ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, Issue 10, October 2013

# ESBA: Enhanced Search Based Approach for Software Module Clustering

**Chandrakanth P**
*Department of CSE*
*YITS*

**Anusha G**
*Department of CSE*
*AITS*

**Kishore C**
*Department of IT*
*SVEC*

***Abstract---- Most of the software systems are very large and complex which is difficult to understand their structure. Once software engineer understands the system structure, difficult to preserve this understanding due changing their structure during the maintenance. Software Module Clustering is an important technique to solve the above said problem. In previous, so many approaches are introduced for software module clustering problem. In which those algorithms are not performed well for the software having large number of classes. We introduce an approach called Enhanced Search Based approach which can perform clustering for large software systems. This approach can generate Module Dependency Graph (MDG) having higher Modularization Quality (MQ) based on the high cohesion and low coupling technique. To evaluate the performance of the proposed approach, we apply this approach on seven large software systems. The result shows that Enhanced Search Based approach can perform well than the other approaches.***

***Keywords---- Clustering, Cohesion, Coupling, Search Based Software Engineering, MDG, MQ.***

## I. INTRODUCTION

Interesting software systems are very and complex, as consequences which is difficult to understand their structure. One of the reasons for the complexity is having so many entities in the source code that depends on each other in an intricate way. To solve the above problem, the Reverse Engineering Research Community has developed a technique that decomposes (partition) the large software system into meaningful subsystems called clusters [8]. Subsystem provides high level information about the software components, their interfaces and their interconnections. This turns to introduction of Software Module Clustering Technique.

There are many ways to approach the module clustering problem. Following Mancoridis et al., who first suggested the search-based approach [2] to module clustering, this paper 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. In previous work, Genetic Algorithm was used, but there is no evidence shows that genetic algorithm provides optimal results for large software systems. So, to overcome this problem, we introduce Enhanced Search Based approach improving Modularization Quality (MQ). By this approach we can increase the MQ value. Increasing MQ value will get optimal results for large software systems also.

## II. BACKGROUND AND RELATED WORK

There are many approaches are introduced for software module clustering. Several search algorithms are also applied like genetic algorithm for Software Module Clustering. These techniques are outperformed to exhaustive Search Approach.

In order to perform software module clustering, we need Module Dependency Graph (MDG). In MDG, modules are the nodes and relationships are the edges. Good Modularized MDG having highest Modularization Quality (MQ) [1] value. MQ is a sum of Modularization Factor (MF) which is the ratio of intra-edges and inter-edges. Modularization Factor ($MF_k$) for cluster k can be defined as follows:

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

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.

MQ can be calculated in terms of MF as

$$MQ = \sum_{k=1}^{n} MF_k$$

Where n is the number of clusters.

### A. Genetic Algorithm

Genetic algorithm [4], [5], [6] uses the concept of population and recombine. Of all optimization algorithms, genetic algorithm [7] is the most widely applied search based technique in Search Based Software Engineering. Figure 1 shows the generic genetic algorithm. An iterative

process is executed, initialized by randomly chosen population. The iterations are called generations and the members of a population are called chromosomes. When a population satisfies some pre-determined condition or certain number of generations has been exceeded, the process will be terminated. On each generation, some members of the population are recombined, crossing over elements of their chromosomes. Fractions of the offspring of this union are mutated and, from the offspring and the original population a selection process is used to determine the new population. Crucially, recombination and selection are guided by the fitness function; fitter chromosomes having a greater chance to be selected and recombined.

```
Set generation number, m=0
Choose the initial population of candidate solutions, P(0)
Evaluate the fitness for each individuals of P(0), F(Pi(0))
Loop
   Recombine: P(m) := R(P(m))
   Mutate: P(m) := M(P(m))
   Evaluate: F(P(m))
   Select: P(m+1) := S(P(m))
   m := m+1
   exit when goal or stopping condition is satisfied
end loop;
```

Fig. 1 A generic genetic algorithm

There are many variations on this overall process, but the crucial ingredients are the way in which the fitness guides the search, the recombine and the population based nature of the process. There is also a variation of genetic algorithms, called genetic programming. Genetic programming has been used in SBSE to form formula that captures predictive models of software projects and in testing.

## III. Software Module Clustering Process

Fig 2 shows the process of software module clustering. 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) [9]. 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 [10].

Once the MDG is created, Bunch applies Enhanced Search 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.

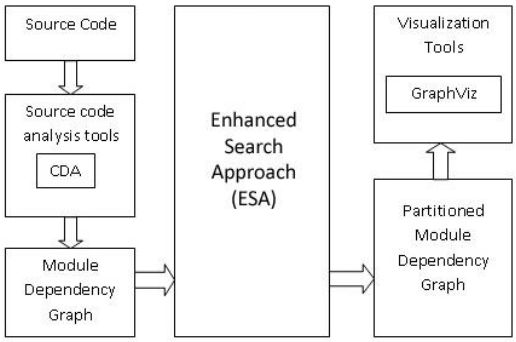


Fig: 2 Automatic Software Modularization Environment

## IV. Enhanced Search Based Approach:

Enhanced Search Based approach applies ideas from the theory of natural selection to navigate through large search spaces effectively. This approach found to overcome some of the problems of multi-objective search approaches. This approach works by measuring MQ for all partitions in Module Dependency Graph. The partition which is having higher MQ value it returns. Fig 3 shows enhanced search algorithm to perform clustering on large software systems. This algorithm can return MDG having height MQ value.

```
EnhancedSearch(MDG m)
        Let A be the set of all valid partitions of m
        Let S be the best partition of M, initialized to ;
        Let MAXmq = -∞
        foreach partition sεA do
                Let MQs = MQ(s)
                if MQs > MAXmq then
                        S←s
                        MAXmq←MQs
                end
        end
return (partition S)
```

Fig. 3 Enhanced Search Algorithm

## V. Results

To evaluate the performance Enhanced Search Based approach, we applied this algorithm to 17 different systems which have shown in table 1. These systems are not necessarily "degraded" systems in terms of their modular structure, but they have been studied widely by other researches to evaluate their algorithms for module clustering

and so they denote reasonable choices for comparison.

