeled the SMCPs as a multi-objective search problem, and two multi-objective evolutionary algorithms, namely MCA and ECA, were proposed.

Although the basic *MQ* has been widely used as the primary objective function and performs well in solving SMCPs, it still has some disadvantages. From the definition of the basic *MQ*, we can see that it simply takes the general relationship of edges between modules, namely intra-edges (intra-connectivity) and inter-edges (inter-connectivity), into consideration, and neglects the general rules in the process of software research and development.

According to the knowledge and experience of software design, a good software system must contain several independent functional clusters, which is easy for implementation, maintenance and management [1, 2,5,8,36]. To achieve this purpose, in order to improve the structure of large software systems, the following requirements should be considered:

(1) To be a reasonable structure of software systems, the direction of edges between different clusters should be as consistent as possible, which is good for constructing independent functional clusters and the maintenance and evolvement of systems.
(2) All global modules exist in the software system should be included in a single cluster. The number of global modules tends to decrease during the development of software design [2,31], which highlights their impacts on the process of software clustering. The operation of classifying all global modules into a single cluster makes the rest clusters be more independent.

Thus, to consider these requirements practically during the process of searching for software module clusters, we design a new modularization quality measure based on the structural similarity used in solving community detection problems [12,15], which is labeled as *MS*. Moreover, a set of evaluation functions based on software design rules are designed to evaluate the quality of partitions obtained by different measures. Then, in order to make a further comparison with the basic *MQ*, three optimization algorithms are employed to solve SMCPs. The efficacy of *MS* is tested on a set of problems extracted from real-world software systems with varying sizes. The experimental results show that *MS* performs well in solving SMCPs and can lead optimization algorithms to find clusters which are more consistent with the general rules in software design.

The rest of this paper is organized as follows. Section 2 describes the proposed modularization quality measure based on similarity. Section 3 introduces the details of three optimization algorithms employed. Section 4 gives the experiments. Finally, the conclusions and future work are presented in Section 5.

2. Similarity-based modularization quality measure

In SMCPs, module dependency graphs (MDGs), which are directed graphs, are used to represent the relationships between different software modules. In MDGs, a vertex stands for a module (e.g. functions, sources files) in software system and a link, or an arc, stands for the relationship (e.g. function calls) between two modules [21]. According to the characterization of edges, MDGs can be classified into two types, the one with and without weighted edges [32]. If an edge is associated with a positive number, called weights, then the MDG is weighted; otherwise, it is unweighted [32].

Suppose an MDG is labeled as $MDG=(V, E, w)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of n modules and $E \subseteq V \times V = \{(v_i, v_j)|v_i, v_j \in V \text{ and } i \neq j\}$ is the set of links between modules, and $w(v_i, v_j)$ is the weight of the edge between modules v_i and v_j. The weight is always larger than 0. For weighted MDGs, the weight of each edge is a positive number, while it