Hybrid Clustering Approach for Software Module Clustering

[1]Kishore C, [2]Dr. K. Ramani, [3]Anoosha G
[1,3]Assistant Professor, [2]Professor
[1,2]Dept. of IT, Sree Vidyanikethan Engineering College, Tirupati
[3]Dept. of CSE, Annamacharya Institute of Technology and Sciences, Tirupati
[1]mr.c.kishore@gmail.com , [2]ramanidileep@yahoo.com , [3]anua13@gmail.com

***Abstract*---Maintenance is one of the main software creation activities in terms of allocated resources. As existing software ages, newly developed systems are built to improve upon existing systems. As software evolves, modularization structure of software degrades and at one point it becomes a challenging task to maintain the software future. Recently, clustering techniques have been used to help with the issues of software evolution and maintenance. It is well known fact that a good modularized software system is easy to understand and maintain. Software Module Clustering is an important task during the maintenance of the software whose main goal is to achieve good modularized software. In recent time, this problem has been converted into search based software engineering problem. All previous work on software module clustering used a single objective formulation of the problem. That is, twin objectives of high cohesion and low coupling have been combined into a single objective called modularization Quality. We introduced hybrid clustering approach which improves modular structure of the software system. We present the results and comparing the results obtained with this from existing single objective formulation on 7 real world model clustering problems. The results of this empirical study prove that hybrid clustering approach produces significantly better solutions than the existing single-objective approach.**

***Index Terms*--- Search Based Software Engineering, Software Module Clustering, Modularization Quality.**

I. INTRODUCTION

Most of the interesting software systems are large and complex which is difficult to understand their structure. One of the reasons for this complexity is that source code contains many entities like classes, modules etc., that depend on each other in intricate ways (e.g., procedure calls, variable references. Once a software engineer understands a system's structure, it is difficult to preserve this understanding, because the structure tends to change during maintenance.

To solve the above problem, the reverse engineering research community has developed techniques to partition the software system's structure into meaningful subsystems called clusters. Generally a subsystem consists of a collection of collaborating source code resources that implement a feature or provide a service to the rest of the system. Source code resources that found in subsystems include classes, modules and possibly other subsystems.

Software module clustering involves partitioning of connected software system modules into group of clusters according to predetermined criteria. The criterion followed in this paper is that clusters are to be strongly connected internally (high cohesion) and weakly connected externally (low coupling) as shown in figure 1. Creating multiple clusters of high cohesion and low coupling is better than creating a single cluster of relatively low cohesion but zero coupling.

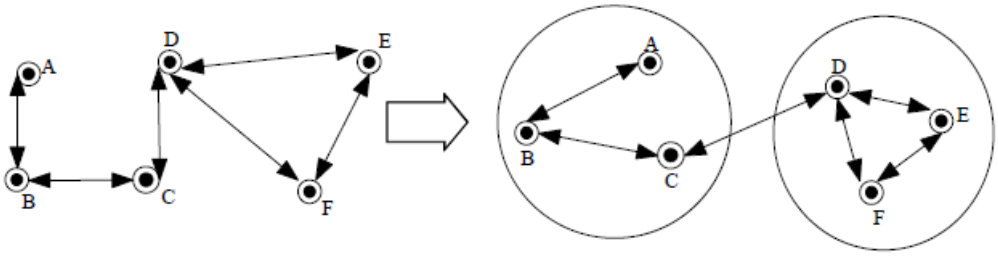

Fig. 1 Example of a possible clustering of a simple MDG

Software module clustering [10] is an important and challenging task in software engineering. It is widely believed that a well-modularized software system is easier to develop and maintain. There are so many ways to solve the module clustering problem. Mancoridis et al., [1] suggested the search-based approach to solve software module clustering, in this paper also we follows the search-based approach. In the search based approach, the attributes of a good modular decomposition are formulated as objectives, the evaluation of which as a "fitness function"[3] guides a search-based optimization algorithm. All previous works done on the software module clustering problem inspired by it have used a single-objective formulation of the problem. That is, the twin objectives of high cohesion and low coupling have been combined into a single objective called Modularization Quality (MQ).