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 |
| icecast | 60 | 650 | Streaming media server based on the MP3 audio codec |
| gnupg | 88 | 601 | Complete implementation of the OpenPGP Internet standard |
| inn | 90 | 624 | Unix news group software |
| bitchx | 23 | 729 | Open source IRC client |
| xntp | 111 | 729 | Time synchronization tool |
| exim | 23 | 1255 | Message transfer agent for use on Unix systems connected to the Internet |
| mod_ssl | 135 | 1095 | Apache SSL/TLS Interface |
| ncurses | 138 | 682 | Software for display and update of text on text-only terminals |
| lynx | 23 | 1745 | Web browser for users on UNIX and VMS platforms |
| nmh | 198 | 3262 | Mail client software |

This Section presents the results of the experiment that compare MQ value obtained for Enhanced Search algorithm and Genetic Algorithm. In Table 2, the results from two approaches were comparable. There is a good evidence suggest that our approach is outperformed the genetic approach. Fig 4 shows MDG of Mtunis System after applied our approach.

TABLE 2
COMPARISON OF MULTI-OBJECTIVE APPROACH AND MCA USING MQ VALUES AS AN ASSESSMENT CRITERIA

| Name of the System | Enhanced Search Algorithm | Genetic Algorithm |
|---|---|---|
| mutunis | 2.694 | 2.294 |
| ispell | 2.562 | 2.269 |
| rcs | 2.896 | 2.145 |
| bison | 2.654 | 2.416 |
| grappa | 12.743 | 11.586 |
| bunch | 13.387 | 12.145 |
| incl | 13.694 | 11.811 |
| icecast | 3.629 | 2.401 |
| gnupg | 7.210 | 6.259 |
| inn | 8.693 | 7.421 |
| bitchx | 3.213 | 3.572 |
| xntp | 8.965 | 6.482 |
| exim | 6.124 | 5.316 |
| mod_ssl | 9.847 | 8.832 |
| ncurses | 11.724 | 10.211 |
| lynx | 951 | 3.447 |
| nmh | 7.589 | 6.671 |

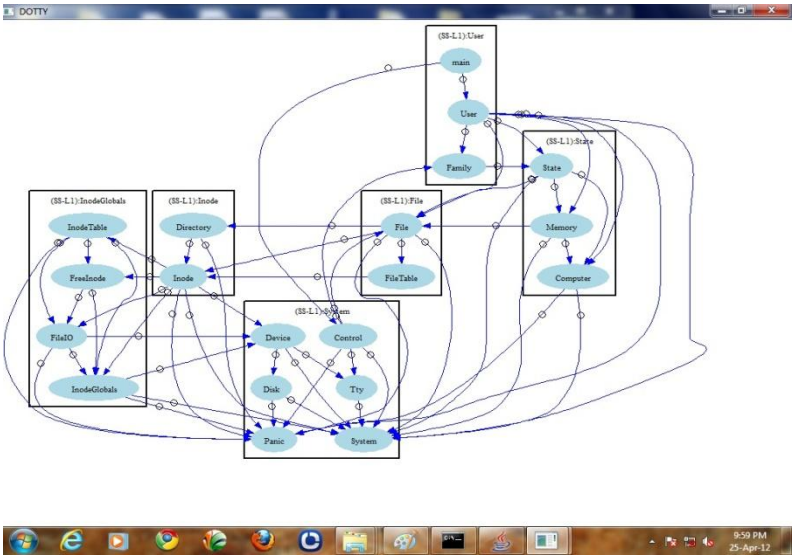

Fig. 4 MDG of Mtunis System after applying Enhanced Search Based approach

## VI. CONCLUSIONS

This paper introduces the Enhanced Search Based approach to software module clustering and presents the results for the application of this technique, comparing the results obtained with those from the existing genetic algorithm on 17 real world module clustering problems. The result indicated that Enhanced Search Based approach produces superior results for large software systems than the existing approaches.

## REFERENCES

[1] 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.

[2] M. Harman and B. F. Jones. Search based software engineering. Information and Software Technology, 43(14):833–839, Dec. 2001.

[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] M. Bowman, L. Briand, and Y. Labiche, "Multi-Objective Genetic Algorithms to Support Class Responsibility Assignment," Proc. 23rd IEEE Int'l Conf. Software Maintenance, Oct. 2007

[5] D. Doval, S. Mancoridis, and B.S. Mitchell, "Automatic Clustering of Software Systems Using a Genetic Algorithm," Proc. Int'l Conf. Software Tools and Eng. Practice, Aug.-Sept. 1999.

[6] L.C. Briand, J. Feng, and Y. Labiche, "Using Genetic Algorithms and Coupling Measures to Devise Optimal Integration Test Orders," Proc. 14th Int'l Conf. Software Eng. and Knowledge Eng., pp. 43-50, 2002

[7] D. Doval, S. Mancoridis, and B.S. Mitchell, "Automatic Clustering of Software Systems Using a Genetic Algorithm," Proc. Int'l Conf. Software Tools and Eng. Practice, Aug.-Sept. 1999.

[8] Brian S. Mitchell, A Heuristic Search Approach to Solving the Software Clustering Problem, March 2002.

[9] 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,

[10] 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.