This paper introduces hybrid clustering approach for software module clustering and presenting results that show how this approach can yield superior results than the existing software clustering approaches [1], [9].

The primary contributions of this paper are as follows:

Hybrid Clustering Approach for software module clustering is introduced.

An empirical study into effectiveness and performance of existing and proposed approaches is presented.

The rest of this paper is organized as follows: Section 2 presents Automated Software Module Clustering process. Section 3 presents background and related work on software module clustering. Section 4 introduces the Hybrid Clustering Approach for modularization. Section 5 results. Section 6 concludes.

II. AUTOMATED SOFTWARE MODULE CLUSTERING PROCESS

Figure 2 shows the process of software module clustering [6],[7]. The first step in the clustering process is to extract the module-level dependencies from the source code and store the resultant information in a database by using source code analysis tool. After all of the module level dependencies have been stored in a database, a script is executed to query the database, filter the query results, and produce, as output, a textual representation of the Module Dependency Graph (MDG). We define MDGs formally, but for now, consider the MDG as a graph that represents the modules (Classes) in the system as nodes, and the relations between modules as weighted directed graph.

Once the MDG is created, Bunch[2],[5] applies our clustering algorithms to the MDG and creates a partitioned MDG. The clusters in the partitioned MDG represent subsystems, where each subsystem contains one or more modules from the source code. The goal of Bunch's clustering algorithms is to determine a partition of the MDG that represents meaningful subsystems. After the portioned MDG is created, we use graph drawing tools such as Graphviz (dotty) to visualize the results.

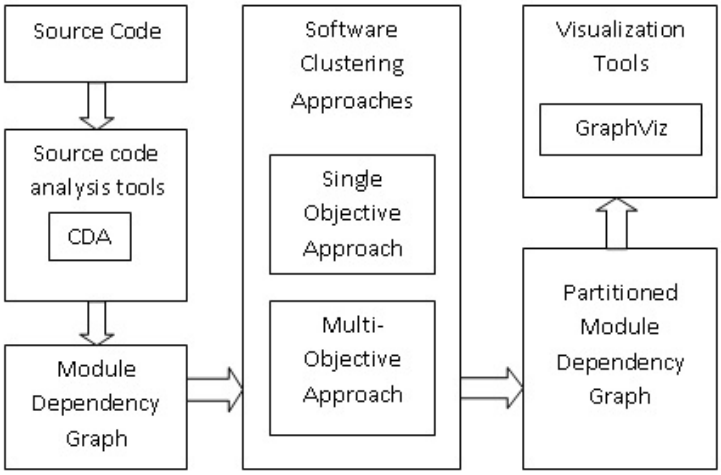

Fig. 2 Automatic Software Modularization Environment

III. BACKGROUND AND RELATED WORK

Software module clustering is an important and challenging task in software engineering. It is well known fact that a well-modularized structure of software system is easier to develop and maintain. To the software module clustering, many approaches have been applied successfully. Hill climbing [4] is one of the approaches and initiates the development of Bunch tool[2],[5] for automated software module clustering. Several other searching technologies [10] have been applied, including genetic algorithms. These experiments have all shown that other techniques are outperformed in both result quality and execution time by hill climbing.

Fitness function

The fitness function[3] need to be defined for formulate software engineering problems.One possible way to measure the fitness is to determine cohesion and coupling values for each of the clusters. The cohesion could be measured as:

$$\frac{N_i}{P_i} \quad (1)$$

Where Ni is the number of inner edges and Pi is the number of possible inner edges. The coupling in a similar way could be measure as:

$$\frac{N_o}{P_o} \quad (2)$$

Where No is the number of edges connected to outside of the cluster and Po is the possible number of connections that can be made to outer edges. We may then measure the fitness of each cluster by:

$$\frac{N_i}{P_i} - \frac{N_o}{P_o} \quad (3)$$

Our overall clustering fitness can be measured as the mean of all clusters:

$$\frac{\sum\left(\frac{N_i}{P_i} - \frac{N_o}{P_o}\right)}{n} \quad (4)$$

Where n is the number of clusters.

Modularization Quality

In this paper, we used the Modularization Quality [8] measure, introduced by Mancoridis et al. The intra-edges are those for which the source and target of the edge lie inside the same cluster. The inter-edges are those for which the source and target lie in distinct clusters. MQ is the sum of the ratio of intra-edges and inter-edges in each cluster, called the Modularization Factor (MF_k) for cluster k. MF_k can be defined as follows:

$$MF_k = \begin{cases} 0, & \text{if } i = 0, \\ \frac{i}{i+\frac{1}{2}j}, & \text{if } i > 0, \end{cases} \quad (5)$$

Where i is the weight of intra-edges and j is that of inter-edges, that is, j is the sum of edge weights for all edges that originate or terminate in cluster k. The reason for the occurrence of the term $\frac{1}{2}j$ in the above equation (rather than merely j) is to split the penalty of the inter-edge across the two clusters that connected by that edge. If the MDG is unweighted, then the weights are set to 1.

The MQ can be calculated in terms of MF as

$$MQ = \sum_{k=1}^{n} MF_k \quad (6)$$

Where n is the number of clusters.

IV. Hybrid Clustering Approach

The Hybrid Clustering Approach considers following set of objectives:

- The number of clusters should be maximum,
- Modularization Quality should be maximum,
- The sum of intra-edges of all clusters should also maximum,
- The sum of inter-edges of all clusters should be minimum,
- The number of isolated clusters should also minimize.

The intra-edges, inter-edges, and the MQ are used to measure the quality of the system partitioned. An isolated cluster is a cluster that contains only one module. With the experience we concluded that isolated clusters are uncommon on well-modularized decompositions and so they are deprecated in the Hybrid Clustering Approach by including the number of isolated clusters and an objective to be minimized. The aim of the Hybrid Clustering Approach measure is to capture the good clustering attributes. That is, it will have maximum amount of cohesion (intra-edges are maximizing) and minimal amount of coupling (inter-edges are minimizing). However, it should not put all modules into a single cluster (maximizing the number of clusters) and not produce a series of isolated clusters (so the number of isolated clusters is minimized).

Since MQ is a well-studied objective function, this is also included as an objective for Hybrid Clustering Approach. This is one of the attractive aspects of a Hybrid Clustering approach; one can always include other candidate single objectives as one of the multiple objectives to be optimized. The MQ value will tend to increase if there are more clusters in the system, so it also makes sense to include the number of clusters in the modularization as an objective.

To illustrate the Hybrid Clustering Approach, consider the Module Dependency Graph in Figure 3. The set of objective values for Hybrid Clustering Approach are as follows:

- Sum of Intra-edges of all clusters (cohesion): 9,
- Sum of Inter-edges of all clusters (coupling): -6,
- The total number of clusters: 3,
- Modularization Quality: 2.207,
- The total number of isolated clusters: 0.

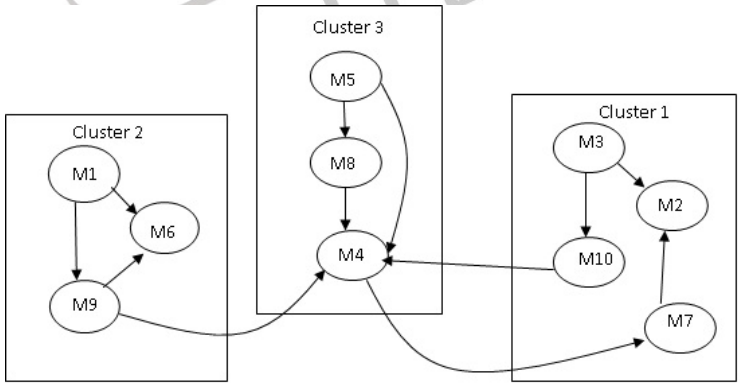

Fig. 3 The Module Dependency Graph after clustering by Hybrid Clustering Approach.

V. Experimental Results

Experiment was conducted on seven real-world clustering systems which are shown in Table 1. Each MDG can be executed 20 times independently. There are two different ways to get the average of MQ value. One way by using the hill-climbing algorithm, this gives only one solution in each run. The average of MQ can be calculated indirectly from the solutions. However, the two-archive algorithm produces the set of solutions. The solution with the highest MQ is chosen to be the best solution in each run.

Thus, the average of the MQ of obtained solutions from the Hybrid Clustering Approach is estimated using the representatives from 20 runs. This is the method to collect the MQ values from the experiment.

TABLE 1: THE SYSTEMS STUDIED

Name	Nodes	Edges	Description
Mutunis	20	57	An operating system for educational purposes written in the turning language
Ispell	24	103	Software for spelling and typographical error correction in files
Rcs	29	163	Revision Control System used to manages multiple revisions of files
Bison	37	179	General-purpose parser generator for converting grammar description into c programs
Grappa	86	295	Genome rearrangement analyzer under parsimony and other phylogenetic algorithms
Bunch	116	365	Software Clustering tool (Essential java classes only)
Incl	174	360	Graph drawing tool

Table 2 present the result comparing Hybrid Clustering Approach and Hill-Climbing approaches by considering MQ value as an assessment criterion. There is a good evidence to suggest that the Hybrid Clustering Approach outperformed the Hill-Climbing approach. That is, the Hybrid Clustering Approach gives higher values for MQ in five from seven problems.

TABLE 2 COMPARISON OF SOLUTIONS FOUND BY THE HYBRID CLUSTERING APPROACH AND HILL-CLIMBING APPROACHES USING MQ VALUE AS AN ASSESSMENT CRITERIA

Name	Hybrid Clustering Approach		Hill-Climbing	
	Mean	STD	Mean	STD
mtunis	2.642	0.158	2.216	0.158
ispell	2.435	0.143	2.159	0.166
rcs	2.994	0.246	2.186	0.354
bison	2.556	0.108	2.646	0.103
grappa	14.016	0.132	12.478	0.103
bunch	10.390	0.207	9.048	0.001
incl	12.602	0.001	14.041	0.070

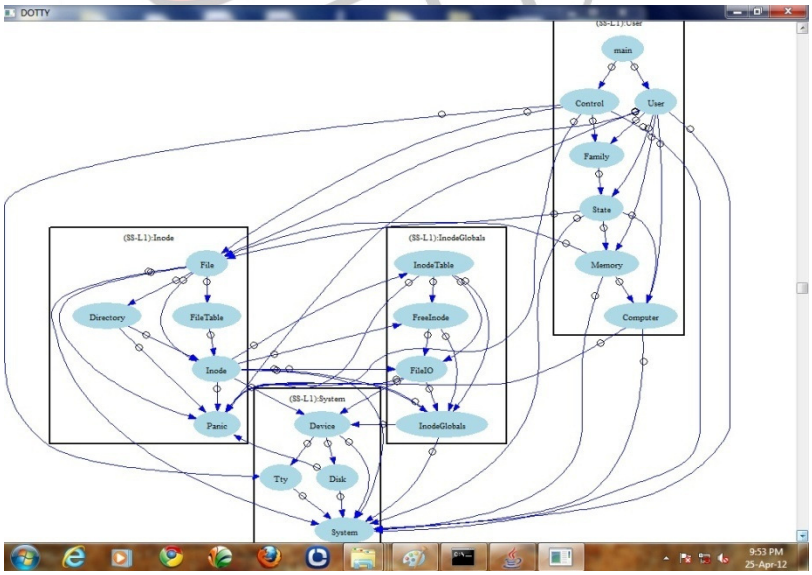

Fig. 4 Clustered MDG of mtunis system using hill-climbing algorithm

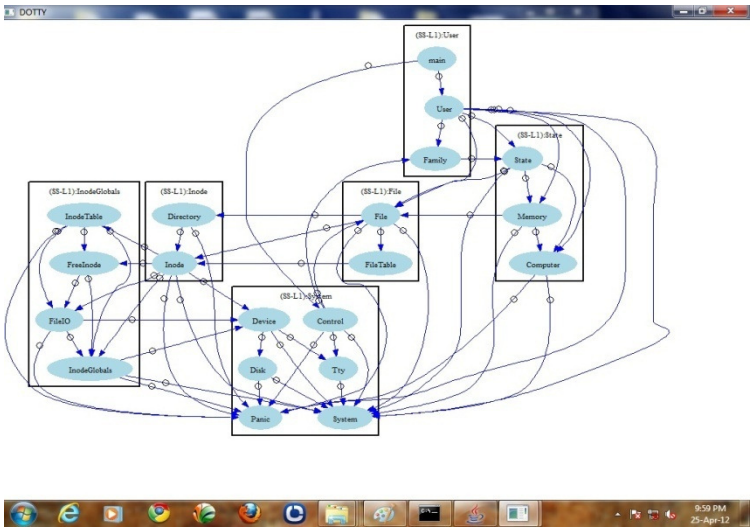

Fig. 5 Mtunis system applying MCA approach

VI. Conclusion and Future Work

This paper introduces the Hybrid Clustering approach to software module clustering and comparing the results obtained with this from the existing Hill-Climbing Approach on 7 real world module clustering problems. The result indicated that Hybrid Clustering approach produces superior results than the existing Hill-Climbing Approach.

Future work could be considering other possible objectives for better modularization.

References

[1] Kata Praditwong, Mark Harman, and Xin Yao, "Software Module Clustering as a Multi-Objective Search Problem", Vol. 37, No. 2, March/April 2011.

[2] B.S. Mitchell and S. Mancoridis, "On the Automatic Modularization of Software Systems Using the Bunch Tool," IEEE Trans. Software Eng., vol. 32, no. 3, pp. 193-208, Mar. 2006.

[3] Baresel, H. Sthamer, and M. Schmidt. Fitness function design to improve evolutionary structural testing. In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pages 1329–1336, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[4] K. Mahdavi, M. Harman, and R.M. Hierons, "A Multiple Hill Climbing Approach to Software Module Clustering," Proc. IEEE Int'l Conf. Software Maintenance, pp. 315-324, Sept. 2003.

[5] S. Mancoridis, B.S. Mitchell, Y.-F. Chen, and E.R. Gansner, "Bunch: A Clustering Tool for the Recovery and Maintenance of Software System Structures," Proc. IEEE Int'l Conf. Software Maintenance, pp. 50-59, 1999.

[6] C.Kishore, Asadi Srinivasulu, "Multi-Objective Approach for Software Module Clustering", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, Issue 2, March 2012.

[7] C.Kishore, Asadi Srinivasulu, Anusha G, "Comparative Study of Software Module Clustering Algorithms: Hill-Climbing, MCA and ECA", International Journal of Advanced Research in Computer Engineering and Technology,

[8] Chandrakanth P, Anusha G, Kishore C, "Software Module Clustering using Single and Multi-Objective Approaches", International Journal of Advanced Research in Computer Engineering and Technology, Vol 1(10), December 2012.

[9] Chandrakanth P, Anusha G, Kishore C, "ESBA: Enhanced Search Based Approach for Software Module Clustering", International Journal of Advanced Research in Computer Engineering and Technology, Vol 2(10), October 2013.

[10] https://www.cs.drexel.edu/~spiros/bunch/MitchellPhD.pdf